CMPE 257 Project Report

****

Classification of Disaster Tweets Using NLP

Group 6

Sushant Mathur - 014489865

Anushree-015247388

Amritpal Sidhu - 015013518

Gayatri Yadkikar - 014562574

Venkatalakshmipravallika Ramachandruni - 015297308

[**Introduction**](#_deazf53dt1nc) **3**

[**Literature Review**](#_tnckwld01139) **3**

[Vector Representation of Words with GloVe[3]](#_c3a2yo8jbwua)4

[BERT[4]](#_ri34k1yifja1)5

[Multilingual Universal Sentence Encoder(Transformer)[11]](#_ntmk4e1pimu2)6

[**Problem Formulation**](#_lgvvqbvp5k2q) **7**

[**Data Exploration & Preprocessing**](#_nix4asep9ifb) **8**

[**Data Visualization**](#_e8rsendaln6y) **10**

[**Model Selection**](#_gh0ff8kk70gw) **12**

[**Evaluation, Result Analysis & Visualization**](#_i46ed0k54skj) **12**

[Logistic Regression:](#_hb9y7uiqd2xq) 12

[Support Vector Machine](#_eaxjeve0y4zd) 14

[Random Forest Classification](#_b0hxwb8r317y) 17

[XGBoost](#_euawzg9oxnah) 17

[Glove pre trained word embeddings (Twitter data)](#_s2j6rskx3dvi) 19

[Glove pre trained word embeddings (Wiki data)](#_50dpbl3oq593) 22

[Google pre trained Word2Vec model](#_g6zysvc8jhj8) 24

[DistilBERT Model](#_l8elhrfoqrqt) 27

[**Sample Predictions**](#_ru29xrwa1adr) **31**

[**Summary of the Models**](#_558u5kx6bzdc) **32**

[**References**](#_ltiy1qf2omsr) **33**

# 

# **Introduction**

Social media has become omnipresent in today’s society. With data becoming more accessible than ever and companies like Amazon and SpaceX launching a satellite based internet solution that will provide even more access, the reach of social media is only going to increase over the next few years. What people post on social media can be used to identify what is going on around a region with great accuracy.

An analysis of what people post on social media can be done to help in things like disaster management, as when something unforeseen occurs people tend to post information on websites like Twitter, Facebook etc. We can retrieve information of events like earthquakes, fires, tsunamis and other natural disasters based off of tweets by analysing them and classifying them as a disaster tweet or not a disaster tweet, which is the topic of our project for class CMPE 257.

We have analysed different approaches and implemented them for the problem of classifying disaster tweets and compared the performance provided by each of them. We took part in the Kaggle competition to see what was the best we could do with various machine learning and deep learning approaches.

These approaches can be used to analyse tweets during a live disaster and help alert authorities so they can better allocate resources and be informed of new information as it comes in.

# **Literature Review**

When researching approaches to dealing with natural language text data, we came across a paper by Zhang and Wallace[1] that, somewhat, acts as a guide to understanding parameter tuning when using CNNs.The paper describes the architecture of neural networks and the need to first embed the textual information into vectorized form.After creating the word embedding neural networks are typically architectured with convolutional layers followed by max pooling layers, which are needed since sentence sizes can vary and this affects the output size of the convolutional filters. Use of regularization by means of dropout, or l2 norm constraint, can also be done.

Zhang and Wallace[1] compared word2vec and GloVe word embeddings on multiple datasets, and found word2vec seemed to score slightly higher than GloVe. The paper also compared how different filter sizes, feature maps, activation functions, pooling strategies, and use of regularization affects the CNN for the various text datasets. This paper proved helpful in understanding using pre-trained word embeddings, and how to tune a neural network for them.

RNNs used to be difficult to train text classification, however Dai and Le[2] show that it can be done with an LSTM RNN. In their paper they propose using a sequential autoencoder to improve the LSTM. The autoencoder is constructed out of a RNN as an encoder that takes in the input, which feeds to a decoder RNN. This paper gave us inspiration to try LSTM for our NLP project.

## Vector Representation of Words with GloVe[3]

Glove stands for Global Vectors for Word Representations. The idea is to represent words in an n dimensional vector space. The euclidean distance between vectors in this space can give us important information about the relationship between the words under consideration. If the distance is less then it generally means that the words have similar meanings.

The need for multiple dimensions is that a single dimension is not enough to capture the nuance in the words. For example, man and woman are very closely related and we might assume that they should be closer together, however even though they both talk about humans, their meanings in English can be the opposite. Similarly we notice this pattern in the words King and Queen.

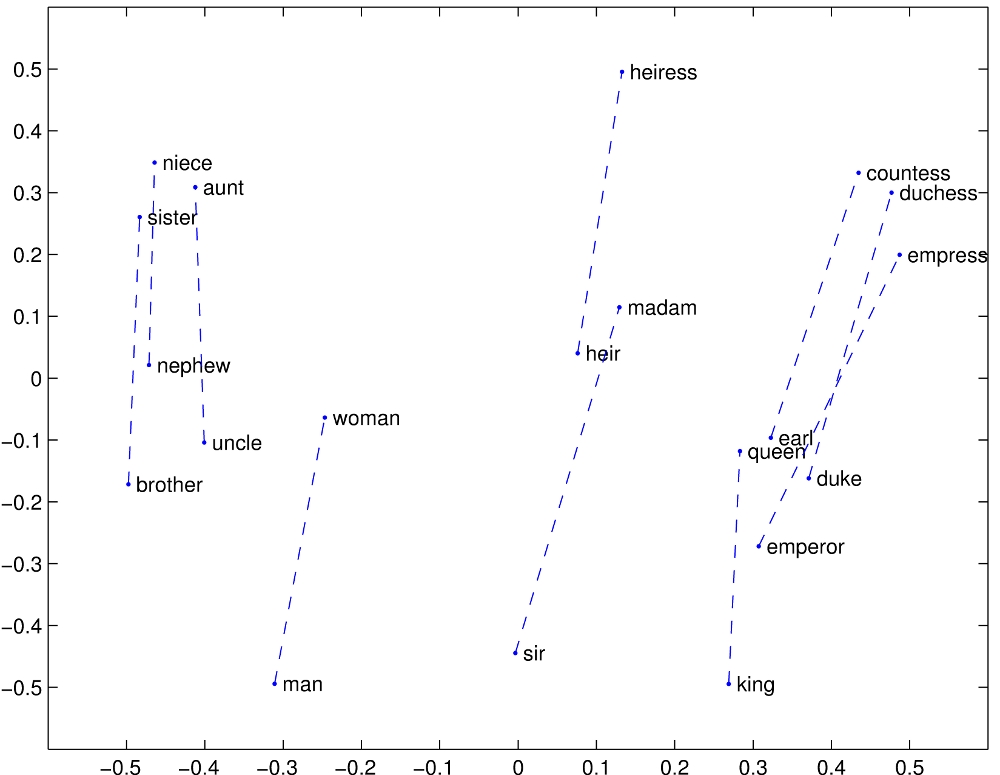


Fig1 . Distance Between Words in 2 dimensions

GloVe can be trained on a corpus of words from which it can generate the vectors for each word in the vocabulary. We can also use pre-trained word vectors. Different corpus will give us different vectors. For example, a model trained on Twitter will have different results than a model trained on Wikipedia. Twitter has a lot of informal language and will contain a lot of abbreviations whereas Wikipedia will be more formal.

## BERT[4]

BERT stands for Bidirectional Encoder Representations from Transformers. It’s a super supervised system for pre-training applied to NLP. BERT is the first deeply bidirectional, unsupervised language representation and is pre trained using only a plain text corpus.

We can represent words as contextual and non-contextual.In contextual representation, the other words and sequences in the sentences have an influence on the vector representation. In a non contextual representation we have the same representation for the same word regardless of what context it is used in.

For example consider the sentences “I am reading a book” and “Let’s book the movie for tonight”. The word book means different things in both the sentences. In the first one it means something to read whereas in the second they are talking about reservations. The advantage of BERT is that it will represent the word bank with it’s previous and next context. This makes it deeply bidirectional.

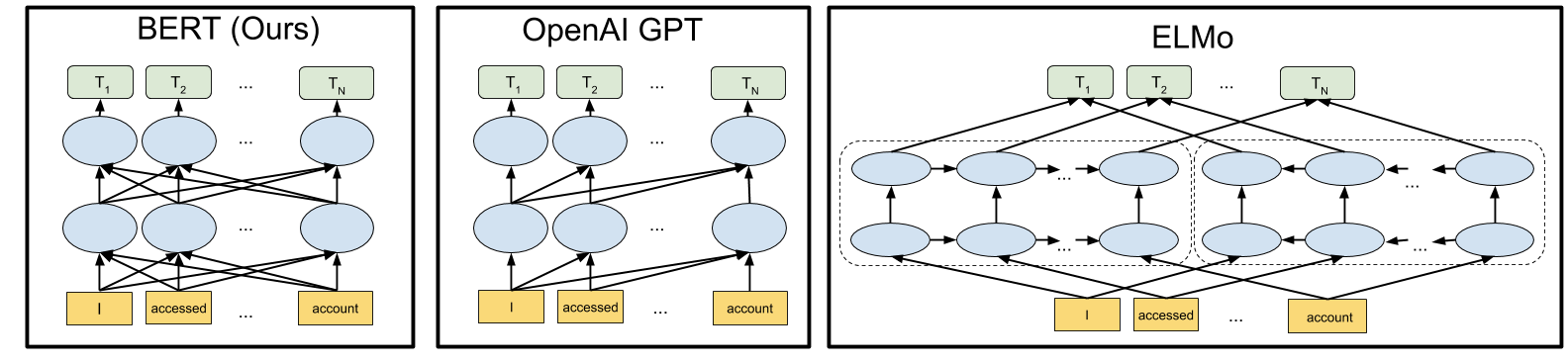


Fig 2. BERT Architecture

## Multilingual Universal Sentence Encoder(Transformer)[11]

It was developed by developers at Google in 2019. A Universal Sentence Encoders encodes sentences to fixed length vectors (The size is 512 in the case of the Multilingual Encoder).The encoders are pre-trained on several different tasks: Two architectures are in use in the encoders: Transformer and Deep Averaging Networks. Transformer uses a "self attention mechanism" that learns contextual relations between words and (depending on model) even subwords in a sentence. Not only a word , but its position in a sentence is also taken into account (like positions of other words). There are different ways to implement the intuitive notion of "contextual relation between words in a sentence" ( so, different ways to construct "representation space" for the contextual words relation). If several "ways" are implemented in a model at the same time: the term "multi head attention mechanism" is used.

