**Twitter Sentiment Analysis Using MapReduce**

**Overview of the Project**

• Access the Twitter Application Programming Interface (API) using python.

• Estimate the public's perception (the *sentiment*) of the tweet’s.

• Calculate the TF-DF (term frequency-document frequency) of terms in a tweet.

* Derive the location of each tweet

**Get Twitter Data**

• The **twitterstream.py** file is used to get live streaming twitter data.

The steps below will help to access the live 1% stream.

• To access the live stream, the **oauth2 library** is used to properly authenticate.

• Create a twitter account if you do not already have one. Respond to the resulting email from Twitter to confirm your Twitter Account.

• Go to **https://dev.twitter.com/apps** and log in with your twitter credentials.

• Click "**Create New App**”

• Fill out the form and agree to the terms. Put in a dummy website if you don't have one you want to use.

• On the next page, click the “**Keys & Access Tokens**” tab along the top, then scroll all the way down until you see the section "Your Access Token”

• Click the button "**Create My Access Token**". You can **Read more about Oauth authorization. (**https://pypi.python.org/pypi/oauth2/**)**

• You will now copy four values into **twitterstream.py**.

• "**API Key (Consumer Key)** “

• "**API secret (Consumer Secret)**“

• "**Access token**”

• "**Access token secret**”.

• Open twitterstream.py and you will see the code like below. Replace the corresponding values in the code.

api\_key = "<Enter api key>"

api\_secret = "<Enter api secret>"

access\_token\_key = "<Enter your access token key here>"

access\_token\_secret = "<Enter your access token secret here>”

• Run the following and make sure you see data flowing and that no errors occur.

**python twitterstream.py > nayak\_nandan\_tweet\_data.txt**

This command pipes the output to a file. Stop the program with Ctrl-C once the required amount of data is accumulated.

**Python MapReduce Framework and Execution**

A python library called **MapReduce.py** that implements the MapReduce programming model is used. The framework faithfully implements the MapReduce programming model, but it executes entirely on a single machine -- it does not involve parallel computation.

**Problem 1: Derive Tweet Sentiment Score**

• For this part, the **MapReduce.py** is usedas framework and **tweets\_sentiment.py** as a base for the script.

• Executing the code for the problem

**python nayak\_nandan\_tweets\_sentiment.py**

• The above file reads two files:

• AFINN-111.txt: AFINN-111.txt contains a list of pre-computed sentiment scores. Each line in the file contains a word or phrase followed by a tab separated sentiment score.

nayak\_nandan\_tweet\_data.txt: The tweet data file created earlier.

**Sample Output:**

nayak\_nandan\_output\_sentiment\_scores.txt

**Problem 2: TF-DF of each term (term frequency – document frequency)**

For this part, the **MapReduce.py** is usedas framework and **tweets\_tfdf.py** as a base for the script.

• Executing the code for the problem:

**python nayak\_nandan\_tweets\_tfdf.py**

• The above given reads one file:

**•** nayak\_nandan\_tweet\_data.txt: The tweet data file created earlier.

**Sample Output:**

nayak\_nandan\_output\_tfdf.txt

**Problem 3: Derive location of each tweet**

• Executing the code for the problem:

**python nayak\_nandan\_tweets\_location.py**

• The above given reads one file:

**•** nayak\_nandan\_tweet\_data.txt: The tweet data file created earlier.

**Sample Output:**

nayak\_nandan\_output\_location.txt

**Algorithm:**

**Problem 1: Derive Tweet Sentiment Score**

**Mapper Input:**

• The input is a python dictionary containing the tweets data: each\_tweet.

• Extract the ‘**text**’ from this tweet which is now stored in a dictionary. You can also read the Twitter documentation (https://dev.twitter.com/docs/platform-objects/tweets) to understand what information each tweet contains and how to access it, but it's not too difficult to deducethe structure by direct inspection.

• Now that the ‘text’ for each tweet is available, it is used to calculate the sentiment score of each tweet. For that you have to lookup each term of the tweet in the AFINN-111.txt dictionary and add up their respective sentiment scores.

• All the terms in AFINN-111.txt are lower case. So, use the **string.lower()** function to convert your tweet to all lower case for this problem.

• To encode the tweet string in utf-8 and remove ASCII punctuation, import the **string** library and use the following code:

**newstring = mystring.encode('utf-8').translate(None, string.punctuation)**

• Some tweets contain **URL’s** (eg: https:\/\/t.co\/8wl9wrT), **hashtags**(starting with #,@) and prefix like “**RT @ManUtd:”,** which should be ignored while calculating the sentiment score. For this purpose python’s **regular expression library(import re)** is used. Also the extra terms found in the tweets but not in AFINN-111.txt are given a **sentiment score of 0**.

**Reducer Output:**

• The script should **print to stdout** the sentiment of each tweet in the file, one numeric sentiment score per line. The first score corresponds to the first tweet, the second score corresponds to the second tweet, and so on. A sample output for reference is given below:

[1, 2.0] # Sentiment score of tweet1 is 2

[2, 0] # Sentiment score of tweet2 is 0

**Problem 2: TF-DF of each term(term frequency – document frequency)**

**Mapper Input:**

• The input is a python dictionary containing the tweets data: each\_tweet.

• Extract the ‘**text**’ from this tweet which is now stored in a dictionary. You can also read the Twitter documentation (https://dev.twitter.com/docs/platform-objects/tweets) to understand what information each tweet contains and how to access it, but it's not too difficult to deducethe structure by direct inspection.

• Now that the ‘text’ for each tweet is available, it is used to calculate the term frequency(tf) of each term of each tweet.

• Convert the string to lower case before calculating the tf. Also encode the string in utf-8 and remove ASCII punctuation by using:

• newstring = mystring.encode('utf-8').translate(None, string.punctuation)

• Some tweets contain **URL’s** (eg: https:\/\/t.co\/8wl9wrT), **hashtags**(starting with #,@) and prefix like “**RT @ManUtd:”,** which should be ignored while calculating the sentiment score. For this purpose python’s **regular expression library(import re)** is used.

**Reducer Output:**

The output is a list of <term, df, a list of <tweet\_no,tf>> for each term in one line. For example,

["a", 4, [[1, 1], [2, 3], [3, 1], [4, 2]]] # This shows that term “a” has df 4 and occurs in tweets 1, 2, 3, 4 with tf 1, 3, 1, 2 respectively.

["all", 1, [[3, 1]]] # This shows that term “all” has df 1 and occurs in tweet 3 with tf 1.

**Problem 3: Derive location of each tweet**

* Read the input file – nayak\_nandan\_tweet\_data.txt
* Read each tweet, if it has location enabled, perform the data processing as done before and then record the location against each presidential candidate and the no. of tweets from that location
* From the total tweets, calculate the fraction of tweets commented on each presidential candidate
* Display the results on the comment share