REAL TIME MINI PROJECT

GROUP 3

Sentiment Analysis on Twitter in Real Time

REAL TIME MINI PROJECT

PROBLEM STATEMENT

Sentiment Analysis on Twitter in Real Time

On the real time tweets streams data apply Sentiment Analysis to get the opinion & trends on different topics in real time. Also find influential people or POI (person of Interest) from the twitter data

Approach

This mini project deals with performing analysis on Real Time Twitter data using Python and Spark Streaming. We have used Twitter Streaming API, Spark Streaming Context and Flask Realtime Dashboards to solve each problem to solve each problem.
All questions in this mini project have been answered along with their corresponding visualizations.
This document consists of the following:
1. Python and Pyspark working code for all questions
2. Output for each code
3. Flask dashboard snapshots (Answer 8) for each question
4. Procedure to execute Python code, Spark code and Flask dashboard code

All global level analysis are performed by providing Global coordinates: `[-180,-90,180,90]`. These coordinates would technically cover tweets flowing across the world.

Specific filters such as “iPhone” are used only where mentioned in the questions.

Since the coordinates are global the output of most of the questions would provide generic insights rather than specific regional or conceptual insights with iPhone related analysis being an exception in specific questions.

TEAM MEMBERS

Sulekha Aloorravi

Sunil Vikram

Suraj Kumar

Siddhartha Murthy

Maheshwaran Sridharan

TABLE OF CONTENTS

1. FIND INFLUENTIAL PEOPLE IN TWITTER:

1.1. For simplicity assume the algorithm to find influential person is directly proportional to followers.

1.2. Find top 20 Influential personalities from the twitter across the globe.

```
#Python script
import os
import tweepy
from tweepy import OAuthHandler
from tweepy import Stream
from tweepy.streaming import StreamListener
import socket
import json
import requests
import requests_oauthlib

consumer_key = 'SntKtNL6CuaXQdz1Oa7ImyY5D'
consumer_secret = 'DI89nzONJClhLKxoHm9pUAk13aqstKHjTtIvKB0y7z0SHUzm16'
access_token = '911915238173188097-KH2VkrtKmo5UmAjehwxAZFMVDWzouuC'
access_secret = 'CAblWrnbvFLGNCPRDp8uvG3ZyD0GM6sAbCWXvlVItF9xl'

class TweetsListener(StreamListener):

    def __init__(self, csocket):
        self.client_socket = csocket

    def on_data(self, data):
        try:
            msg = json.loads(data)
            if 'user' in msg:
                self.client_socket.send(msg['user']['screen_name'].encode('utf-8')+'||'+str(msg['user']['followers_count'])+'||'
        except BaseException as e:
            print("Error on_data: %s" % str(e))
        return True
```

```
    def on_error(self, status):
        print(status)
        return True

def sendData(c_socket):
    auth = OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_token, access_secret)

    twitter_stream = Stream(auth, TweetsListener(c_socket))
    #This below mentioned location coordinates covers the whole world data
    twitter_stream.filter(locations=[-180,-90,180,90])

if __name__ == "__main__":
    s = socket.socket()     # Create a socket object
    host = "192.168.244.136"      # Get local machine name
    port = 5551             # Reserve a port for your service.
    s.bind((host, port))    # Bind to the port

    print("Listening on port: %s" % str(port))

    s.listen(5)                 # Now wait for client connection.
    c, addr = s.accept()        # Establish connection with client.

    print( "Received request from: " + str( addr ) )

    sendData( c )
```

```
Listening on port: 5551
Received request from: ('192.168.244.136', 54286)
```

```
#Spark Streaming
from __future__ import print_function
import sys
from pyspark import SparkContext
from pyspark.streaming import StreamingContext
from pyspark.sql import Row, SparkSession
import io
import requests

def getSparkSessionInstance(sparkConf):
    if ('sparkSessionSingletonInstance' not in globals()):
