CSEE5590-0001/490-0003: Big Data Programming

Increment 3 Report

Project Tile: Spark ETL and Sentiment Analysis

Team Members:

- Neeraj Padarthi - 19
- Hiresh Jakkala Bhaskar- 11
- Hari Y – 29

Goals and Objectives:

Motivation:

In this data-driven world, handling data has become vital in the decision-making process in many industries such as Telecom, Banking, Financial and Health sector servicing industries. Managing the sheer volumes of data and getting insights from it would be the main factors. One of the amazing frameworks that can handle big data in real-time and perform different analysis, using Apache Spark.

Objectives:

Our Project's main idea is to do the ETL process using Spark Streaming and implementing the machine learning concepts on this real-time data. The source of our system is Twitter data and we would be using Streaming Content which is real-time processing of data, by using streaming API we would be collecting the data in a near real-time process for a set of defined keywords. Then we would be performing the transformations on the streaming set of RDD's and load the data into the Hive system which is similar to basic ETL process. Also, we would be performing the EDA on twitter data while capturing the context of the data. Our project would also highlight the Sentiment Analysis System where we populate real-time sentiments for the tweets. It also identifies the major keyword factors for a tweet to be categorized into positive or negative sentiment.

Significance:

For sentiment analysis we are using a existing ML tool(TextBlob) to predict the sentiment of the tweets and based on that we will analyse the keywords which are vital in the prediction of sentiment and we will train a new ML model with

that analysis to predict the sentiment of tweets which will increase the accuracy of the prediction.

Features:

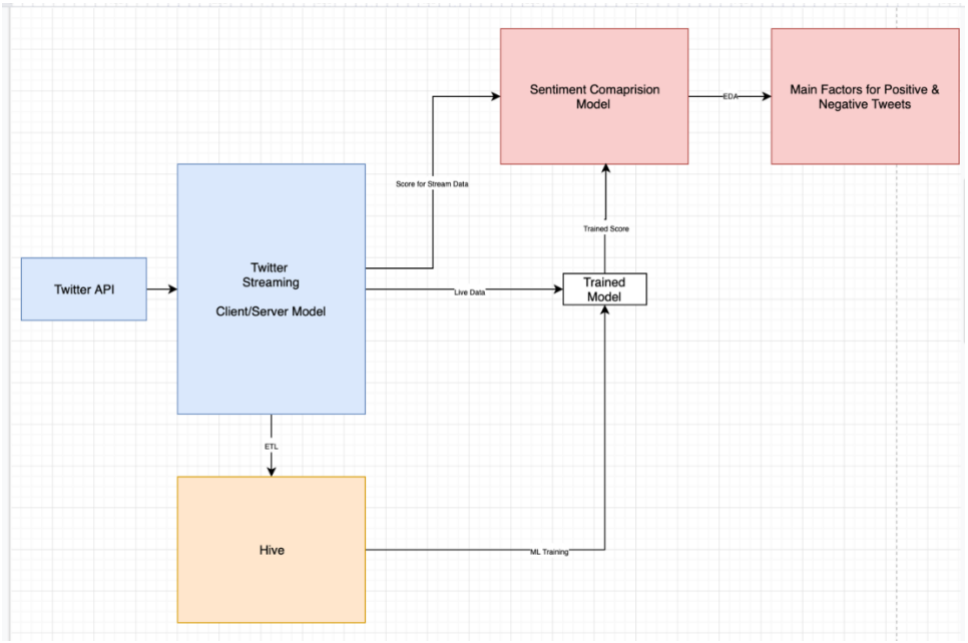
The project's features includes collecting real-time tweets from twitter streaming API, performing ETL (Pre-Processing the data, Extracting necessary information and loading the data in to the Hive), and using TextBlob, predict the sentiment for each tweet and with that train a new ML model with tweets as input and sentiment as output, to enhance the prediction of sentiment for each tweet.

Increment - 1

Dataset:

We are using twitter streaming API to collect near real-time tweets on a specific set of keywords. Each tweet is in JSON format, which is a key-value pairs. It has various information about a tweet like tweet text, who tweeted it, user information, location where the tweet originated, source of the tweet like Android, Iphone etc. We are collecting real time data from streaming API using Spark streaming context with a window of 20 seconds and the streaming API was able to download 500kb of data in which we are filtering for keywords like sports, cricket, football, hockey and we are extract ~80kb from that.

