# Twitter data Sentiment Analysis

## Project Report

Ram Sirusanagandla - rsirusan@uncc.edu

**Introduction**

Sentiment analysis is an application of natural language processing which is used understand and analyze the text data. This application provides the sentiment of the user's text which helps in understanding the author's emotion and attitude. In this project, I have scraped the data from the twitter.

The main objective of the project is to develop a model which analyses the sentiment of tweets that are related to a university. I have taken the tweets related to UMASS university. I scraped the data from the twitter by using the tweepy and few twitter API's based on the hashtags that are related to that particular university. While collecting the data, I must ensure to use more number of hashtags which helps us in getting more unique tweets.

While creating the model, this data undergoes several data pre-processing steps, which includes cleaning the raw data by removing emoji's, extra punctuations, symbols, hashtags and the unwanted numbers. This is the crucial and important step as the data is cleaned and all the irrelevant information is removed in this step which allows us to focus only on the text to analyse the sentiment.

In the next step we convert the preprocessed text into numerical representation which is used by the machine learning models. We've used the bag-of-words and TF-IDF approach. This method creates a matrix of word counts and provides the importance weights to the words. This method acts as an input for our model.

We made use of metrics like accuracy, precision, recall, and F1-score to evaluate how well our model performs. These metrics offer insightful information about the model's aptitude for accurately categorizing attitudes.

By performing sentimental analysis on the tweets related to the university, we can get helpful information like the opinion, emotion and attitude of the twitter user towards to the university. This information helps university to understand the mindsets of the people, track sentiments and make the decisions accordingly that helps the universities to enhance their reputation and also address the concerns.

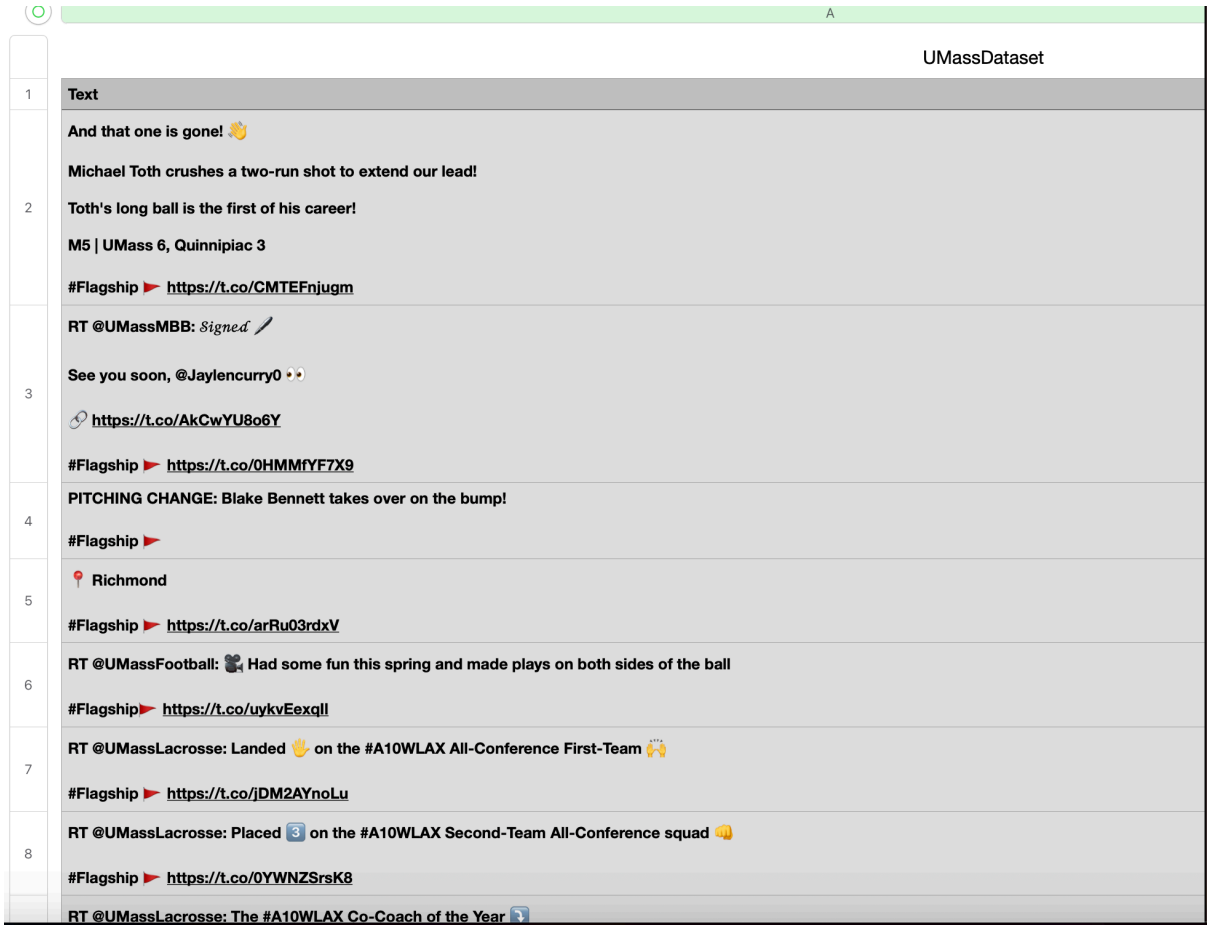
In the parts that follow, we'll go into detail about data collection methods, data preparation strategies, model architecture, evaluation measures, and the outcomes of our sentiment analysis model. With this study, we hope to show the effectiveness and potential of sentiment analysis in gathering valuable information from social media data.