        globals()['sparkSessionSingletonInstance'] = SparkSession\
            .builder\
            .config(conf=sparkConf)\
            .getOrCreate()
    return globals()['sparkSessionSingletonInstance']


if __name__ == "__main__":
    host, port = sys.argv[1:]
    sc = SparkContext.getOrCreate()
    ssc = StreamingContext(sc, 5)

socket_stream = ssc.socketTextStream("192.168.244.136", 5551)

lines = socket_stream.window(60)

tweets = lines.flatMap(lambda line: line.split("\n"))
```

```
# Convert RDDs of the words DStream to DataFrame and run SQL query
def process(time, rdd):
        print("========= %s =========" % str(time))
        from pyspark.sql.types import NumericType
        try:
            # Get the singleton instance of SparkSession
            spark = getSparkSessionInstance(rdd.context.getConf())

            header = ["ScreenName","FollowersCount","Language","TweetsCount","CountryCode","RetweetCount"]
            rowRdd = rdd.map(lambda row : row.split("||"))
            jsonDataFrame = spark.createDataFrame(rowRdd, header)
            jsonDataFrame = jsonDataFrame.withColumn("FollowersCount",jsonDataFrame["FollowersCount"].cast("Int"))
            jsonDataFrame = jsonDataFrame.withColumn("TweetsCount",jsonDataFrame["TweetsCount"].cast("Int"))
            jsonDataFrame = jsonDataFrame.withColumn("RetweetCount",jsonDataFrame["RetweetCount"].cast("Int"))

            # Creates a temporary view using the DataFrame.
            jsonDataFrame.createOrReplaceTempView("tweets")

            # Do word count on table using SQL and print it
            #1.Find Influential people in twitter:
            #1.1 For simplicity assume the algorithm to find influential person is directly proportional to followers.
            #1.2 Find top 20 Influential personalities from the twitter across the globe.
            FollowersCountDF = \
            spark.sql("select ScreenName, FollowersCount from tweets order by FollowersCount desc limit 20")
            FollowersCountDF.show()
            send_df_to_dashboard(FollowersCountDF)

        except:
            pass
```

```
def send_df_to_dashboard(df):
    # extract the hashtags from dataframe and convert them into array
    top_tags = [str(t.ScreenName) for t in df.select("ScreenName").collect()]
    # extract the counts from dataframe and convert them into array
    tags_count = [p.FollowersCount for p in df.select("FollowersCount").collect()]
    # initialize and send the data through REST API
    url = 'http://192.168.244.136:5001/updateData'
    request_data = {'label': str(top_tags), 'data': str(tags_count)}
    response = requests.post(url, data=request_data)

tweets.foreachRDD(process)

ssc.start()

ssc.awaitTermination()
```

Output:

```
========= 2017-10-07 01:43:00 =========
+---------------+--------------+
|     ScreenName|FollowersCount|
+---------------+--------------+
|           lafm|       1882070|
|     yunusgunce|        107834|
|     TrendsLima|        106919|
|     grantstern|         28271|
|    nacho991960|         24708|
| BasquetManresa|         23841|
|     pfvrmikael|         19112|
| TrendsSantiago|         16263|
|   JoaquinNarro|         15367|
|   sandysamayra|         13232|
|    Pra_Raquell|         10509|
|     NJEstates2|         10343|
|    Hlomani_Job|          9962|
|amirali71457249|          8340|
|      Oilplated|          7322|
|  realgiftsambo|          7037|
|Germantownrunne|          6931|
|    arielldelis|          5145|
|    X_Mookieeee|          5040|
|       Tessa_25|          4847|
+---------------+--------------+

========= 2017-10-07 01:43:05 =========
+---------------+--------------+
|     ScreenName|FollowersCount|
+---------------+--------------+
|           lafm|       1882070|
|           sky1|        346457|
|       caval100|        122215|
|ObrasPublicasEc|        116983|
|     yunusgunce|        107834|
```

Insights:

These are the list of users with maximum followers across the world while we were executing the script.