Detail design of Features:



Initially, we have created an account in Twitter Developers API. From the provided API tokens and credentials, we downloaded the tweets using the Spark Streaming application. Our project has a Client/Server kind of model where we got tweets using the Tweepy PY library which acted as the Server stage of our application. Now, we have utilized the Spark Streaming application to send the request and, on the success, the application would receive the tweets from the server on a window-based model.

Once the client receives the tweets on a windowing-based model then we will be mainly performing two operations. Firstly, on the stream data, we would show the sentiment on fly for each tweet that is streamed. Secondly, we would be storing the tweets into HIVE system after performing the set of transformations on the streamed tweets. After storing the data in HIVE, we would be building a classification model by training on the stored tweets. Once, the model is trained then we would run the model on real-time tweets and predict the sentiment.

The next stage of our project is to compare the scores provided by the model and the direct sentiment scores which were provided on the tweets. It also identifies the major keyword factors for a tweet to be categorized into positive or negative sentiment.

Analysis for Increment-1:

For downloading twitter data, we have created a twitter developer account, and we used tweepy python library and spark streaming context for downloading tweets. Each tweets if a raw json data, which has complex json formats as well, we pre-processed the data and converted the complex json to a simple text format which is each to handle and sent that to client side program through socket, from client side we transformed them in to table structure on which we can write hive queries to do some analysis. The analysis results are explained briefly in preliminary results.

Implementation:

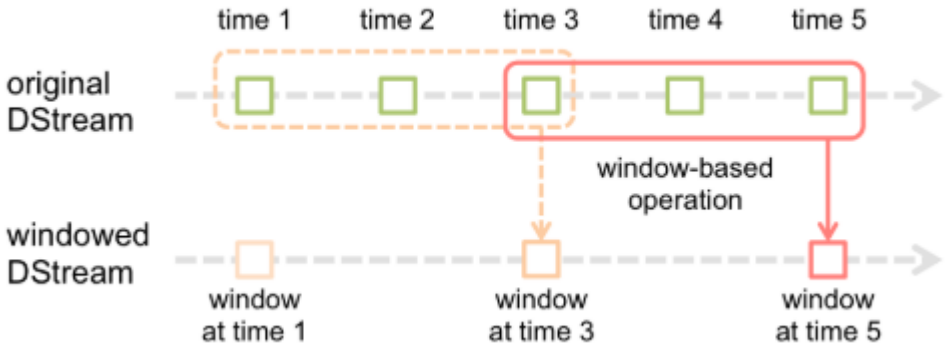
- **Client-Serve Implementation:**

  For downloading the streaming twitter tweets, we followed client-serve approach,

  In server side, we used tweepy library for connecting to twitter streaming API with required credentials which we got from twitter developer account. We used socket connection to transfer data from server to client. In server, the data which we got from twitter streaming

API is in JSON format, which we pre-processed it in server side and sends simple text data with required fields to the client side. The tweepy will keep on streaming data from twitter and puts the data in to the socket.

In client side, we used spark streaming context to collect the data from the socket, the spark streaming context is configured with a window length of 20 time units and sliding-interval of 20 seconds, which means for every 20 seconds spark reader will read next 20 windows of data.



- **Pre- Processing Data**:

  Once the data is observed by the client, we are applying map and reduce for transforming the data in to table form and storing it in a hive table. From the hive, it is easier to write Spark SQL queries and to perform some analysis on the data.