Transformers have 2 steps. Encoding: read the text and transform it into a vector of fixed length, and decoding: decode the vector (produce prediction for the task). For example: take sentences in English, encode, and translate (decode) sentences in German.

For our model we need only an encoding mechanism: sentences are encoded in vectors and supplied for classification to Support Vector Machine.

# **Problem Formulation**

Our project consists of trying to read raw text data that would be a tweet that has been recorded from the microblogging site “Twitter”. The labels have been provided as 0 and 1. 0 indicating that the tweet is not a disaster and 1 indicating that the tweet is talking about a disaster.

We will use various machine learning and deep learning techniques that have been taught in class throughout the course. We will also use text preprocessing to clean the data as a lot of tweets will contain punctuation and links that do not provide any meaningful information towards their classification.

This classifier can then be used on live data during a disaster to pull relevant tweets and pick up important information from them to aid in disaster relief efforts. News agencies and disaster relief organizations can use such a system that programmatically pulls tweets and classifies them for their benefit.

# **Data Exploration & Preprocessing**

The dataset contains tweets with a label which tells you if they are really a disaster (1) or not (0). It is split into training (7613 tweets) and testing (3263 tweets).The attributes of each tweet are:

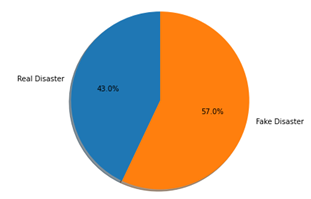
· Id - unique identifier of each tweet

· text -Tweet from the user

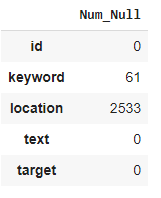
· keyword - Keyword from the tweet describing the disaster type

· location - Location from where the tweet was sent

· target - denotes whether a tweet is about real disaster or not.



There are missing values in the keyword and location columns.



**Preprocessing text data**

It is crucial to remove the noise and reduce the complexity of data to achieve a higher success rate. We have done the below pre-processing to the tweet data.

– Converting text to lowercase.

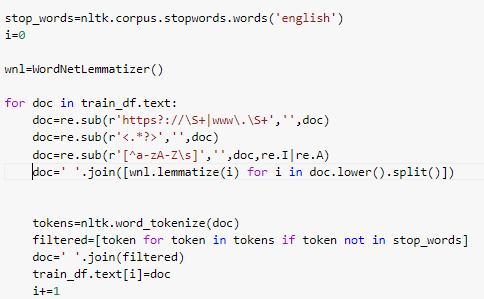
– Removal of mentions.

– Removal of URLs and Hyperlinks.

– Removal of punctuations.

– Removal of stop words.

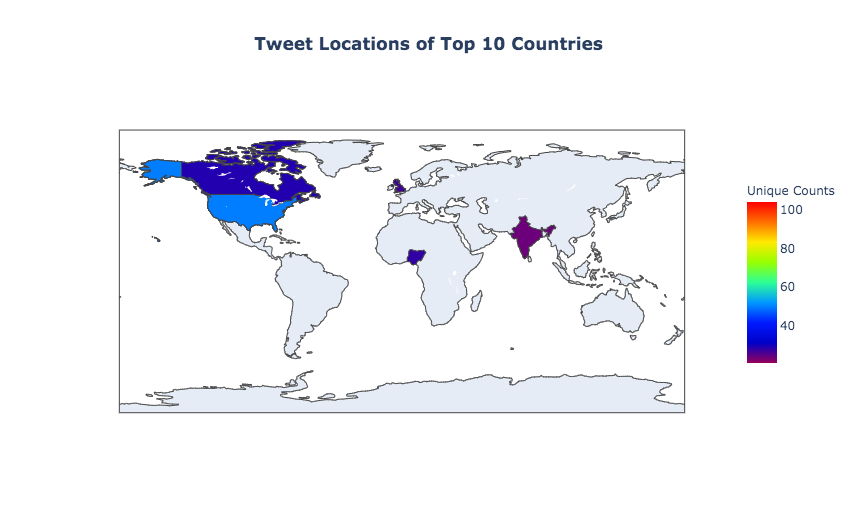
– Lemmatization



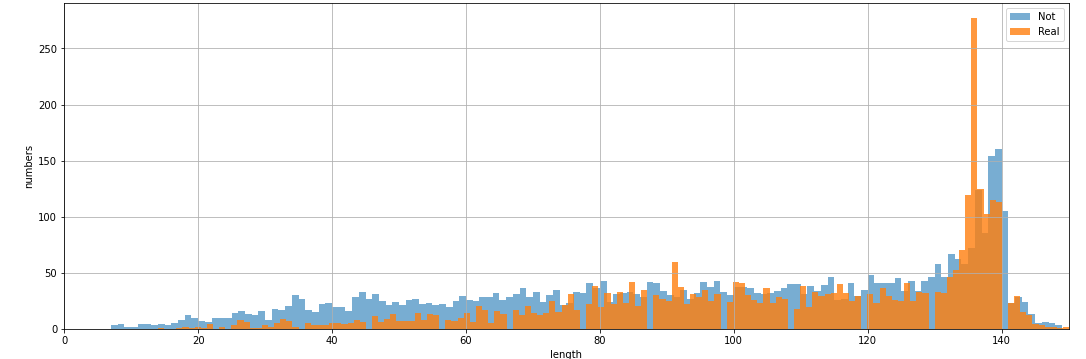
# 

# **Data Visualization**

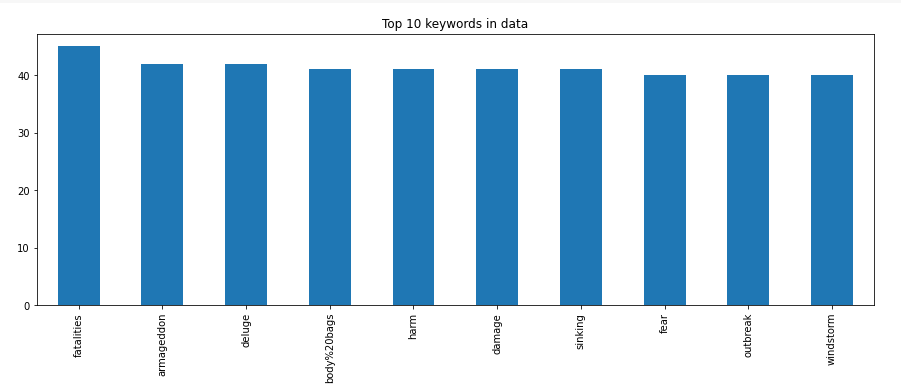
Top 10 countries with the most tweets with the USA being on the top.They have many unique values and a large number of locations have null values. Hence could not be used as a feature.



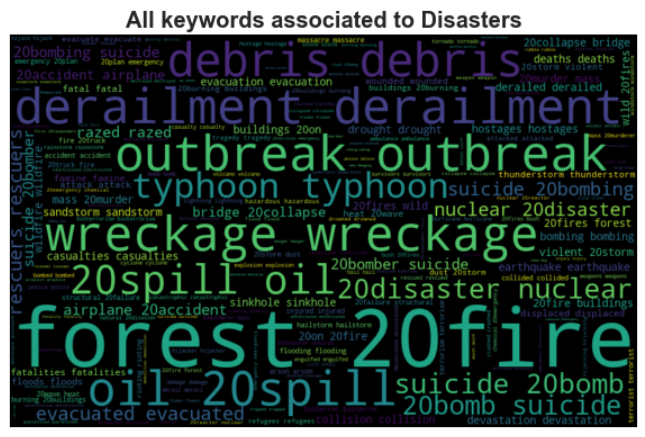
This below graph compares the length of the real and fake disaster tweets. It is a plot of the number of tweets against the length of each tweet.



These are the top 10 keywords in the data along with its frequency.



Below is a visual representation of a word cloud with 100 common words of a real disaster where size of each word indicates its frequency or importance.



# **Model Selection**

The baseline features are the words in tweets, after preprocessing to remove capitalization, punctuation and stopwords. We experimented with different classification models and feature selection methods. 20% of the data was held out as a validation set to use for experiments, including parameter optimization.We assessed eight different classification models that have been successful in similar work: Logistic Regression, SVM, random Forest, XG Boosting, Bidirectional LSTM with different pre trained word embeddings and DistilBERT. We experimented with different word vectorization techniques like TF IDF and Countvectorizer and found TF IDF giving relatively better accuracy in the validation set. Each model was evaluated on the validation data with the SVM yielding the best F1 performance.

# **Evaluation, Result Analysis & Visualization**

We initially experimented with different classification models and feature selection methods.

## Logistic Regression:

We decided to use a simple bag of words approach with logistic regression because judging by the dataset, we figured that the importance of certain words holds a lot more importance than others. For example, using “earthquake” in a tweet increases the probability of the tweet being about a disaster remarkably. A simple logistic regression working with the words we have in our vocabulary of training data.

We got a test accuracy of 80.5% for this approach on Kaggle which gave a rank of 1023 out of 2617 teams.

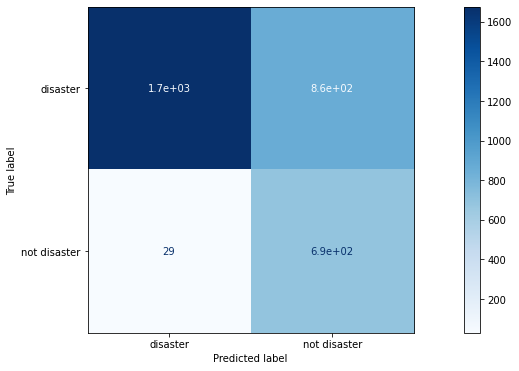
We needed to tune the hyperparameters to see what best fits our data. After using sklearn’s grid search CV, these are the parameters it gave us and the same were used to train the classifier.