**Data Collection:**

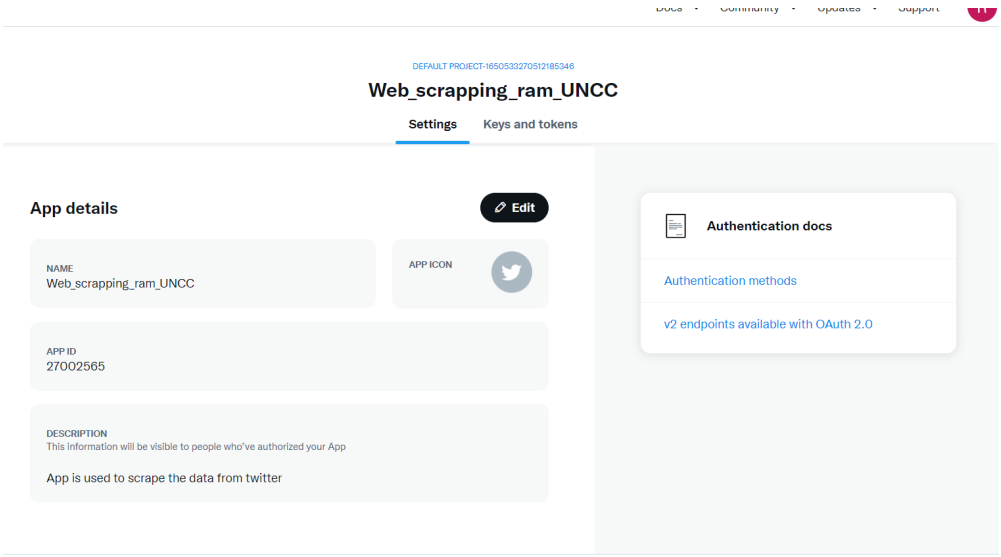
Data for this project is collected from Twitter. The Twitter has an API named Tweepy which helps in scrapping the data from twitter timeline.

This data collection involves many steps like account creation, authentication, searching, collecting, and storing. The first step is creating a Twitter account, which gives us access to view all the tweets in the application. Now, we need to create twitter developer account.

