Twitter Sentiment Classification

A project on Multi-class sentiment Classification on Covid-19 tweets dataset

Introduction

Corona virus was declared a pandemic by World Health Organization on 11th March 2020. While the whole world was halted, social media sites were the most active places that allowed people to express, seek help, stay updated and reciprocate.

This project, aimed to analyze various types of "Tweets" gathered during pandemic times, will be helpful for different stakeholders. For example,

- Government can make use of this information in policymaking as they would be able to know how people are reacting to this new strain and what all challenges they are facing.
- Various profit organizations can make a profit by analyzing various sentiments as one of the tweets telling us about the scarcity of masks and toilet papers. These organizations can able to start the production of essential items thereby can make profits.
- Various NGOs can decide their strategy of how to rehabilitate people by using pertinent facts and information.

Sentiment analysis, also known as “opinion mining” or “emotion Artificial Intelligence” alludes to the utilization of

- Natural Language Processing (NLP)
- Text Mining
- Computational Linguistics and
- Bio Measurements

to methodically recognize, extricate, evaluate, and examine emotional states and subjective information.

Sentiment analysis is generally concerned with the voice in client materials; for example, surveys and reviews on the Web and web-based social networks.

Problems in Sentiment Analysis

Despite the availability of software to extract data regarding a person's sentiment on a specific product or service,organizations and other data workers still face issues regarding the data extraction. Sentiment Analysis of Web Based Applications Focus on Single Tweet Only.

With the rapid growth of the World Wide Web, people are using social media such as Twitter which generates big volumes of opinion texts in the form of tweets which is available for the sentiment analysis.. This translates to a huge volume of information from a human viewpoint which make it difficult to extract a sentences, read them, analyze tweet by tweet, summarize them and organize them into an understandable format in a timely manner. Added to this, there is difficulty of Sentiment Analysis with inappropriate English.

Along with this, there are issues in extracting due to http links, numbers, word contractions, tags to people and punctuations, which are not consequential for sentiment analysis.

Survey

- **Natural-language processing** (**NLP**) is an area of computer science and artificial intelligence concerned with the interactions between computers and human (natural) languages.In this, the vectors x are derived from textual data to reflect linguistic properties.
    - **BAG OF WORDS:** It creates a **vocabulary** of all the unique **words** occurring in all the documents in the training set
    - **TF-IDF:** short for **term frequency-inverse document frequency**, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.
    - **WORD2VEC:** used for learning vector representations of words called “word embeddings”. It takes the semantic meaning of words.

Links for reference

Sentiment Analysis using SVM

TF-IDF Vectorizer scikit-learn. Deep understanding TfidfVectorizer by… | by Mukesh Chaudhary | Medium

A Guide to Text Classification and Sentiment Analysis

Sentiment Analysis: Predicting Sentiment Of COVID-19 Tweets

Project Details

Project Details

Following are the specifications of our project:

- To extract the meaningful parts from the Covid-19 tweets dataset and convert it into usable form for sentiment analysis.
- Following are the sentiments as our class labels:
  - Extremely negative
  - Negative
  - Neutral
  - Positive
  - Extremely positive

Project Details(Contd.)

- To train the following models with the train dataset:
  - Multi-class Naive Bayes Classifier
  - Stochastic Gradient Descent Multi-class Classifier
  - State Vector Machine Multi-class Classifier
  - Multi-class Logistic Regression
  - Bidirectional Long Short Term Memory(LSTM) Model
- To find the accuracy resulting with these models and compare them
- To test the accuracy of the model trained using the training dataset on the testing dataset, hence successfully concluding the sentiment classification of tweets