For TF-IDF:

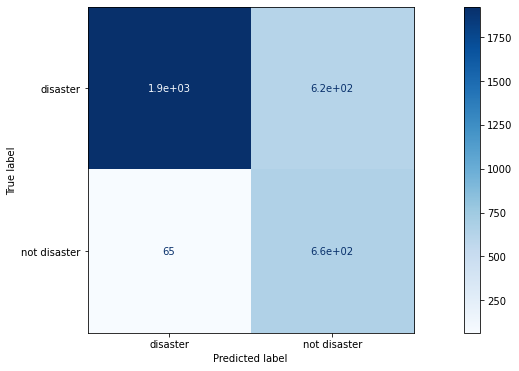
Best: 0.794580 using {'C': 1.0, 'penalty': 'l2', 'solver': 'liblinear'}

For Count Vectorized:

Best: 0.803582 using {'C': 10, 'penalty': 'l2', 'solver': 'newton-cg'}



Confusion matrix for Count Vectorized Logistic Regression



Confusion matrix for TF-IDF Logistic Regression

Classification Report:

precision recall f1-score support

not disaster 0.98 0.95 0.97 1318

disaster 0.93 0.98 0.96 966

accuracy 0.96 2284

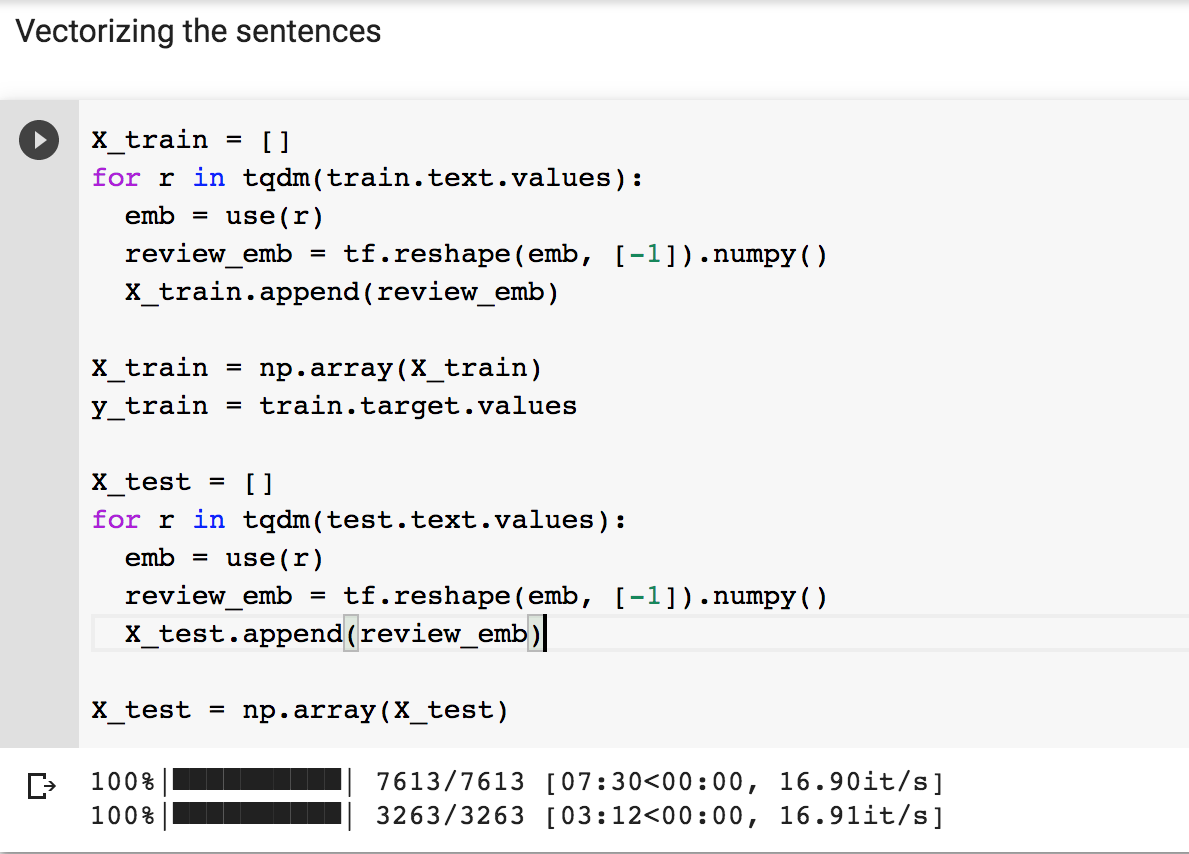
macro avg 0.96 0.96 0.96 2284

weighted avg 0.96 0.96 0.96 2284

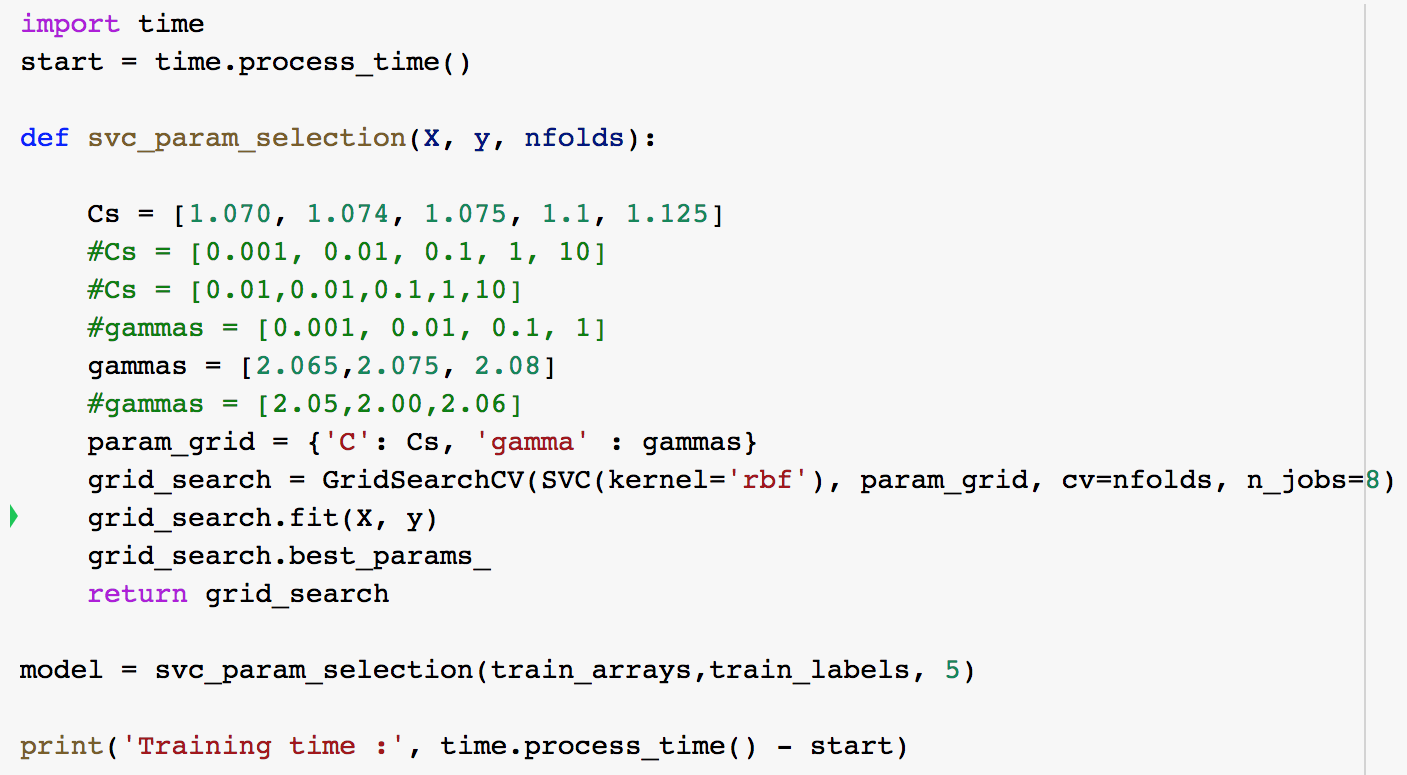
## Support Vector Machine

A Support Vector Machine (SVM) is a discriminative classifier which intakes training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples.It uses a technique called the kernel trick to transform data and then based on these transformations it finds an optimal boundary between the possible outputs.

In our project we made use of a Universal sentences encoder(**Transformer**) from tensorflow\_hub, which is pre-trained model to vectorize the sentences, and then SVM with the RBF(Radial Basis Function) kernel to do the classification.In the transformer we feed apply the raw text to the transformer, which transform every string to fixed length 512 dimensional vector. Then this vector is supplied to SVM to receive the output.



We used GridSearchCV, a library function from sklearn’s model selection package . The reason behind using this is it becomes difficult to manually change the hyperparameters and fit them on training data every time. **GridSearchCV** helps to loop through the predefined hyperparameters and fit the model on the training set.



After implementing the model, we got the below best hyperparameters :

**Best parameters**:- {'C': 1.125, 'gamma': 2.075}

We got the accuracy of this model as **83.7%.** After submitting the results on kaggle , we got the score as 0.8364 and the leaderboard standing is **321** out of **2369** teams .

Apart from this we also implemented SVM using a linear kernel which gave the accuracy of 80%.

The results of the SVM are as follows:

**Classification Report:**

precision recall f1-score support

0 0.81 0.92 0.86 1276

1 0.88 0.73 0.79 1008

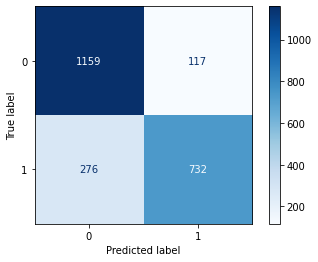
accuracy 0.83 2284

macro avg 0.84 0.82 0.83 2284

weighted avg 0.84 0.83 0.83 2284

As the support is constant ,so the reported score for training data is stable and the data is balanced and there is no need to sampling or rebalancing the data.