This data will keep changing. This code is executed by providing the location coordinates as

```
[-180,-90,180,90]
```

For instance, if we need to get the followers for specific list of politicians, we need to provide only their names as input.

To make this output more meaningful in production environment, specific filters need to be provided while streaming.

2. GET TRENDING TOPICS IN TWITTER RIGHT NOW:

2.1 Find the word from the tweet which is occurring most of the times in the whole tweets corpus.
2.2 Top 10 trending topic

```
#2.1. Find the word from the tweet which is occurring most of the times in the whole tweets corpus.
#Python
def clean_tweet(tweet):
        return ' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)", " ", tweet).split())

class TweetsListener(StreamListener):

  def __init__(self, csocket):
      self.client_socket = csocket

  def on_data(self, data):
      try:
          msg = json.loads( data )
          #print( msg['text'].encode('utf-8') )
          tweet_text = clean_tweet(msg['text'])
          #print(tweet_text)
          self.client_socket.send( tweet_text + '\n' )
          return True
      except BaseException as e:
          print("Error on_data: %s" % str(e))
      return True

  def on_error(self, status):
      print(status)
      return True
```

```
def sendData(c_socket):
  auth = OAuthHandler(consumer_key, consumer_secret)
  auth.set_access_token(access_token, access_secret)

  twitter_stream = Stream(auth, TweetsListener(c_socket))
  #Use the below line to get whole world's trending topics
  #twitter_stream.filter(locations=[-180,-90,180,90])
  #Due to too much of tweets volume and execution time, I am using the below line to get output
  #twitter_stream.filter(track=['iphone'])
  twitter_stream.sample()
```

```
Listening on port: 7799
Received request from: ('192.168.244.136', 47812)
```

```
#Spark
nonempty_lines = lines.filter(lambda x: len(x) > 0)
words = nonempty_lines.flatMap(lambda x: x.split(" "))
def send_df_to_dashboard(df):
    # extract the hashtags from dataframe and convert them into array
    top_tags = [str(t.word_tweet) for t in df.select("word_tweet").collect()]
    # extract the counts from dataframe and convert them into array
    tags_count = [p.Count for p in df.select("Count").collect()]
    # initialize and send the data through REST API
    url = 'http://192.168.244.136:5001/updateData'
    request_data = {'label': str(top_tags), 'data': str(tags_count)}
    response = requests.post(url, data=request_data)
def get_tweets(time, rdd):
    print("========= %s =========" % str(time))
    from pyspark.sql.types import NumericType
    try:
        header = ["word_tweet"]
        spark = getSparkSessionInstance(rdd.context.getConf())
        Rowrdd = rdd.map(lambda x: x.split("\n"))
        jsonDataFrame = spark.createDataFrame(Rowrdd,header)
        df = jsonDataFrame.where(~jsonDataFrame.word_tweet.isin("RT","a","you","the","to","de","s","and","is","on","t","I","of","
        df.createOrReplaceTempView("tweets")
        TrendTweetDF = \
        spark.sql("select word_tweet,count(*) as Count from tweets group by word_tweet order by count(*) desc limit 10")
        TrendTweetDF.show()
        send_df_to_dashboard(TrendTweetDF)
    except:
            pass
words.foreachRDD(get_tweets)
ssc.start()
```

```
========= 2017-10-08 12:34:35 =========
+----------+-----+
|word_tweet|Count|
+----------+-----+
|        at|   30|
|         w|   10|
|       The|    9|
|        do|    9|
|        gt|    9|
|        my|    8|
|      your|    7|
|        he|    7|
|         3|    6|