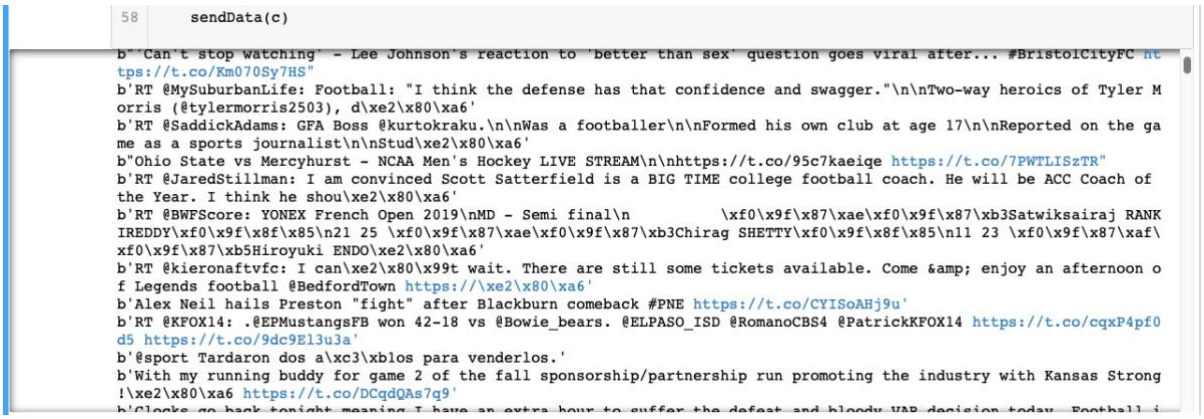
- **Tweet Analysis:**

  On the pre-processed data,

  1. Finding the top 10 hash tags in the timeframe.
  2. Finding the count of positive and negative tweets using Decision Rule analysis.
  3. Comparing the number of tweets tweeted for different games like cricket, football etc.
  4. Finding the most used URL's in tweets
  5. Comparing the number of tweets tweeted from different sources like iPhone, Android, Web App etc.
  6. Analysing the number of tweets originated from different locations.

Preliminary Results:

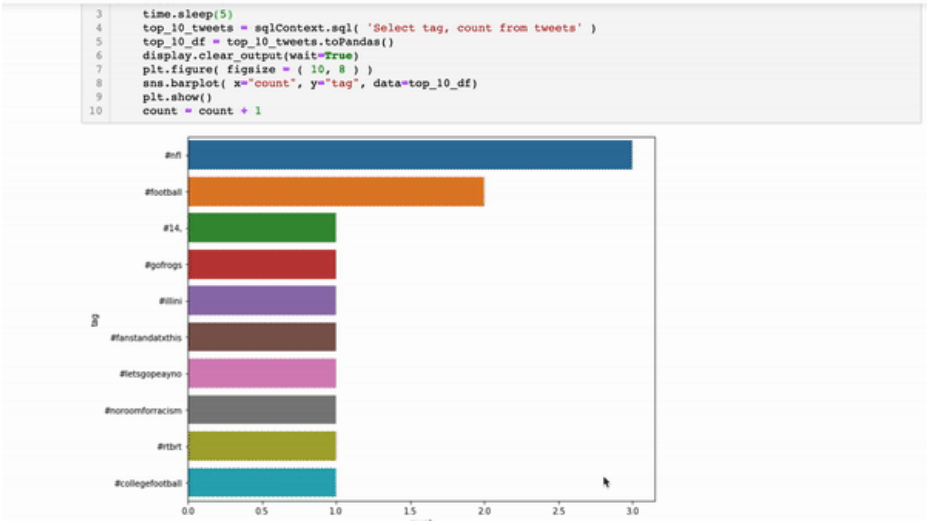
- Server Side Streaming:
    - We are able to connect to twitter streaming API and download the data in JSON format.



- Client Side Analysis:

  We have performed some analysis on the pre-processed data as explained below,

    - Finding the top 10 hash tags in the timeframe



The above graph describes the top 10 tags which have a number of counts during the window frame. So, we can clearly see that the tags which are #WordSeries and #FootBall has more number of tweets.