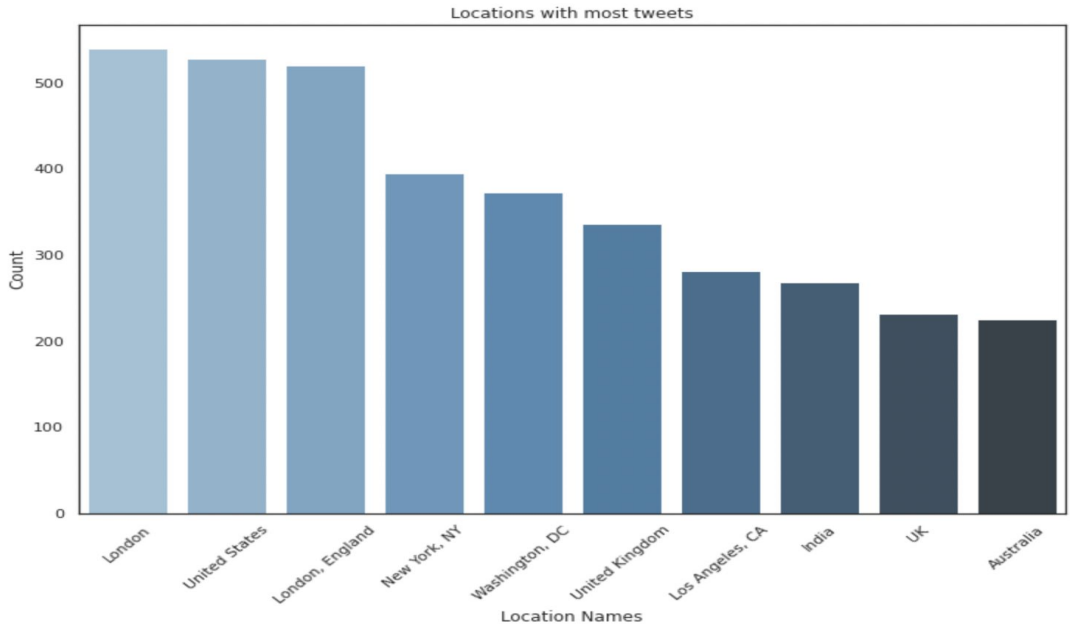
Data Cleaning

Extracting the following and removing them:

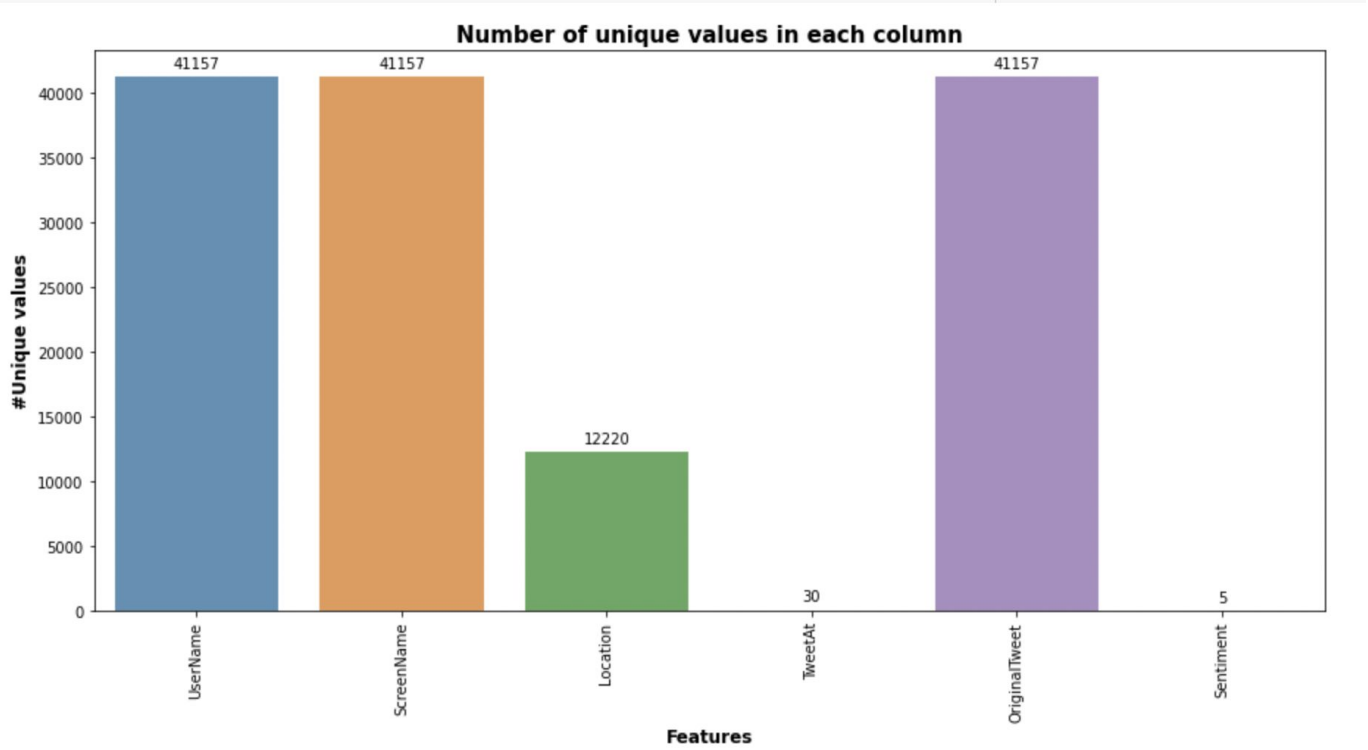
- Hashtags
- Handles
- Links to websites
- HTML tags
- Numbers and symbols
- Stemming
- Lemmatization