For this, we need to give our project name first/ In setting we can get all the API keys and access tokens required. The API can be viewed only once.



| | A |
|---|---|
| | UMassDataset |
| 1 | Text |
| 2 | And that one is gone! 👋<br>Michael Toth crushes a two-run shot to extend our lead!<br>Toth's long ball is the first of his career!<br>M5 \| UMass 6, Quinnipiac 3<br>#Flagship 🚩 https://t.co/CMTEFnjugm |
| 3 | RT @UMassMBB: *Signed* 🖋<br>See you soon, @Jaylencurry0 👀<br>🔗 https://t.co/AkCwYU8o6Y<br>#Flagship 🚩 https://t.co/0HMMfYF7X9 |
| 4 | PITCHING CHANGE: Blake Bennett takes over on the bump!<br>#Flagship 🚩 |
| 5 | 📍 Richmond<br>#Flagship 🚩 https://t.co/arRu03rdxV |
| 6 | RT @UMassFootball: 🎥 Had some fun this spring and made plays on both sides of the ball<br>#Flagship🚩 https://t.co/uykvEexqlI |
| 7 | RT @UMassLacrosse: Landed 🖐 on the #A10WLAX All-Conference First-Team 🙌<br>#Flagship 🚩 https://t.co/jDM2AYnoLu |
| 8 | RT @UMassLacrosse: Placed 3️⃣ on the #A10WLAX Second-Team All-Conference squad 👊<br>#Flagship 🚩 https://t.co/0YWNZSrsK8 |
| | RT @UMassLacrosse: The #A10WLAX Co-Coach of the Year 🔄 |




**Data Cleaning:**

The above is the account and the project in the developer account. For this, we will get 1500 tweets per month in elevated access. Using the keys and secrets nearly 2000+ tweets we scrapped from the twitter.

From the above image, we can say that data is not clean. For this using nltk libraries stop word and regular expressions, I have removed all the stop words and tags.

```
def data_processing(text):

    #converting all the data into lower case
    text = text.lower()

    #Removing tags
    text = re.sub(r"\@w+|\#",'',text)

    #Removing all the links
    text = re.sub(r"https\S+|www\S+|https\S+",'', text, flags = re.MULTILINE)   #it will sea

    #Removing all other other than characters.
    text = re.sub(r'{^\w\s}','',text)
    text = re.sub('[^a-zA-Z]',' ', text)

    #Tokenizing inorder to remove all the extra spacing
    text_tokens = word_tokenize(text)

    #Stop words are words which doesn't contribute in building meaning of the sentence.
    #Before vectorizing removing these will help in building vector with required words.
    stop_words = set(stopwords.words('english'))
    filtered_text = [w for w in text_tokens if not w in stop_words]

    #Joining all back
    return ' '.join(filtered_text)
```

Above is the code snippet for the data preprocessing. Here, I used 4 main regex.

In the first one I have removed the tags and hashtags and in the second regex I have remove the links, in the third and fourth I have removed all other expect characters.

**Model Architecture:**

Before that, I have used TFIDF vectorizer and Count vectorizer to get the word count and to find the importance of the word in that sentence and in that document. The count vectorizer input is given into TFIDF. This will form a sparse matrix.

Some terms used:

Activation function: Used to determine whether the neuron should be active or not.

ReLu: Mostly used for continuous data. It's function is max(0,x). It means it keeps only the positive values.

Sigmoid: It is mostly used for classification problems. It gives output value in between -1 and 1.

Count Vectorizer: CountVectorizer encodes each document by counting the instances of each word in the vocabulary once the vocabulary has been constructed. Each document is given a feature vector, with each value denoting the frequency of a word in the document. Since these words frequently have minimal semantic meaning, CountVectorizer offers an option to remove them from the lexicon.

TF-IDF (Term Frequency-Inverse Document Frequency) is a technique used for feature extraction in natural language processing tasks. It is commonly used to show the importance of a word in a document within a large collection of documents.

TF-IDF combines two metrics: term frequency (TF) and inverse document frequency (IDF). The term frequency (TF) and inverse document frequency (IDF) values are multiplied to provide the TF-IDF score for a word in a document.

Term Frequency (TF): It estimates each time a particular word appears in a document. It is measured as its ratio of the entire amount of words in a document to the number of times a word appears in that document.