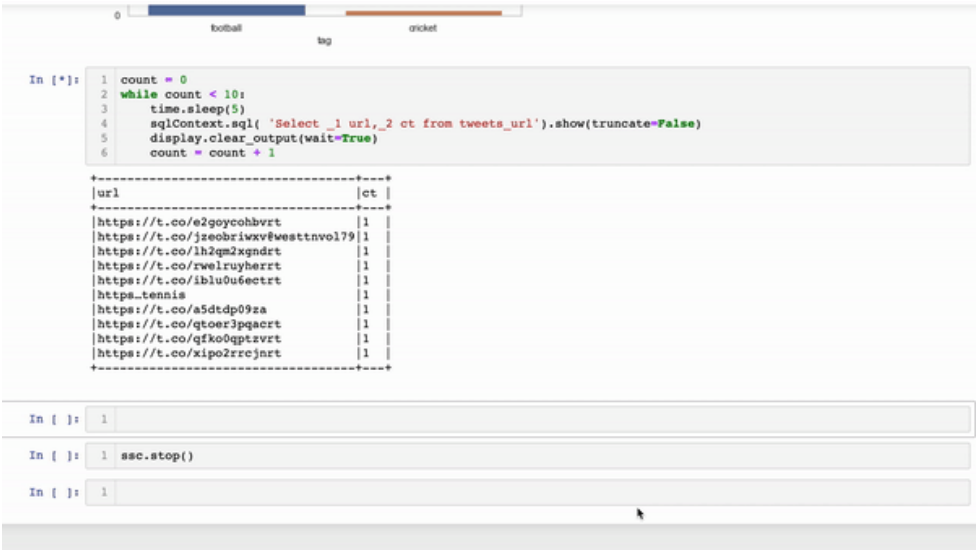
- Finding the count of positive and negative tweets using Decision Rule analysis

We have used the decision rule-based model, to segregate the tweets which would fall under the positive and negative sentiments. We have taken a set of words that would mainly separate the tweets. So, we can clearly say that negative tweets are more than positive tweets.

- Comparing the number of tweets tweeted for different games like cricket, football etc

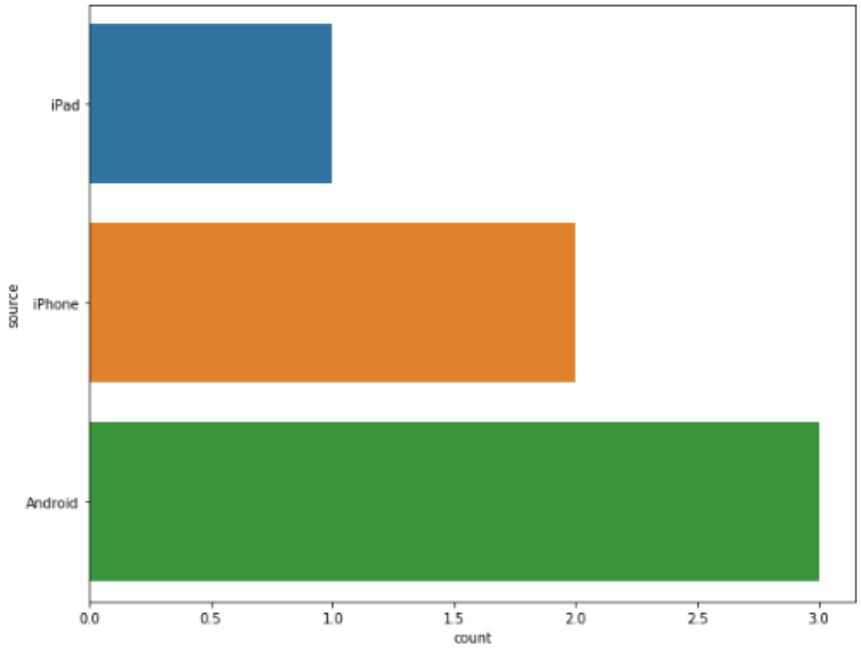
The above graph describes the game tweets which has more number of counts during the window frame. So we can clearly see that there number of tweets for the football game is more.