Data Analysis

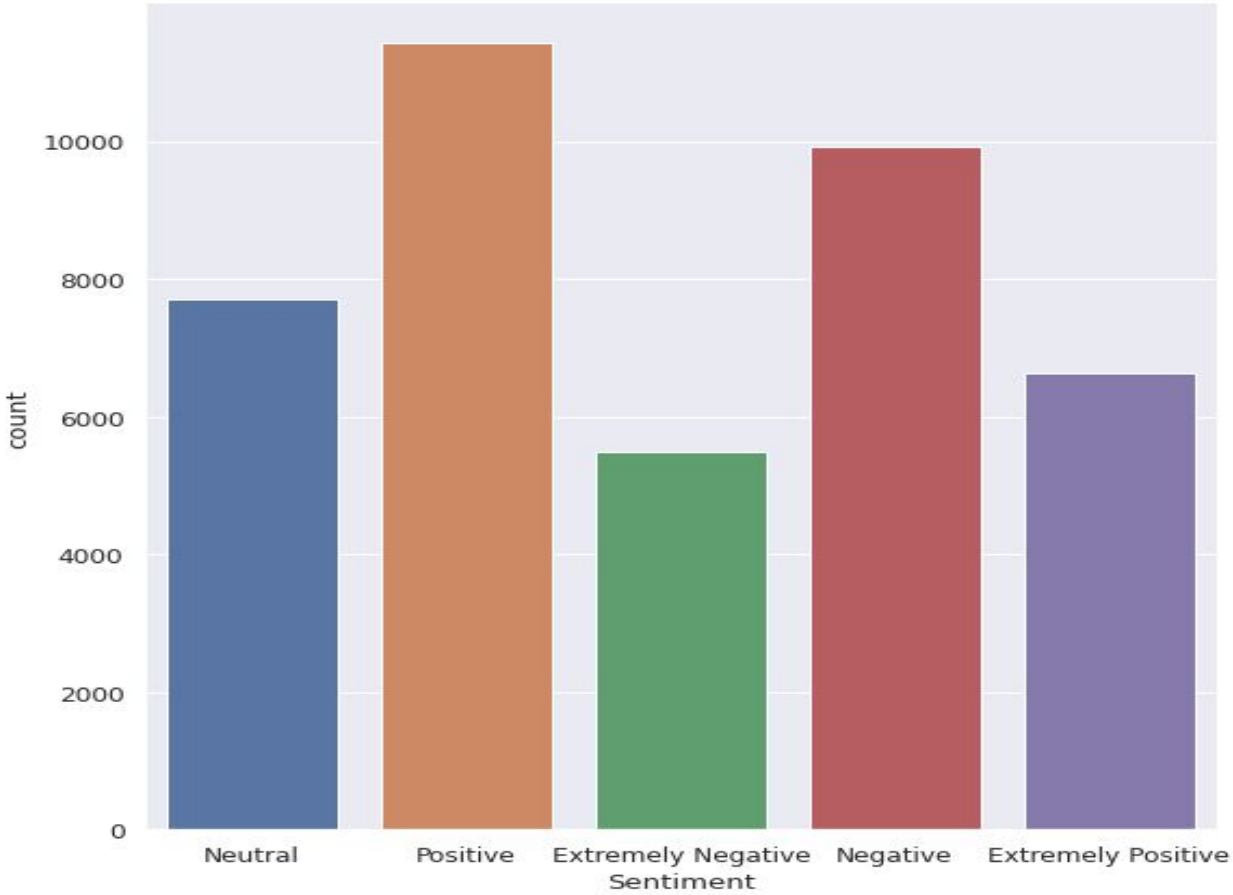
Location with most tweets



Number of unique values in each column



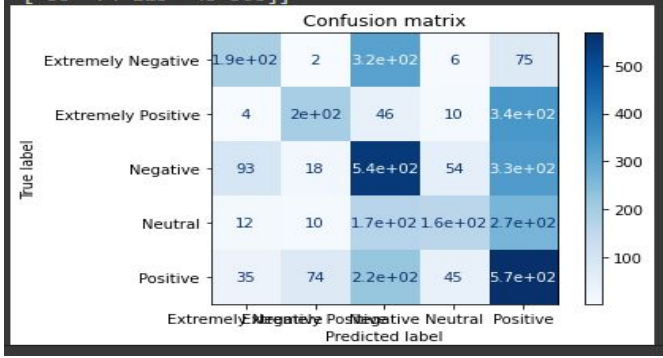
Classes

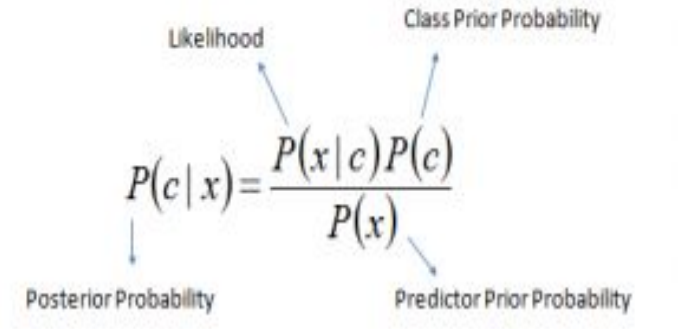

10000
8000
6000
4000
2000
0
count
Neutral
Positive
Extremely Negative
Negative
Extremely Positive
Sentiment

Mathematical Aspects & Implementation Result

Multi-class Naive Bayes Classifier.

```
training accuracy Score    :  0.6821925796340841
Testing accuracy Score :  0.4389152185360716
                    precision    recall  f1-score   support

Extremely Negative       0.33      0.57      0.42       338
Extremely Positive       0.33      0.66      0.44       302
          Negative       0.52      0.42      0.47      1300
           Neutral       0.26      0.58      0.36       277
          Positive       0.60      0.36      0.45      1581

          accuracy                           0.44      3798
         macro avg       0.41      0.52      0.43      3798
      weighted avg       0.50      0.44      0.44      3798

Confusion matrix
[[194   2 315   6  75]
 [  4 198  46  10 341]
 [ 93  18 545  54 331]
 [ 12  10 169 162 266]
 [ 35  74 225  45 568]]
```

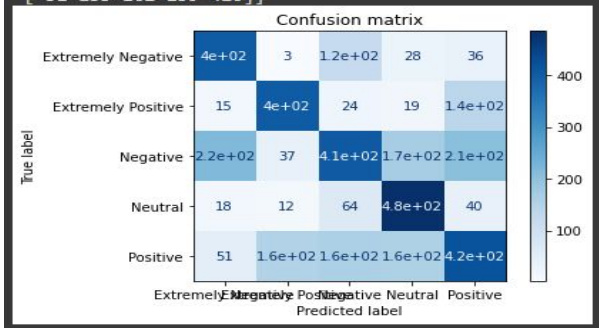




$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Stochastic Gradient Descent Multi-class Classifier

```
Training accuracy Score    :  0.8041645406613699
Testing accuracy Score :  0.557925223802001