Inverse Document Frequency (IDF): It analyses a word's occurrence over the full corpus of documents. The logarithm of the coefficient of the total number of documents to the number of documents containing the word is in use to calculate it.

Maxpooling: It is picks all the elements in the defined kernel size and keeps only the max value in all them.

LSTM: (Long Short Term Memory): The modeling of sequential data using this recurrent neural network (RNN) architecture is particularly common in natural language processing (NLP) tasks. The primary concept of LSTM is to use memory cells with long-term store and retrieval capability.

The primary applications of this deep learning architecture are visual recognition tasks like picture categorization, object identification, and image segmentation.

Convolutional Layers: Every layer applies several filters to the input and creates feature maps, each of which represents a different feature or pattern.

Pooling layers: In  order to make the network more computationally efficient and resistant to minute changes in the input, pooling layers are utilized to minimize the spatial dimensions of the feature maps.

CNNs have transformed the field of computer vision by producing cutting-edge outcomes on a variety of image-related tasks. CNNs have been utilized in NLP for applications like text categorization, sentiment analysis, and text production.

GRU (Gated Recurrent Unit): It is a recurrent neural network (RNN) architecture designed to model sequential data. It was introduced as an enhancement to the LSTM network, offering comparable capabilities with fewer parameters.

GRU networks, like other recurrent neural networks, are trained using backpropagation through time (BPTT). Natural language processing, speech recognition, machine translation, and time series analysis have all made extensive use of GRUs. They give an efficient method for modeling sequential data while resolving the issues of vanishing gradients and capturing long-term interdependence.

Most of the sentiments were done manually by me. But I have included the BERT model in the code to show how to find out the sentiments of the data. These sentiments acts as the labels for the model. In the dataset which I have only Text is the only feature which is needed to predict the sentiment and all other are not required. Likewise, the sentiment_value column which has all the sentiments Next step is dividing the data into training set and testing set using train_test_split. I have splitter data into 70-30.

Next, I have developed a deep learning model with dense layers in it.

The model is a dense model with the following neurons and activations in it.

- First layer is the embedding layer. It has input length as 38 and it has 64 neurons in it. Now this layer takes the input and gives the output to the next layer. Next layer is the LSTM layer.

```
]: model.summary()
```

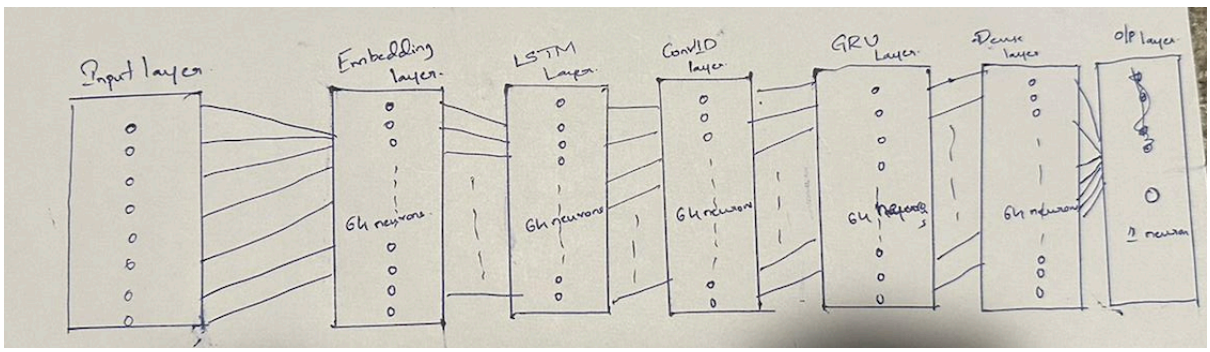
```
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_7 (Embedding)     (None, 38, 64)            331776

 lstm_8 (LSTM)               (None, 38, 64)            33024

 conv1d_7 (Conv1D)           (None, 32, 64)            28736

 max_pooling1d_7 (MaxPooling  (None, 5, 64)            0
 1D)

 dropout_20 (Dropout)        (None, 5, 64)             0

 gru_7 (GRU)                 (None, 64)                24960

 dropout_21 (Dropout)        (None, 64)                0

 dense_14 (Dense)            (None, 64)                4160

 dropout_22 (Dropout)        (None, 64)                0

 dense_15 (Dense)            (None, 1)                 65

=================================================================
Total params: 422,721
Trainable params: 422,721
Non-trainable params: 0
_________________________________________________________________
```