- Finding the most used URL's in tweets



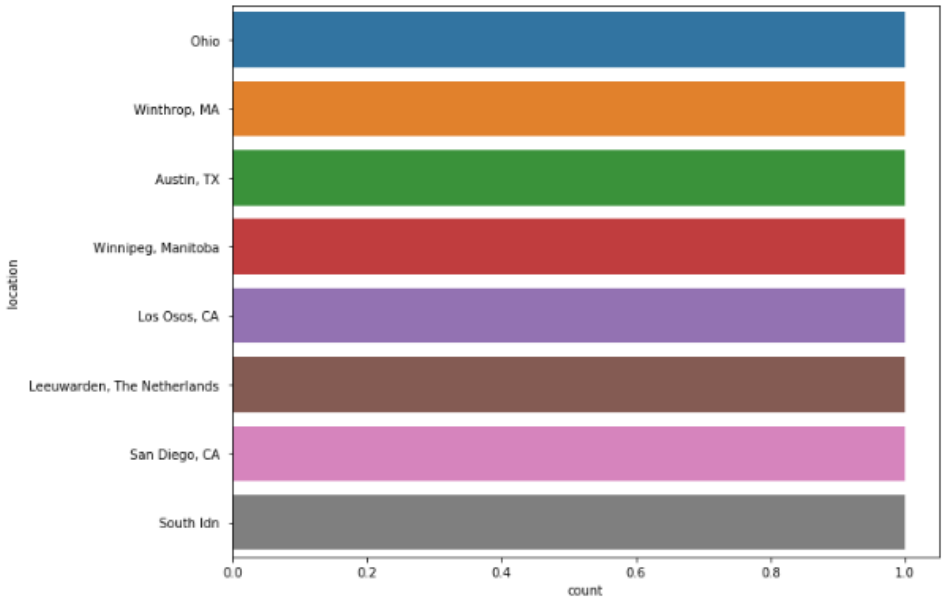
We have extracted the URL's from the tweets and performed the count on these URLs. So the above table displays the distinct URLs present in the tweets during the time frame.

- Comparing the number of tweets tweeted from different sources like iPhone, Android, Web App etc.



The above plot is for comparing the number of tweets tweeted from a particular source. From the above plot we can infer that more tweets are originated from Android devices.

- Analysing the number of tweets originated from different locations.



From the above plot we can conclude that more tweets are tweeted from ohio and Texas.

Project Management

Implementation status report

- **Work completed:**
  - **Downloading twitter streaming data**
    - Researching various ways of implementation **done by Hari**
    - Server side implementation **done by Hiresh**
    - Client side implementation **done by Neeraj**
  - **Analysis of Data:**
    - **Neeraj -** Finding the top 10 hash tags in the timeframe, Finding the count of positive and negative tweets using Decision Rule analysis.

        - **Hari -** Comparing the number of tweets tweeted for different games like cricket, football etc, Finding the most used URL's in tweets
        - **Hiresh -** Comparing the number of tweets tweeted from different sources like iPhone, Android, Web App etc, Analysing the number of tweets originated from different locations.
    - **Contributions (members/percentage)**

        - **Neeraj** – 35%
        - **Hiresh** – 35%
        - **Hari** – 30%

- **Work to be completed**
    - **Description**
        - Doing tweets ETL to HIVE system. Identifying the columns which needed to be loaded into HIVE and performing the transformations on the selected attributes.
        - Training the model and predicting the score on the Near Real-Time tweets
        - Utilizing the pre-trained models and providing the Sentiment Score for the Near Real-Time tweets.
        - Comparing the differences between the scores provided the pre-trained model and trained model.
        - Identifying the main factors for a tweet to be categorized to a positive and negative tweet.
    - **Responsibilities**
        - Neeraj – Performing the ETL and transformation of tweets to Hive system and training the model
        - Hiresh - Using the pre-trained model and providing the scores for the tweets. Comparing the scores between the models.
        - Hari – Doing ETL on the classified data and identifying the main factors for categorizing the tweets.

Increment – 2

Analysis of data

Saving and Loading the Streaming Data:

Using Spark Streaming Context, we are collecting the tweets for every 20 seconds and saving it into a local folder with the following options: coalesce=1, which will store the data in a single file; headers=true, store the file with headers. Once we collected the enough tweets to train a model, we are merging all the csv files using Glob and loading the merged file into Spark SQL context and creating hive external tables. We collected around 50k tweets. On this data, we used TextBlob to predict sentiment for each tweet and with this data we are training a machine learning model to predict sentiment on sports context.

Implementation

Model Training

We have followed the below process in order to train our Sentiment ML Model.