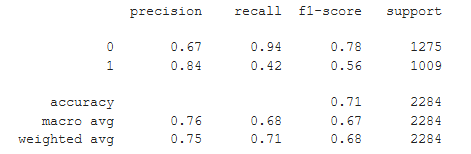
**Confusion Matrix:**



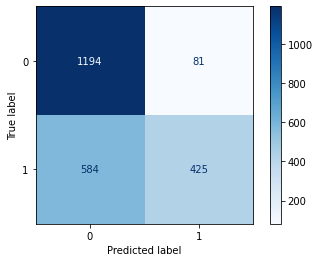
## Random Forest Classification

We also tried a random forest classifier using count vectorization and TF-IDF transformation. This strategy resulted in an accuracy of 70.8%.

**Classification Report:**



**Confusion Matrix:**



# 

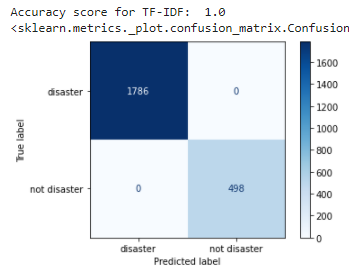
## XGBoost

XGBoost is decision tree-based ensemble machine learning algorithm which implements the gradient boosting framework.It uses a gradient descent algorithm to minimize the loss when adding new models.

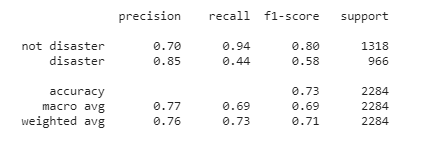
Two main reasons to use XGBoost are the speed and model performance. It is really fast when compared to other implementations of gradient boosting and also dominates structured or tabular datasets on classification and regression predictive modelling problems.

We used a simple XGBoost model with the default parameters after preprocessing the data and converting to vector representation.We got an accuracy of 78% .

**Confusion Matrix:**

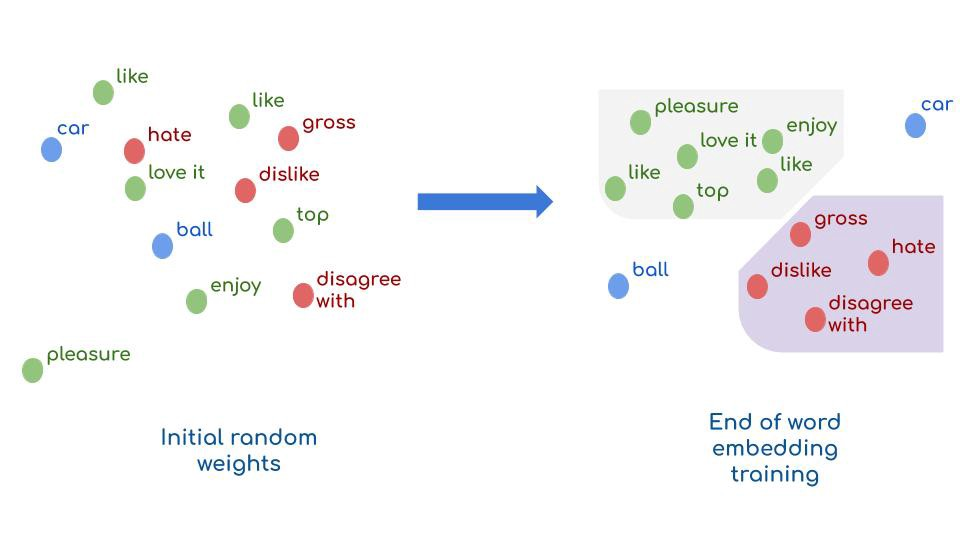


**Classification Report:**



**Pretrained Word Embeddings**

An embedding is a dense vector that represents a word (or a symbol). By default, the embedding vectors are randomly initialized, then will gradually be improved during the training phase, with the gradient descent algorithm at each back-propagation step, so that similar words or words in the same lexical field or with common stem will end up close in terms of distance in the new vector space



Pre-trained word embedding is an example of Transfer Learning. The main idea behind it is to use public embeddings that are already trained on large datasets. Specifically, instead of initializing neural network weights randomly, weights are set to these pre trained embeddings as initialization weights. This trick helps to accelerate training and boost the performance of NLP models.

In this project three pre trained word embeddings are explored.

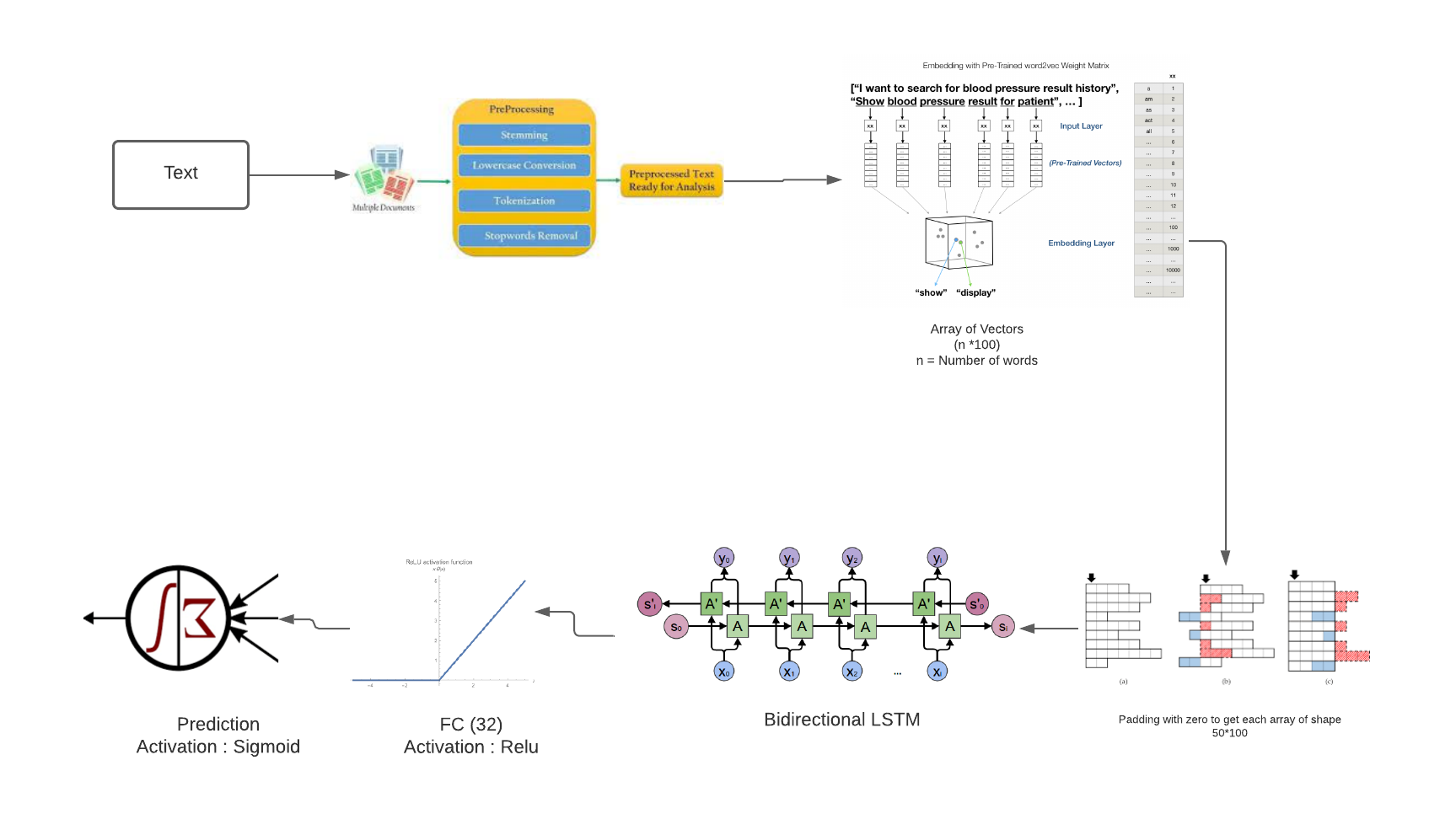
## **Glove pre trained word embeddings (Twitter data)**

These word vectors are trained on 2B Twitter tweets and has 1.2M vocabulary in various dimensions(25d,50d,100d,300d).Pretrained 100d word embeddings are used in the model.

**Methodology :**

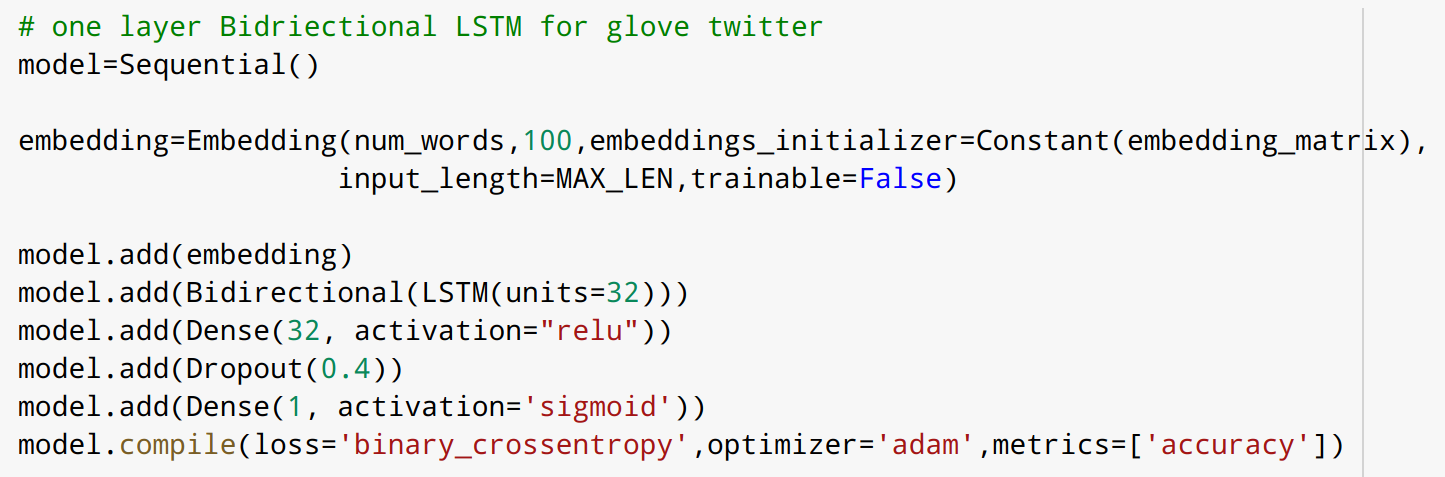
* Converting each word to a vector representation using pre-trained GloVe embeddings on Twitter data
* Load Glove model which has word embedding, each embedding of vector 100 dimensions
* Get embedding of those words which are in the corpus and construct a embedding matrix
* Assign the matrix as weight to first layer
* Train a network with LSTM cells which is followed by a set of fully connected layer with ReLu activation function
* Dense layer with sigmoid activation function is added as an output layer in order to classify a given tweet.