                    precision    recall  f1-score   support

Extremely Negative       0.68      0.57      0.62       706
Extremely Positive       0.67      0.66      0.67       611
          Negative       0.39      0.52      0.45       779
           Neutral       0.78      0.56      0.65       863
          Positive       0.44      0.50      0.47       839

          accuracy                           0.56      3798
         macro avg       0.59      0.56      0.57      3798
      weighted avg       0.59      0.56      0.57      3798

Confusion matrix
[[403   3 122  28  36]
 [ 15 404  24  19 137]
 [219  37 407 172 206]
 [ 18  12  64 485  40]
 [ 51 155 162 159 420]]
```

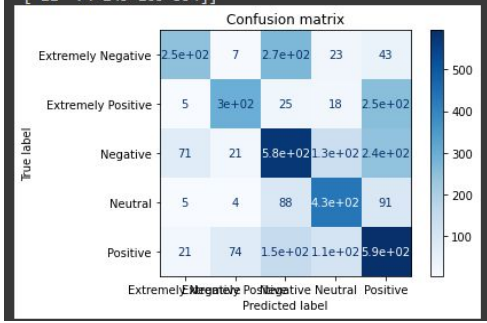


SGD is a simple yet very efficient approach to fitting linear classifiers and regressors under convex loss functions such as (linear) Support Vector Machines and Logistic Regression. **SGDClassifier** is a linear classifier (by default in `sklearn` it is a linear SVM) that uses SGD for training.With SGD we can:

- Initialise random weights and biases for our model
- Calculate the loss upon the class output by the model — it will be a lower value for an accurate prediction and a larger value for a wrong prediction
- Adjust the parameters in return by a certain amount — called the learning rate — so that the model begins to learn
- …and repeat the process from step 2 until it's time to stop.