- Importing the Necessary libraries
- Processing the Data
- Initializing the Model Variables/Dividing the data/Fit and Validate the model

Importing the Necessary libraries

- We have used the below libraries to build our ML model which are as follows
- Pandas (For accessing the tweets, in order to manipulate and do analysis on tweets)
- Numpy (For performing the array operations on Pandas DF)

- Glob (For reading the multiple dataframes from a location)
- RE (For cleaning the data using the regex)
- Textblob (For identifying the polarity score of tweets)
- Sklearn.model_selection (For dividing the data into test/train)
- sklearn.feature_extraction.text (For using the TFIDF model)
- Imported the Sklearn's LogisticRegression, DecisionTreeClassifier, RandomForestClassifier and GradientBoostingClassifier to build various ML Models and the check the accuracy and to choose a best model.
- Imported sklearn.metrics accuracy_score and confusion_matrix to check the accuracy and f1score for the above models.
- Joblib (For dumping and loading the trained TFIDF and ML models)

Processing the Data

- We have loaded all the files into one single DF using the GLOB library.
- As tweets are free text, we just verified the percentage of non-alpha tweets which came around 6%.
- We have used a function in order to clean the tweets. As we feeding the tweets to a ML model. We need to remove the unnecessary words from the tweets. Using regex we have implemented this approach.
- Used the Textblob model in -order to get the polarity scores of the tweets and assigned to a new column in the pandas dataframe
- Created 2 new column in order to get the Sentiment score and description which is defined on the basis on the polarity score. If <=0 then we tagged it to the Negative sentiment and score to 0 whereas if it >0 we tagged it to the Positive sentiment and scores to 1

Initializing the Model Variables/Dividing the data/Fit and Validate the model

- Using the test_train_split we have divided the data into train and test splits
- Initialized the TFIDF vector and performed the fit and transformed in a single step on the train data.
- We got the features for train features and test features separately into different variables from TFIDF
- Initialized the Logistic Regression, Decision Tree Classifier, Random Forest Classifier and Gradient Boosting Classifier models
- Created a pipeline structure with each of the model with respective for each hyperparameters
- Now, we fitted the TFIDF train features for each of the classifier along with the sentiment description which has the positive and negative
- Next we predicted the ML models on the test features
- Observed the accuracy by all the models.
- Now, we downloaded the TFIDF and ML model(LR) using the jsonlib library

Sentiment Comparison Model (User Defined Functions):

We created three UDF's they are for,

- **Data Cleaning** – Removing junk characters like smileys, other language characters etc which may affect the performance of the ML model to be trained.
- **TextBlob Prediction** – This UDF will take tweet as a parameter and will call TextBlob function with the tweet, the TextBlob will provide sentiment polarity for the tweet. We made a threshold of 0 i.e. if polarity is less than 0 then the sentiment is 'Negative' otherwise 'Positive'.
- **Model Prediction** – Once we trained a ML model we are storing the trained ML model into a folder using jsonlib. In this UDF we are loading the model using jsonlib and predicting sentiment for a tweet and returning the sentiment.

Preliminary Results:

Sample Data:

Out[76]:

	tweet_txt
0	he had a clear lane to the end zone instead he...
1	he had a clear lane to the end zone instead he...
2	formula 1 mucho golf y buena carnâ□¦denmark ju...
3	then cruyff took over changed the way football...
4	uppgifter: city trappar upp jakten â□□ pÃ¥â□¦ ...

Sentiment Analysis using TextBlob

	tweet_txt	sentiment_value	sentiment_score	sentiment_description
0	he had a clear lane to the end zone instead he...	0.180000	0	positive
1	he had a clear lane to the end zone instead he...	0.072857	0	positive
2	formula mucho golf y buena carn denmark just w...	0.166667	0	positive
3	then cruyff took over changed the way football...	0.315000	0	positive
4	uppgifter city trappar upp jakten p https t co...	-0.250000	1	negative

Model Training:

```
Classifiers = {'lg':LogisticRegression(random_state=42,C=5,max_iter=200),\
               'dt':DecisionTreeClassifier(random_state=42,min_samples_leaf=1),\
               'rf':RandomForestClassifier(random_state=42,n_estimators=100,n_jobs=-1),\
               'gb':GradientBoostingClassifier(random_state=42,n_estimators=100,learning_rate=0.3)}
```

```
def ML_Pipeline(clf_name):
    clf = Classifiers[clf_name]
    fit = clf.fit(train_features,train['sentiment_description'])
    pred = clf.predict(test_features)
    Accuracy = accuracy_score(test['sentiment_description'],pred)
    Confusion_matrix = confusion_matrix(test['sentiment_description'],pred)
    print('==='*35)
    print('Accuracy of '+ clf_name +' is '+str(Accuracy))
    print('==='*35)
    print(Confusion_matrix)
```