|      like|    6|
+----------+-----+

========= 2017-10-08 12:34:40 =========
+----------+-----+
|word_tweet|Count|
+----------+-----+
|        at|   18|
|        my|    7|
|        we|    7|
|       are|    6|
|       The|    6|
|        10|    6|
|         3|    6|
|      shit|    5|
|       his|    5|
|       but|    5|
+----------+-----+
```

```
#2.2 Top 10 Trending Topics in real time
#Python
class TweetsListener(StreamListener):

  def __init__(self, csocket):
      self.client_socket = csocket

  def on_data(self, data):
      try:
          msg = json.loads( data )
          #print( msg['text'].encode('utf-8') )
          #tweet_text = clean_tweet(msg['text'])
          #print(tweet_text)
          self.client_socket.send( msg['text'].encode('utf-8') + '\n' )
          return True
      except BaseException as e:
          print("Error on_data: %s" % str(e))
      return True

  def on_error(self, status):
      print(status)
      return True

def sendData(c_socket):
  auth = OAuthHandler(consumer_key, consumer_secret)
  auth.set_access_token(access_token, access_secret)

  twitter_stream = Stream(auth, TweetsListener(c_socket))
  #Use the below line to get whole world's trending topics
  twitter_stream.filter(locations=[-180,-90,180,90])
  #Due to too much of tweets volume and execution time, I am using the below line to get output
  #twitter_stream.filter(track=['iphone'])
  #twitter_stream.sample()
```

```
#Spark
hashtags = lines.flatMap(lambda text: text.split(" ")).filter(lambda word: word.lower().startswith("#"))
def send_df_to_dashboard(df):
    # extract the hashtags from dataframe and convert them into array
    top_tags = [str(t.topic_tweet) for t in df.select("topic_tweet").collect()]
    # extract the counts from dataframe and convert them into array
    tags_count = [p.Count for p in df.select("Count").collect()]
    # initialize and send the data through REST API
    url = 'http://192.168.244.136:5001/updateData'
    request_data = {'label': str(top_tags), 'data': str(tags_count)}
    response = requests.post(url, data=request_data)
def get_tweets(time, rdd):
    print("========= %s =========" % str(time))
    from pyspark.sql.types import NumericType
    try:
        header = ["topic_tweet"]
        spark = getSparkSessionInstance(rdd.context.getConf())
        Rowrdd = rdd.map(lambda x: x.split("\n"))
        jsonDataFrame = spark.createDataFrame(Rowrdd,header)
        jsonDataFrame.createOrReplaceTempView("tweets")
        TrendTweetDF = \
        spark.sql("select topic_tweet,count(*) as Count from tweets group by topic_tweet order by count(*) desc limit 10")
        TrendTweetDF.show()
        send_df_to_dashboard(TrendTweetDF)
    except:
            pass
hashtags.foreachRDD(get_tweets)
ssc.start()
```

```
========= 2017-10-08 03:53:45 =========
+----------------+-----+
|     topic_tweet|Count|
+----------------+-----+
|          #Hiring|    9|
|             #job|    5|
|       #CareerArc|    4|
|         #ARKvsSC|    4|
|             #Job|    3|
|            #Jobs|    2|
|            #job?|    2|
|     #Automotive|    2|
|#ALDUBHeartsCare|    2|
|        #foodporn|    2|
+----------------+-----+

========= 2017-10-08 03:53:50 =========
+----------------+-----+
|     topic_tweet|Count|
+----------------+-----+
|             #job|    9|
|          #Hiring|    9|
|            #job?|    4|
|       #CareerArc|    3|
|        #foodporn|    2|
|        #Houston,|    2|
|         #ARKvsSC|    2|
|#ALDUBHeartsCare|    2|
|             #Job|    2|
|#CustomerService|    2|
+----------------+-----+
```