State Vector Machine Multi-class Classifier

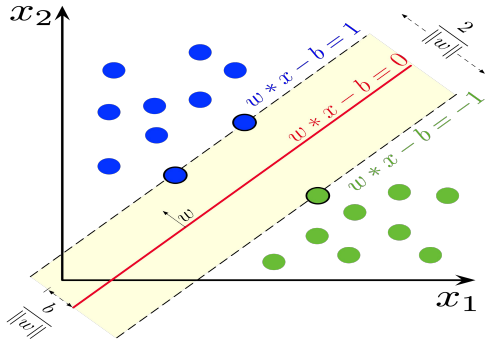
```
Training accuracy Score    :  0.8984862842286853
Testing accuracy Score :  0.564507635597683
                    precision    recall  f1-score   support

Extremely Negative       0.42      0.71      0.52       348
Extremely Positive       0.50      0.74      0.59       403
          Negative       0.55      0.52      0.54      1111
           Neutral       0.70      0.61      0.65       712
          Positive       0.63      0.49      0.55      1224

          accuracy                           0.56      3798
         macro avg       0.56      0.61      0.57      3798
      weighted avg       0.59      0.56      0.57      3798

Confusion matrix
[[246   7 273  23  43]
 [  5 297  25  18 254]
 [ 71  21 576 131 242]
 [  5   4  88 431  91]
 [ 21  74 149 109 594]]
```



The number of classifiers necessary for *one-vs-one multiclass classification* can be retrieved with the following formula (with n being the number of classes):

$$\frac{n * (n-1)}{2}$$

For Our Project n = 5.



$$\min_{\omega,b,\zeta} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \zeta_i$$

$$y_i(w^T \phi(x_i) + b) \geq 1 - \zeta_i$$

where ζ_i denotes the distance to the correct margin with $\zeta_i \geq 0,\ i = 1, \ldots, n$

where C denotes a regularization parameter

were $w^T w = \|w^2\|$ denotes the normal vector

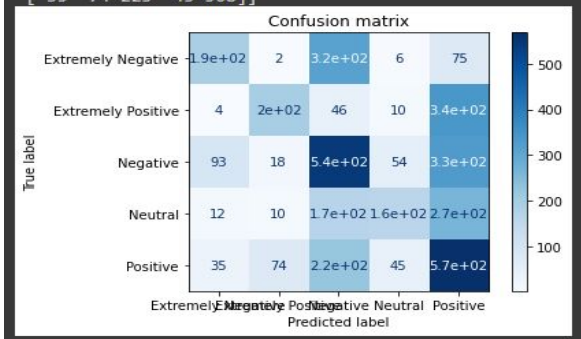
where $\phi(x_i)$ denotes the transformed input space vector

where b denotes a bias parameter

where y_i denotes the i-th target value

Multi-class Logistic Regression

```
Training accuracy Score    :  0.8714434968535122
Testing accuracy Score :  0.610057925223802
                   precision    recall  f1-score   support

Extremely Negative       0.56      0.67      0.61       502
Extremely Positive       0.63      0.72      0.67       529
          Negative       0.56      0.56      0.56      1039
           Neutral       0.71      0.64      0.67       692
          Positive       0.61      0.56      0.58      1036

          accuracy                           0.61      3798
         macro avg       0.62      0.63      0.62      3798
      weighted avg       0.61      0.61      0.61      3798

Confusion matrix
[[194   2 315   6  75]
 [  4 198  46  10 341]
 [ 93  18 545  54 331]
 [ 12  10 169 162 266]
 [ 35  74 225  45 568]]
```