Model Accuracy:

```
===================================================
Accuracy of lg is 0.8393316195372751
===================================================
[[415  86]
 [ 39 238]]
```

Sentiment Comparison Model

```
+--------------------+------------+-----------------+
|               tweet|custom_model|pre_trained_model|
+--------------------+------------+-----------------+
|@jimmybutler will...|    positive|         negative|
|                    |    negative|         negative|
|philippians 1:6 h...|    positive|         negative|
|                    |    negative|         negative|
|philippians 1:6 h...|    positive|         negative|
|11/29 10am-11:30...|    negative|         negative|
|$75 for 1… https:...|    positive|         positive|
|                    |    negative|         negative|
|@jimmybutler will...|    positive|         positive|
|sent with @nhl ht...|    positive|         negative|
|                    |    negative|         negative|
|@nconigliaronews ...|    positive|         positive|
|                    |    negative|         negative|
|   @ncaadii round ①|    negative|         negative|
|                    |    negative|         negative|
|@wingatefootball ...|    negative|         negative|
|                    |    negative|         negative|
|#onedog https://t...|    negative|         negative|
|“ i don’t really ...|    positive|         positive|
|                    |    negative|         negative|
+--------------------+------------+-----------------+
```

Project Management

Implementation status report

- **Work completed:**
  - **Downloading twitter streaming data**
    - Saving and Loading the Streaming Data **done by Hari**
    - Data Pre-Processing and UDFs creation **done by Hiresh**
    - Model Training **done by Neeraj**

  - **Contributions (members/percentage)**
    - **Neeraj** – 40%
    - **Hiresh** – 35%
    - **Hari** – 25%

- **Work to be completed**
  - **Description**
    - Training the model on the 1M tweet dataset, to check the accuracy score on the pre-trained models and the Hive dataset trained model.
    - Checking the accuracy on the pre-trained and hive data trained model.
    - Finding out the main factors for a tweet to be categorized to a positive and negative tweet and doing some EDA on it.
  - **Responsibilities**
    - Neeraj – Checking the accuracy
    - Hiresh - Training the model on the 1M tweet dataset
    - Hari – EDA

Increment – 3

Analysis of data

Implementation

Comparing TextBlob and Context Based Model

For comparing the accuracy of TextBlob and Context Based Model, we used Sentiment140 dataset, the data is in CSV. It has 6 columns:
0 - the polarity of the tweet (0 = negative, 2 = neutral, 4 = positive)
1 – tweet ID
2 – tweet date (Sat May 16 23:58:44 UTC 2009)
3 – the query (lyx.
4 – username (robotickilldozr)
5 – tweet text (Lyx is cool)

It has more than 1M, then we filtered it with the keywords related to sports and able to get 10K tweets. Then we pre-processed the tweets (Column naming, removing unwanted columns), applied TFIDF and trained a model (logistic regression).
Then we made prediction for the tweets using TextBlob and model trained previously to compare the performance of both the model and the results are as follow,

Model Name	Accuracy
TextBlob	40.01%
Context Based Model	88.25%

We got prediction accuracy for Textblob as 40% and context based model – 88.25%

Model's Accuracy

```
accuracy_score(df_filtered['sentiment_description'], df_filtered['tbsentiment_description'])
```

```
0.40015769761482356
```

```
accuracy_score(df_filtered['sentiment_description'], df_filtered['model_description'])
```

```
0.8825152769564361
```

From this, we can conclude that the context based model performs better than traditional global model.

Word Cloud Analysis Implementation

- We have categorized the tweets to be positive or negative tweets by using the data visualization technique called Word Cloud.
- Basically it represents the text data in which the size of each word indicates its frequency or its importance of the word.
- It is used to highlight the words in that category which are dominant.
- The process followed is we have imported the library which is word Cloud and Stopwords and initialized it.
- We have created a function called word cloud which takes the 2 parameters called source and stop words.
- Here source is the keywords that categorize the tweets based on it and stopwords are the list of words which are passed to this model, by this, we avoid the unimportant words from the model.