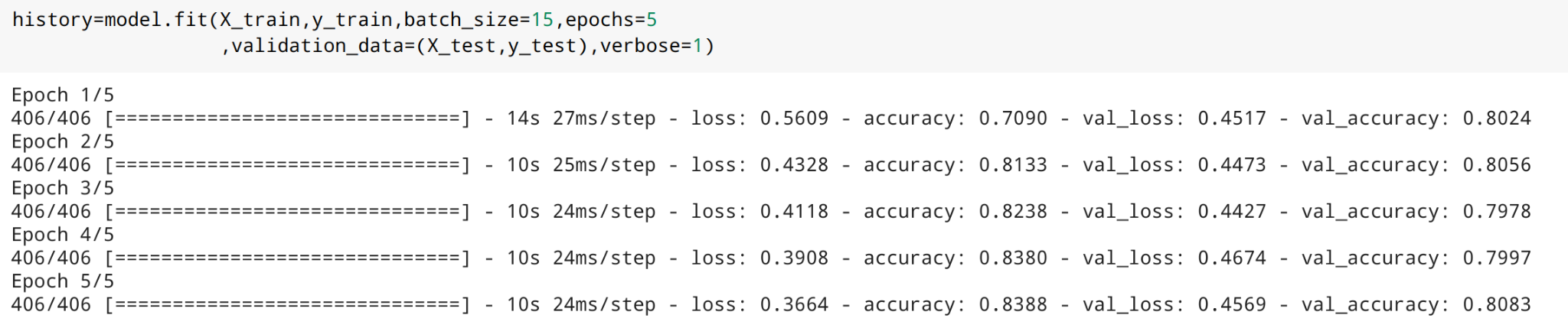
**Model Architecture:**



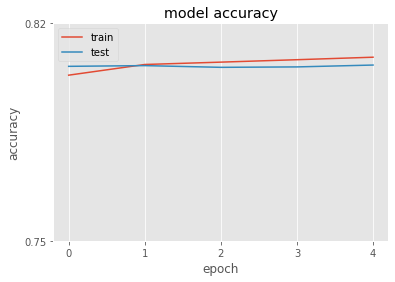
**Model Summary:**

Training Data is split into training and validation datasets and accuracy of below in each epoch is as follows





Accuracy and Loss plots of the model are as follows



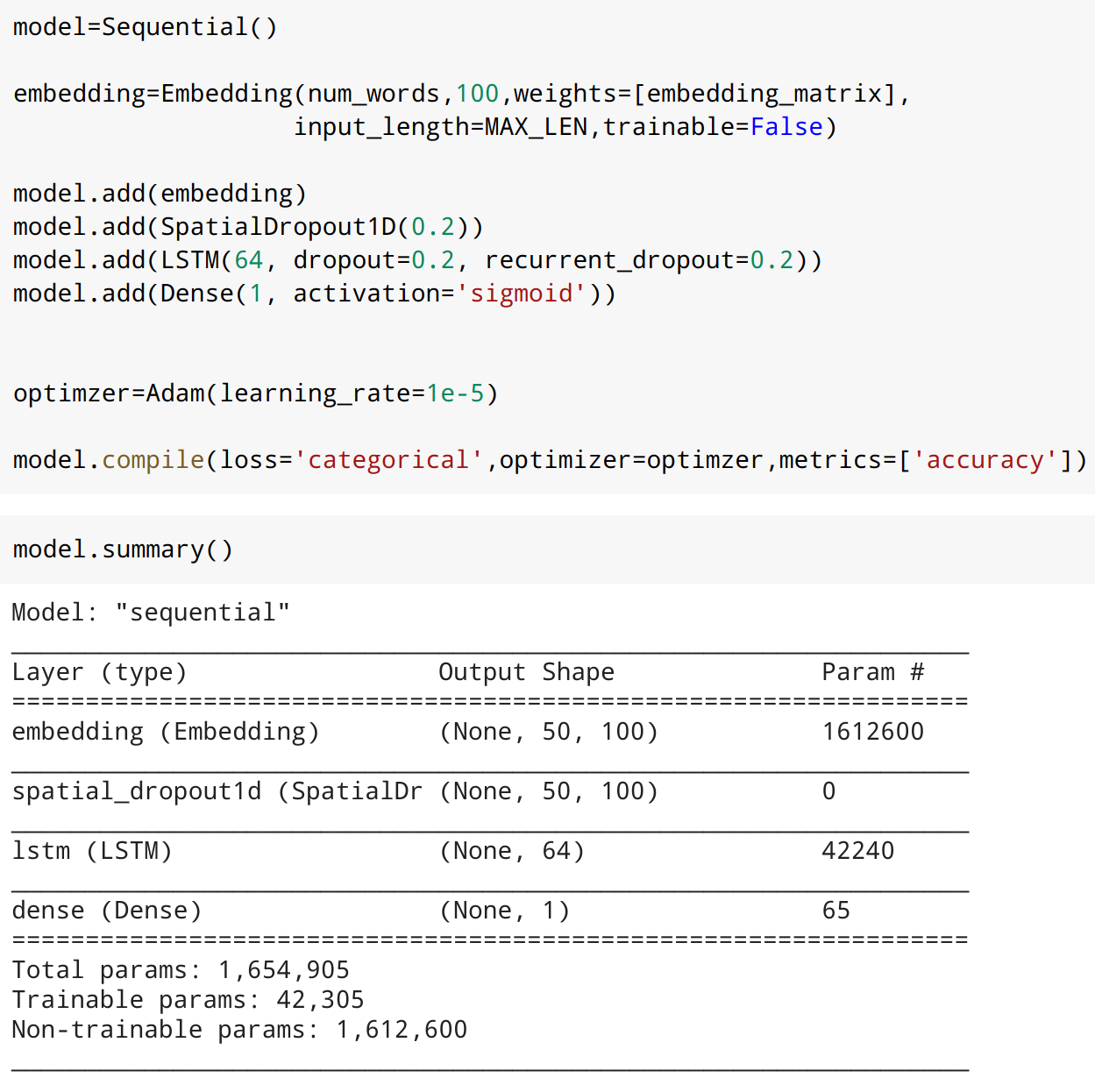


After experimenting with different batch\_sizes and number of epochs,it was observed that by increasing the number of epochs and batch\_size, the model was overfitting the training data which decreased the validation accuracy.

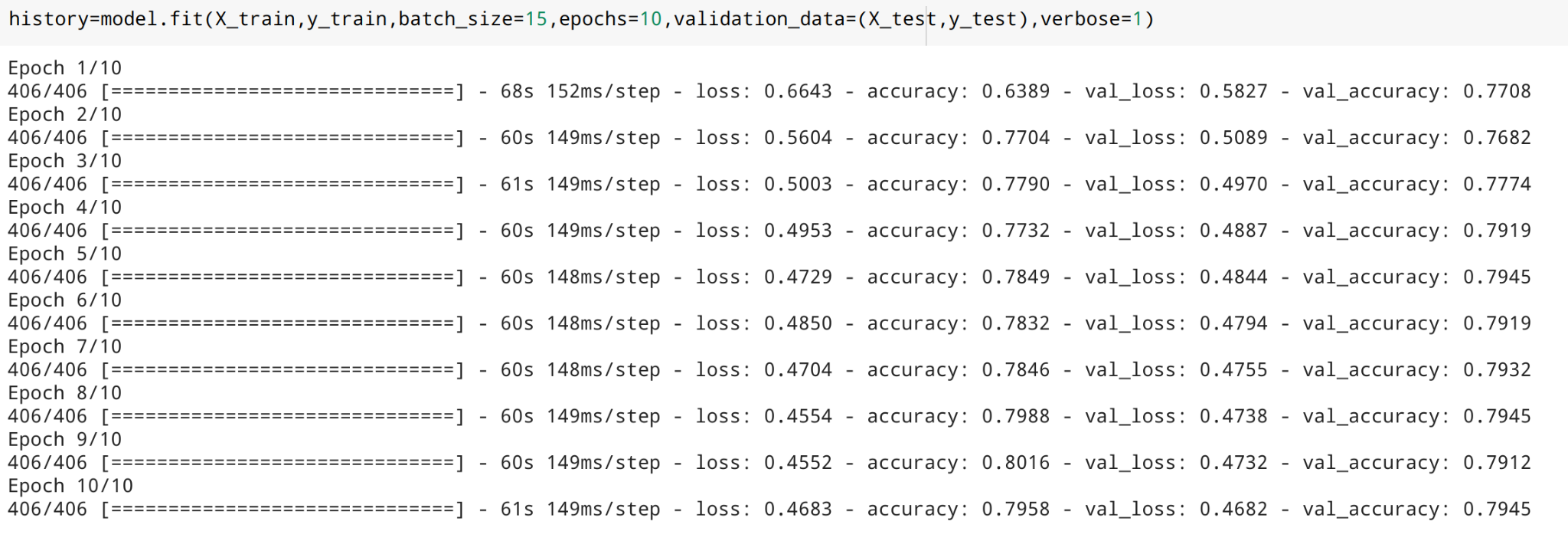
## **Glove pre trained word embeddings (Wiki data)**

These word vectors are trained on 6Billion tokens and has 400K vocabulary in various dimensions(25d,50d,100d,300d).Pretrained 100d word embeddings are used in the model.