Associate a set of weights with each class, then use a normalized exponential output

$$p(C_k|\mathbf{x}) = y_k(\mathbf{x}) = \frac{\exp(z_k)}{\sum_j \exp(z_j)}$$

where the activations are given by

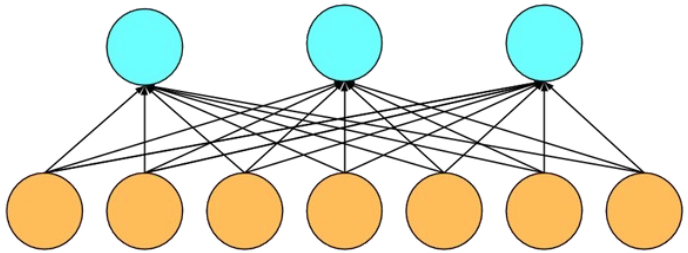
$$z_k = \mathbf{w}_k^T \mathbf{x}$$

The function $\frac{\exp(z_k)}{\sum_j \exp(z_j)}$ is called a softmax function

We will treat each class as a binary classification problem.
This is called the one-vs-all method.
In the one vs all method, when we work with a class, that class is denoted by 1 and the rest of the classes becomes 0.

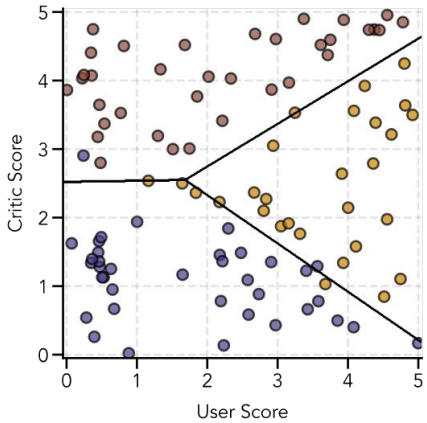
Multi-class Logistic Regression(Contd.)

Because now we have k outputs and not 1 we'll need weights connecting each of our inputs to each of our outputs. Graphically, the network looks something like this:



We can represent these weights one for each input node, output node pair in a matrix ***W***
We generate the linear mapping from inputs to outputs via a matrix-vector product ***xW+b***
The bias term is now a vector, with one component for each output node. The whole model, including the activation function can be written:

$$\hat{y} = \text{softmax}(\boldsymbol{x}W + \boldsymbol{b})$$

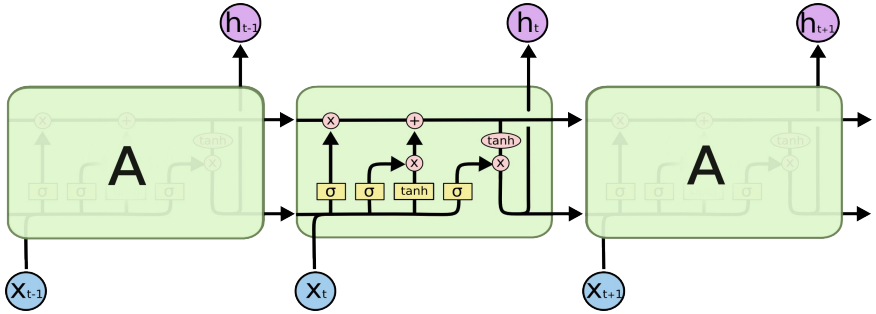


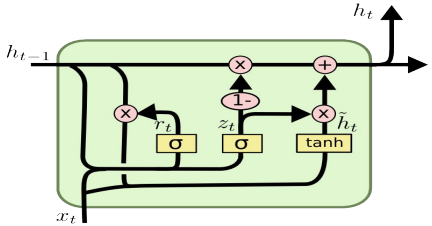
Assume we have ***d*** inputs and ***k*** outputs. Let's note the shapes of the various variables explicitly as follows:

$$\underset{1\times k}{\boldsymbol{z}} = \underset{1\times d}{\boldsymbol{x}}\ \underset{d\times k}{W} + \underset{1\times k}{\boldsymbol{b}}$$

Bidirectional Long Short Term Memory(LSTM) Model

In traditional neural networks, all the inputs, and outputs are independent of each other, but in cases like when it is required to predict the next word of a sentence, the previous words are required and hence there is a need to remember the previous words(as the next word will depend on your previous input).





$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Conclusion

The project used **tweet sentiment analysis** as a way of accessing written comments to decide whether articulation is positive, negative or neutral and to what degree. Various **machine learning** and **NLP approaches** were explored. It was observed that among machine learning models, logistic regression gave highest **accuracy of 61.006%**, whereas **Bi-directional LSTM** gave accuracy as high as **76.46%** .

Interesting area for future study includes the **fluctuations in the performance** of sentiment analysis algorithms in cases where **multiple features** are considered. In other words, combining various features was found to lead to **improve the performance** in most cases, but substandard performance in others.