- We are creating dynamic Dataframe on the parameters passed, if for an instant positive and stop words yes is passed. Now, the data frame is created on this basis and passed to the model which given as the list of parameters to this model.
- If positive is passed then the White background is selected whereas the Black background is selected when it is negative as the parameter passed.
- We can define the image width and height along with the stopwords listed to the model.
- Stop words used are co, https, hey, hello and school which clearly does not segregate the tweets to be positive or negative tweets and it not the most import features as well.
- Major words for tweets to be positive are Football, Golf, King, High, Madrid
- Major Words for tweets to be negative are United, Hockey, quit, Pleasantville, cards
- We also fit the data using the random forest classifier which has an attribute called feature_importances_. This model fetches the importance of each word which is a score.
- By using this we can see the most important features and least important features for this model.
- Important features are football, rt, good, live, win
- Least important features are aa, aarod, aahmad, aalac, accept

Preliminary Results:

Stop Words:

```
stopwords.add('co')
stopwords.add('https')
stopwords.add('hey')
stopwords.add('hello')
stopwords.add('school')
```

Positive Words:

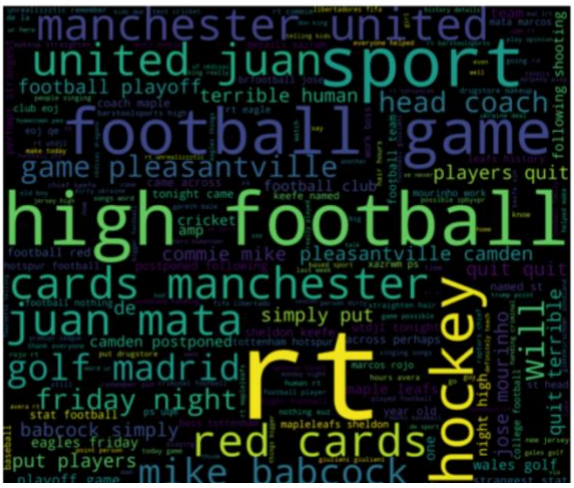
```
1 wordcloud('positive',"yes")
```

```
==========================================================================================
word cloud of positive is plotted below
```

Negative Words:

```
1 wordcloud('negative',"yes")
```

```
==========================================================================================
word cloud of negative is plotted below
```



Important Words:

	Word	Importance
4572	football	0.027384
11069	rt	0.025595
5819	https	0.021401
5149	good	0.017158
7389	live	0.012511
5418	hand	0.008793
3047	decided	0.008685
8651	new	0.008460
11109	running	0.008370
14067	win	0.007705
7326	like	0.007343
10756	respect	0.007231
6120	instead	0.007206
11316	score	0.007170
5121	golf	0.007159
1003	baseball	0.007138
1178	best	0.006942

Least Important Words:

```
1 impordf.loc[impordf['Importance']<=0.0]
```

	Word	Importance
0	aa	0.0
1	aaarod	0.0
5	aahmad	0.0
6	aalac	0.0
9	aaronwilson	0.0
10	aaspgmg	0.0
17	abbey	0.0
19	abbyhoffmann	0.0
22	abcnetwork	0.0
23	abcperth	0.0
25	aber	0.0
27	abk	0.0
29	abo	0.0
31	abonn	0.0
34	abreu	0.0

Project Management

Implementation status report

- **Work completed:**
  - **Comparing Text blob and Context Based Models**
    - Building the Model on 1M/EDA Visulaization **done by Neeraj**

        - Building the Model on 1M/Accuracy comparison **done by Hiresh**
        - Documentation/EDA on the visualization libraries **done by Hari**

    - **Contributions (members/percentage)**
        - **Neeraj** – 35%
        - **Hiresh** – 40%
        - **Hari** – 25%

References/Bibliography

- https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/intro-to-tweet-json
- https://docs.google.com/file/d/0B04GJPshIjmPRnZManQwWEdTZjg/edit
- http://help.sentiment140.com/for-students
- https://spark.apache.org/docs/latest/streaming-programming-guide.html
- https://www.rittmanmead.com/blog/2017/01/getting-started-with-spark-streaming-with-python-and-kafka/
- https://towardsdatascience.com/almost-real-time-twitter-sentiment-analysis-with-tweep-vader-f88ed5b93b1c