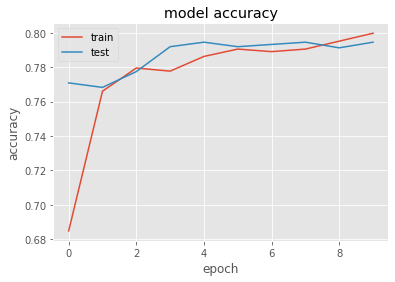
Below is the summary of the model used

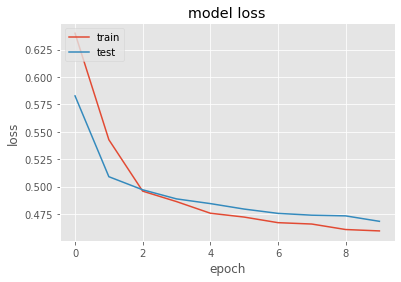


Below is the training and validation accuracy over each epoch



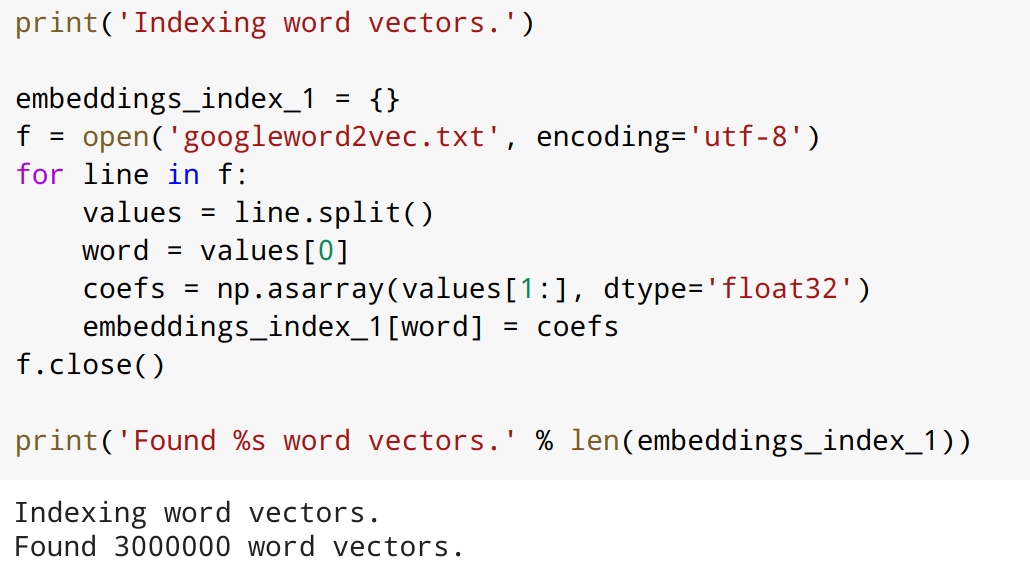
Accuracy and loss plots of the model is as follows



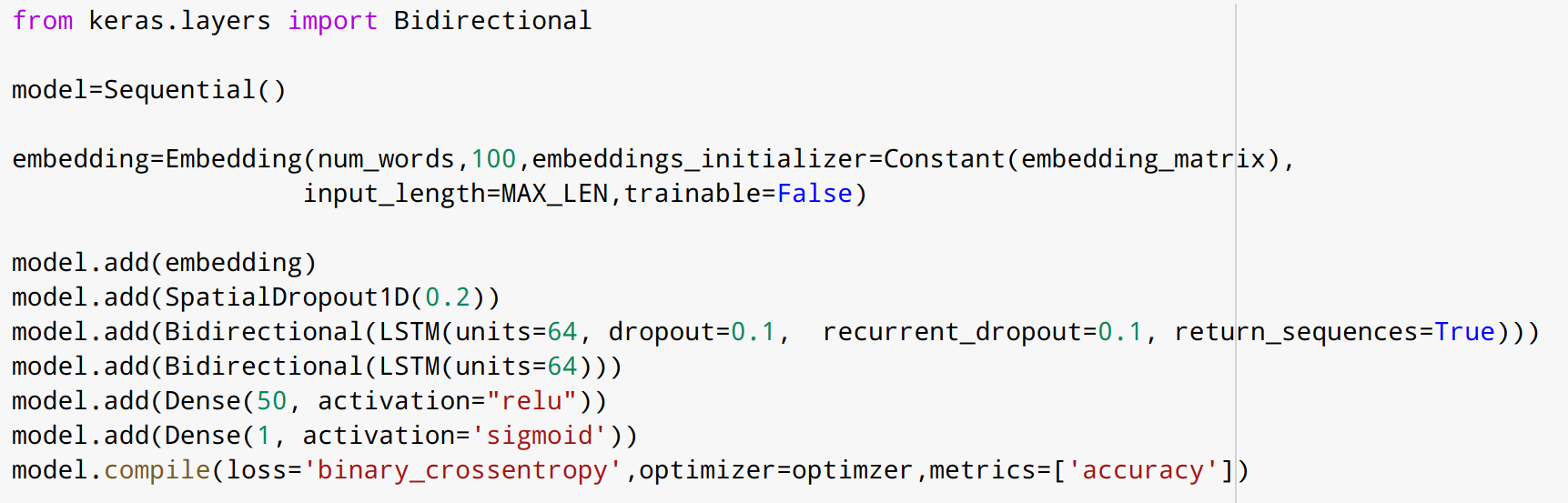


## **Google pre trained Word2Vec model**

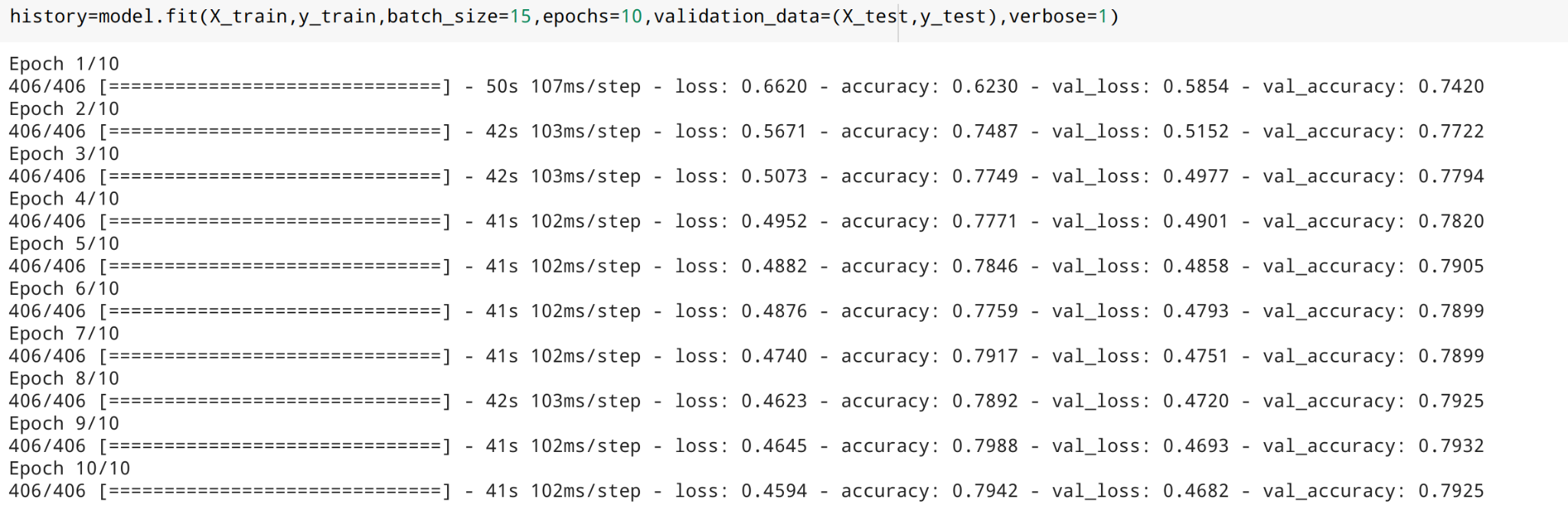
It includes word vectors for a vocabulary of 3 million words and phrases that were trained on roughly 100 billion words from a Google News dataset. The vector length is 300 features.Below is the summary of vocab size of the pretrained model



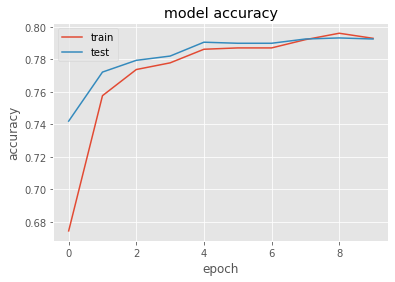
Model is a 2 layer Bidirectional LSTM followed by an output layer. Below is the screenshot of the model used.

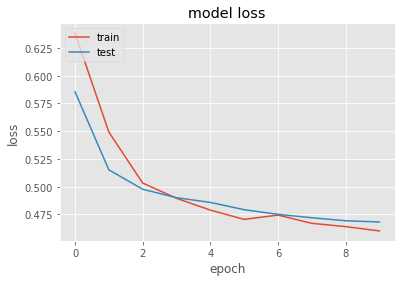


Training and validation accuracy of the model over each epoch is as follows



Accuracy and Loss plots of the model are as follows





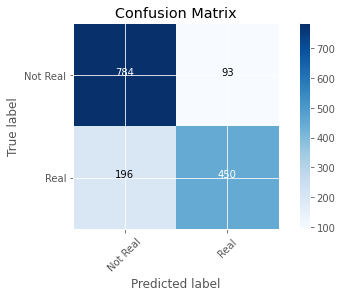
**Observations:**

Accuracy of all the 3 trained models are very close to each other.

|  |  |
| --- | --- |
| Glove(Twitter data) + LSTM | 80.83% |
| Glove(Wiki data) + LSTM | 79.45% |
| Google(Word2vec) + LSTM | 79.25% |

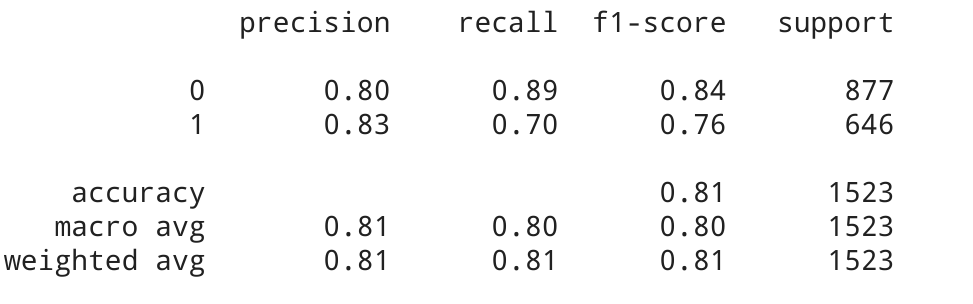
However, the model with pre-trained word embeddings trained on Twitter data gave slightly better accuracy of 80.83%.Hence this model is chosen as the better model among all the three trained models.