In [268]:
```
model = Sequential()
model.add(Embedding(size_of_words, 64, input_length = maxlen))
model.add(LSTM(64,dropout = 0.4, recurrent_dropout = 0.4, return_sequences=True))
model.add(Conv1D(64, 7, activation = 'relu'))
model.add(MaxPooling1D(6))
model.add(Dropout(0.5))
model.add(GRU(64, dropout=0.3))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu', kernel_regularizer=regularizers.l2(0.01)))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))
```

- This LSTM layer has 64 neurons in it. Dropout and recurrent drop out function is almost same. It means, for every run, it will block some neurons in it ignorer to stop the overfitting.
- This layer's output is given to the next layer which is Conv1D. This layer's has a kernel size of 7 and has 64 neuron's in it. For this, I have used ReLu as it's activation function. This layer's output is given to Maxpooling which has pool size as 6. Before giving output to the next layer, there is a dropout of 0.5 which does the same as explained above.
- This output is given to GRU with 64 neurons in it and it has dropout as 0.3.
- This output is given to dense layer with 64 neurons in it has activation function as relu and I have used L2 regularizer of 0.01 in order to avoid the over fitting. It has a dropout of 0.5.
- The final layer has only one neuron in it and as this is a classification problem, this model has to predict whether it is or not like this. So, I have used sigmoid for it as activation function.

Now, the model is ready, I have given the training set to fit this model and given 20 epochs with 64 as batch size.

The model will look something like this :



**Results and Score:**

The created model has an accuracy of above 80 percent for both test and train dataset I.e., model is able to predict most of the test data. As the data mostly has positive reviews, model is also little aligned more towards positive reviews and unable to predict for some negative reviews.

```
Accuracy: 0.7115384615384615
Precision: 0.5062869822485206
Recall: 0.7115384615384615
F1-score: 0.5671868526446623
```

The above are the scores for the model.

Accuracy means how the model is able to predict. So here it is able to predict for 71% of the data.

Precision: It is the score of the model which predicts correct positives in all positives predictions. The model is able to predict for 50% of the data.

Precision = TP/TP+FN

Recall: It is the score of the model which predicts correct positives in all actual positives. Model is able to do for 71 percent.

Recall = TP/ TP+FP

F1 score: It is the harmonic mean of both recall and precision. Model's score for this is 50%.

F1 = 2*P*R/P+R

**Limitations:**

The model which I have constructed is having very good accuracy.  It is able to predict for all positive reviews and for some negative reviews.

I have tried several hyper parameters values and achieved this accuracy. But this came be improved. Model is little over fitted. We can say this from the below graph.


This model is overfitted. The hyper parameters which I have used in designing the model aren't good enough and also the dropouts and regulations has to be improved as per to avoid this situation.

The data which I have used has mostly positive reviews. This is also one of the reason behind the model alignment towards the positive reviews.

**Improvements:**

- Model should be more complex and it should have more complex activation functions like ELU. This is one the change which can be made.
- I should have tested with more hyper parameters values and then selected the best one.
- While training the model, I should have used random samples for each epoch ignorer to avoid the overfitting.
- More data preprocessing should have done. So that the word meanings and shortcut words and all would have understood by the model.

**Conclusion:**

The designed model performing well for most of the data which is been scrapped using Tweepy API from Twitter. As the twitter data is free flowing data, it is very difficult to convert all the words and make it understandable by the model. By using nltk and regex and TFIDF the model performed well for most of the predictions. By performing all the necessary improvement suggested above, the model will be more promising and will have greater accuracy.