Insights:

Even though majority of the prepositions are removed through filters in Spark code, we still have grammatical words in English as most tweeted words in this output. This demonstrates that English is the most widely used language (no language filter is provided in the code) and more prepositions, adjectives and pronouns are used in tweets compared to the actual message that needs to be conveyed.

Trending topics around the world while executing this code was **Job searches** and **food**.

As per observations, most of the times these were the trending topics discussed across unless something specifically significant incident happens. On the day of Las Vegas attack, I observed the Hashtag of **Las Vegas** occurring maximum number of times.

3. SENTIMENT ANALYSIS:

3.1. Filter tweets & take tweets which has mention of “iPhone”.

3.2. Get sentiments about iPhone every hour.

```python
#Python
import os
import tweepy
from tweepy import OAuthHandler
from tweepy import Stream
from tweepy.streaming import StreamListener
import socket
import json
import requests
import requests_oauthlib

consumer_key = 'SntKtNL6CuaXQdz1Oa7ImyY5D'
consumer_secret = 'DI89nzONJClhLKxoHm9pUAk13aqstKHjTtIvKB0y7z0SHUzm16'
access_token = '911915238173188097-KH2VkrtKmo5UmAjehwxAZFMVDWzouuC'
access_secret = 'CAblWrnbvFLGNCPRDp8uvG3ZyD0GM6sAbCWXvlVItF9xl'

class TweetsListener(StreamListener):

    def __init__(self, csocket):
        self.client_socket = csocket

    def on_data(self, data):
        try:
            msg = json.loads(data)
            #if 'user' in msg:
            self.client_socket.send(msg['text'].encode('utf-8')+'\n')
        except BaseException as e:
            print("Error on_data: %s" % str(e))
        return True
```

```
    def on_error(self, status):
        print(status)
        return True

def sendData(c_socket):
    auth = OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_token, access_secret)

    twitter_stream = Stream(auth, TweetsListener(c_socket))
    twitter_stream.filter(track=['iphone'])
    #twitter_stream.filter(locations=[-180,-90,180,90])
    #twitter_stream.sample()
if __name__ == "__main__":
    s = socket.socket()     # Create a socket object
    host = "192.168.244.136"      # Get local machine name
    port = 5555            # Reserve a port for your service.
    s.bind((host, port))    # Bind to the port

    print("Listening on port: %s" % str(port))

    s.listen(5)                 # Now wait for client connection.
    c, addr = s.accept()        # Establish connection with client.

    print( "Received request from: " + str( addr ) )

    sendData( c )
```

```
Listening on port: 5555
Received request from: ('192.168.244.136', 53566)
```

```
#Spark
from __future__ import print_function
import sys
from pyspark import SparkContext
from pyspark.streaming import StreamingContext
from pyspark.sql import Row, SparkSession
import io
import requests
import re
from textblob import TextBlob

def getSparkSessionInstance(sparkConf):
    if ('sparkSessionSingletonInstance' not in globals()):
        globals()['sparkSessionSingletonInstance'] = SparkSession\
            .builder\
            .config(conf=sparkConf)\
            .getOrCreate()
    return globals()['sparkSessionSingletonInstance']


if __name__ == "__main__":
    #if len(sys.argv) != 3:
     #   print("Usage: sql_network_wordcount.py <hostname> <port> ", file=sys.stderr)
      #  exit(-1)
    #host, port = sys.argv[1:]
    sc = SparkContext.getOrCreate()
    #Enter 3600 to get iphone sentiments for every hour.
    ssc = StreamingContext(sc, 60)
    #To receive more and quicker output for reporting purpose I have used 60 here.

socket_stream = ssc.socketTextStream("192.168.244.136", 5555)
```

```
#Enter 3600 to get iphone sentiments for every hour.
#To receive more and quicker output for reporting purpose I have used 60 here.
lines = socket_stream.window(60)

def clean_tweet(tweet):
        return ' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)", " ", tweet).split())

def get_tweet_sentiment(tweet):
        # create TextBlob object of passed tweet text
        analysis = TextBlob(clean_tweet(tweet))
        # set sentiment
        if analysis.sentiment.polarity > 0:
            return 'positive'
        elif analysis.sentiment.polarity == 0:
            return 'neutral'
        else:
            return 'negative'

def send_df_to_dashboard(df):
    # extract the hashtags from dataframe and convert them into array
    top_tags = [str(t.Sentiment) for t in df.select("Sentiment").collect()]
    # extract the counts from dataframe and convert them into array
    tags_count = [p.Count for p in df.select("Count").collect()]
    # initialize and send the data through REST API
    url = 'http://192.168.244.136:5001/updateData'
    request_data = {'label': str(top_tags), 'data': str(tags_count)}
    response = requests.post(url, data=request_data)

tweety = lines.flatMap(lambda line: line.split("\n"))
```

```
def get_tweets(time, rdd):
        print("========= %s =========" % str(time))
        from pyspark.sql.types import NumericType
        try:
            header = ["Tweet"]
            spark = getSparkSessionInstance(rdd.context.getConf())
            spark.udf.register('get_tweet_sentiment', get_tweet_sentiment)
            rowRdd = rdd.map(lambda row: row.split("\n"))
            jsonDataFrame = spark.createDataFrame(rowRdd, header)
            jsonDataFrame.createOrReplaceTempView("tweets")
            #2. Sentiment Analysis:
            #2.1 Filter tweets & take tweets which has mention of “iphone”.
            iPhoneTweetDF = \
            spark.sql("select Tweet, get_tweet_sentiment(Tweet) as Sentiment from tweets")
            iPhoneTweetDF.show()
            sentiCountDF = \
            spark.sql("select get_tweet_sentiment(Tweet) as Sentiment, count(get_tweet_sentiment(Tweet)) as Count from tweets
            sentiCountDF.show()
            send_df_to_dashboard(sentiCountDF)
        except:
            pass

tweety.foreachRDD(get_tweets)
ssc.start()
ssc.awaitTermination()
```