**Confusion matrix of the Glove(Twitter) model:**



It can be observed that 196 real disaster tweets were classified as not real disaster tweets.

**Classification Report of the Glove(Twitter) model:**



## **DistilBERT Model**

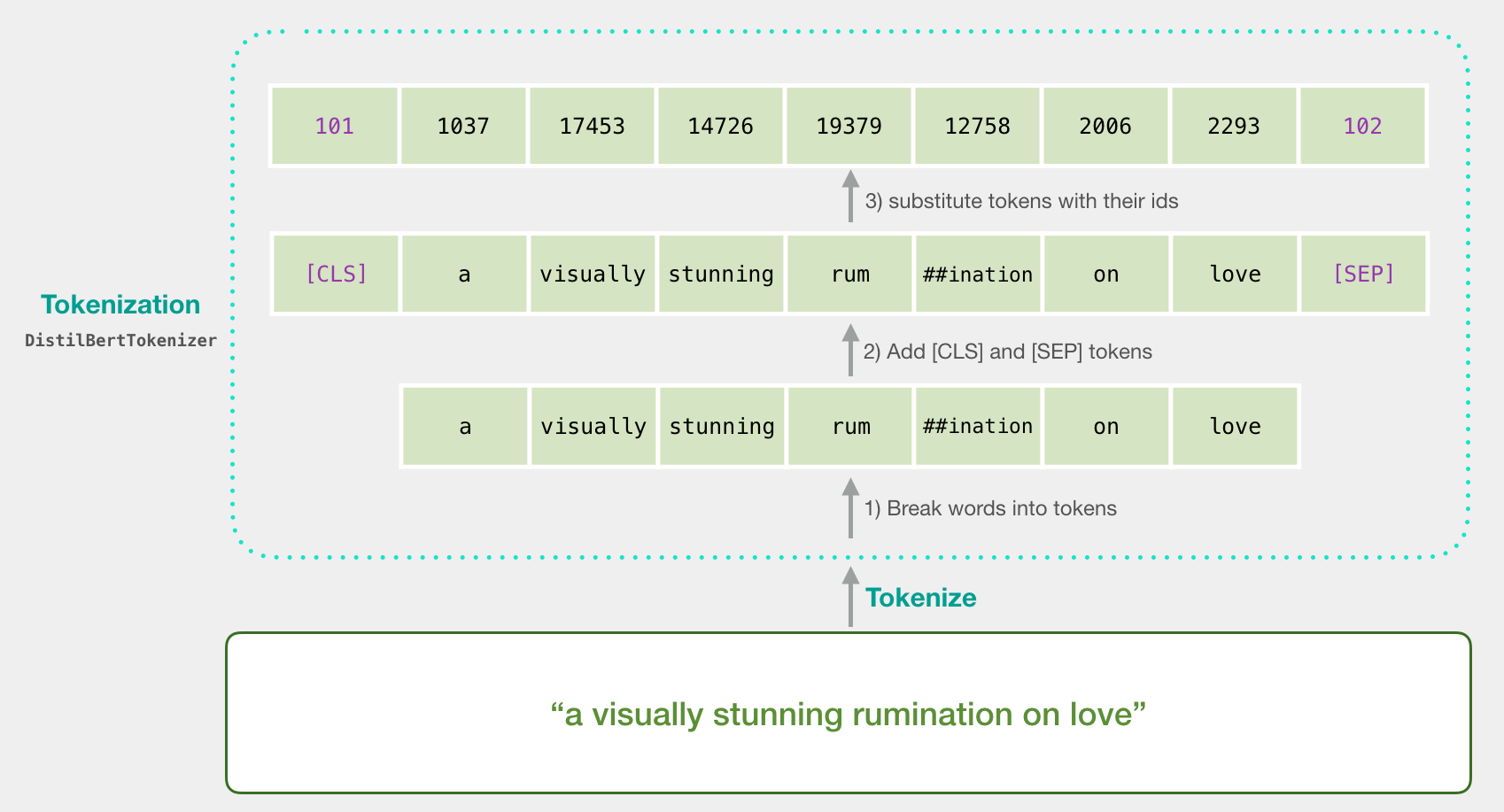
DistilBERT is a small, fast, cheap and light Transformer model trained by distilling BERT base and open sourced by the team at HuggingFace.It has 40% less parameters than bert-base-uncased,runs 60% faster while preserving over 95% of BERT's performances.DistilBERT model is made up of two models

* DistilBERT processes the sentence and passes along some information it extracted from it onto the next model.
* The next model, a basic Logistic Regression model from scikit learn will take in the result of DistilBERT’s processing, and classify the tweet as either real or not real (1 or 0, respectively)

**Preparing Dataset**

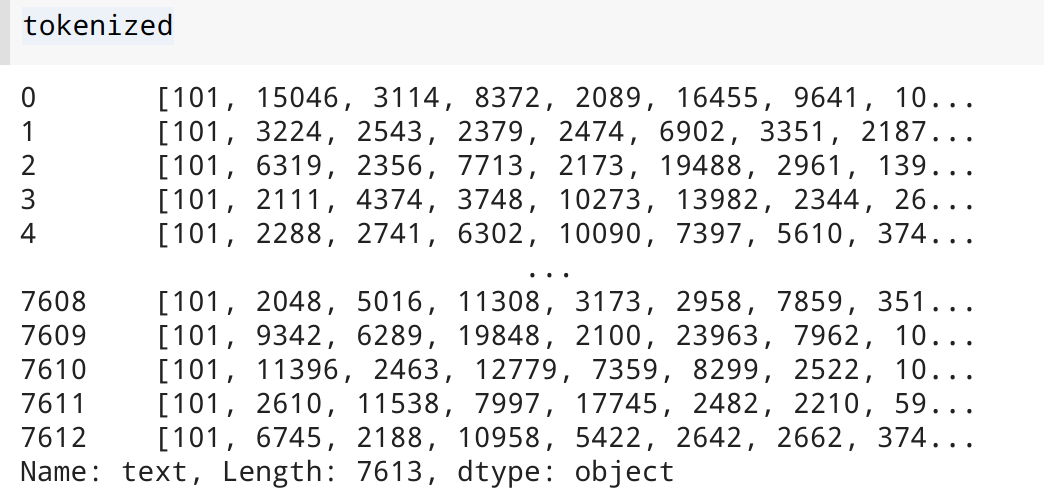
**Step 1 Tokenization**

To tokenize the sentences - break them up into word and subwords in the format BERT is comfortable with.



Tokenizer adds tokens [CLS] and [SEP] at starting and ending of each sentence and maps the words into ids using DistilBert pretrained tokenizer and pretrained weights.

Tokenized data of tweets is as follows

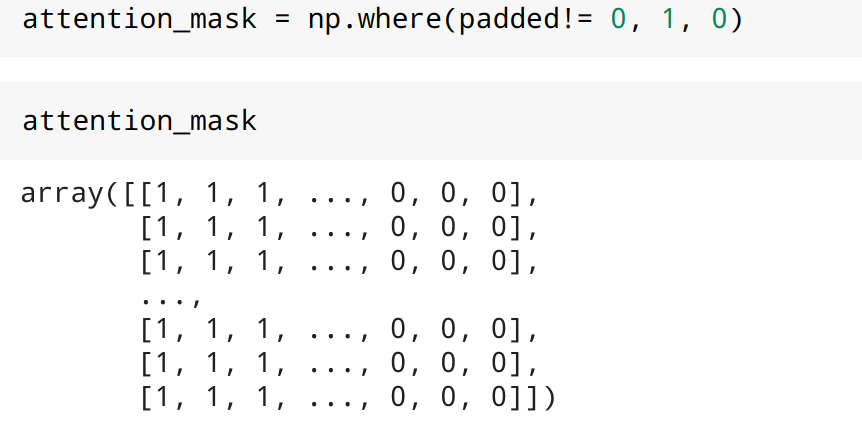


**Step2 Padding**

All lists of input sentences are padded to the same size, so we can represent the input as one 2-d array, rather than a list of lists (of different lengths).

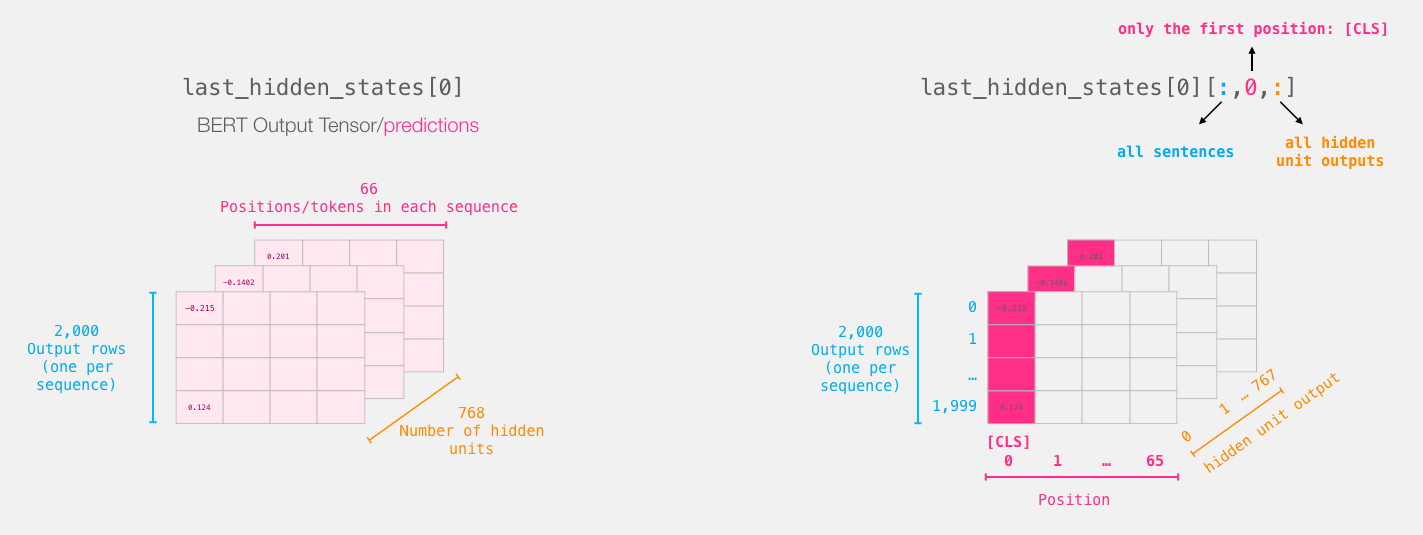
**Step 3 Masking**

Create another variable attention\_mask to tell BERT to ignore (mask) the padding added when it's processing its input

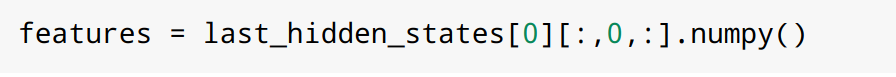


**Training the first model of DistilBERT:**

The model() function runs processed sentences through BERT. The results of the processing will be returned into last\_hidden\_states.

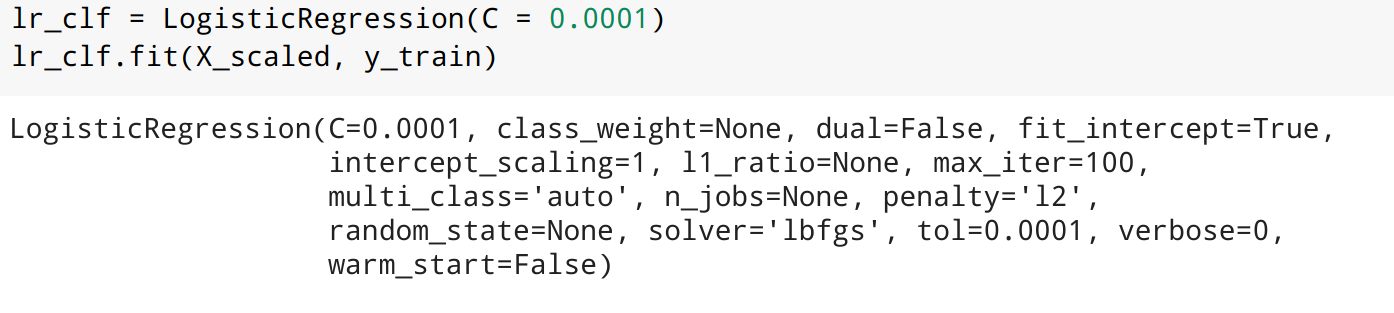


Last\_hidden\_states serve as features to logistic regression model in the next step



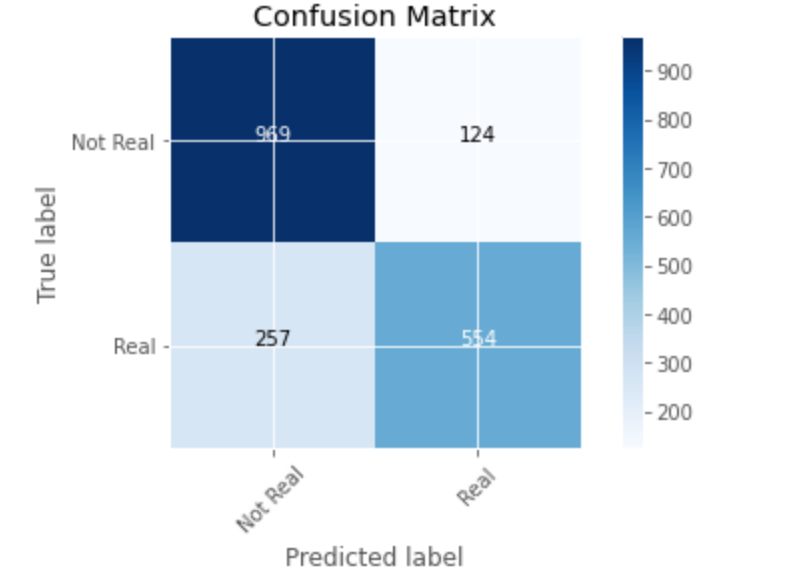
**Training the second model of DistilBERT**

After splitting the dataset into training and validation, the model is trained using Logistic Regression.

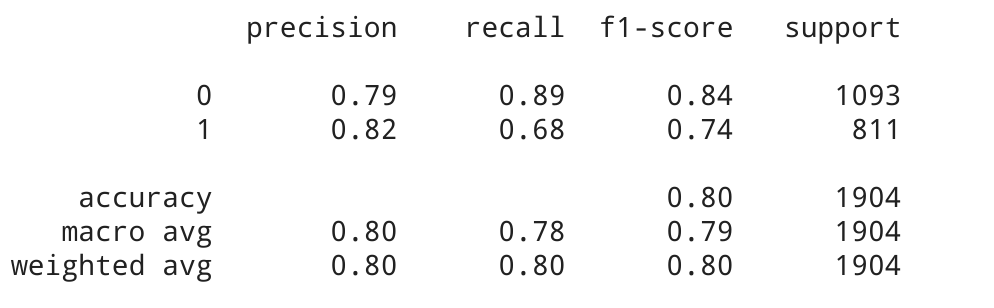


The model gave an accuracy of 79.9%. Below are the confusion matrix and classification report of the model.

**Confusion Matrix**



**Classification Report**



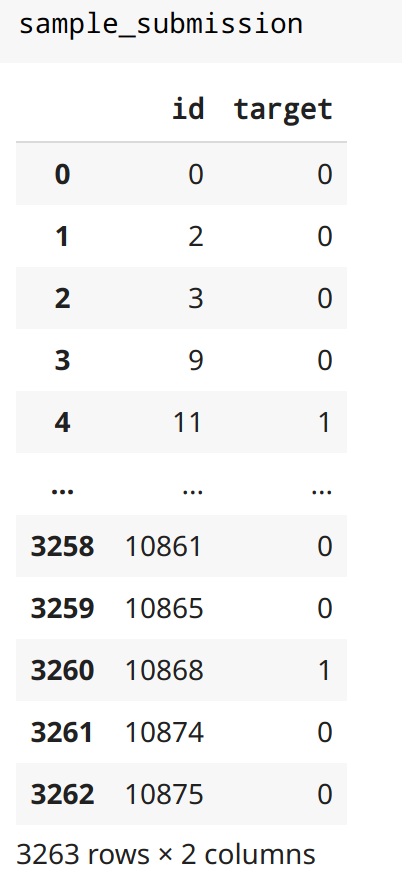
# 

# 

# 

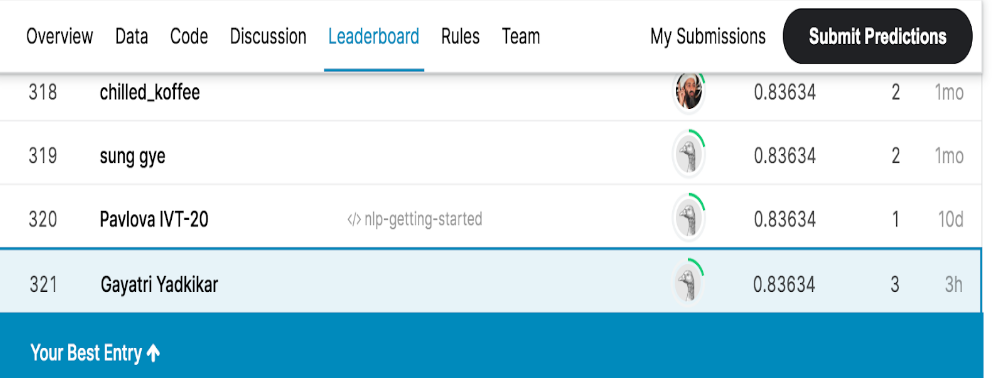
# **Sample Predictions**

Below are the sample predictions submitted to Kaggle challenge. Value ‘0’ indicates that tweet is predicted as not a real disaster and value ‘1’ indicates that tweet is predicted as a real disaster.



# **Summary of the Models**

|  |  |
| --- | --- |
| **Model** | **Accuracy** |
| Logistic Regression | 80.05% |
| SVM | 83.72%  (Kaggle Rank - 321 out of 2359 teams) |
| Random Forest | 73.6% |
| XG Boosting | 78.4% |
| Glove(Twitter data) + Bidirectional LSTM | 80.83% |
| Glove(Wiki data) + Bidirectional LSTM | 79.45% |
| Google(Word2vec) + LSTM | 79.25% |
| DistilBERT | 79.9% |



# **References**

[1] Y. Zhang and B. Wallace, “A Sensitivity Analysis of (and Practitioners’ Guide to) Convolutional Neural Networks for Sentence Classification.” 2017

[2] A. Dai and Q. Le, “Semi-supervised Sequence Learning.” 2015

[3] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. [GloVe: Global Vectors for Word Representation](https://nlp.stanford.edu/pubs/glove.pdf).

[4] <https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html>

[5]<https://towardsdatascience.com/pre-trained-word-embedding-for-text-classification-end2end-approach-5fbf5cd8aead>

[6]<https://github.com/jalammar/jalammar.github.io/blob/master/notebooks/bert/A_Visual_Notebook_to_Using_BERT_for_the_First_Time.ipynb>

[7]<https://arrow.tudublin.ie/cgi/viewcontent.cgi?article=1182&context=scschcomdis>

[8]<https://www.aclweb.org/anthology/W16-6201.pdf>

[9]<https://arxiv.org/pdf/1803.11175.pdf>

[10]<https://towardsdatascience.com/use-cases-of-googles-universal-sentence-encoder-in-production-dd5aaab4fc15>

[11]<https://tfhub.dev/google/universal-sentence-encoder-multilingual-large/3>