Output:

```
+--------------------+---------+
|               Tweet|Sentiment|
+--------------------+---------+
|モイ！初見さんどうぞiPhoneか...|  neutral|
|モイ！iPhoneからキャス配信中...|  neutral|
|ジョナサン・アイブ氏が語った「iP...|  neutral|
|¡Participo en el ...|  neutral|
|¿Nos das un RT? #...|  neutral|
|RT @Apple: iPhone...| positive|
|RT @LeshaLappar: ...|  neutral|
|также делаю ремон...|  neutral|
|           やべ、電池切れだぜ|  neutral|
|iPhoneを電池切れっていう表現...|  neutral|
|RT @businessinsid...|  neutral|
|RT @yannoujr10: L...|  neutral|
|RT @iPhone8Promo1...| positive|
|https://t.co/wZQU...|  neutral|
|                    |  neutral|
|       #280character|  neutral|
|  #NationalPoetryDay|  neutral|
|RT @MochiManggae:...|  neutral|
|RT @MochiManggae:...|  neutral|
|Mobioutlet Apple ...| negative|
+--------------------+---------+
only showing top 20 rows

+---------+-----+
|Sentiment|Count|
+---------+-----+
| positive|   22|
|  neutral|  670|
| negative|    9|
+---------+-----+
```

Insights

From the above output it is evident that majority of population is neutral about iPhones even though positive sentiments are larger compared to negative. This kind of analysis would help manufacturers to decide on public opinion regarding their products and create or modify products accordingly.

4. GEO BASED ANALYSIS:

4.1. Country based tweets count.
4.2. Top countries in terms of Tweets Volume.
4.3. List of top 15 countries in terms of tweet counts.

```
#Python
import os
import tweepy
from tweepy import OAuthHandler
from tweepy import Stream
from tweepy.streaming import StreamListener
import socket
import json
import requests
import requests_oauthlib

consumer_key = 'SntKtNL6CuaXQdz1Oa7ImyY5D'
consumer_secret = 'DI89nzONJClhLKxoHm9pUAk13aqstKHjTtIvKB0y7z0SHUzm16'
access_token = '911915238173188097-KH2VkrtKmo5UmAjehwxAZFMVDWzouuC'
access_secret = 'CAblWrnbvFLGNCPRDp8uvG3ZyD0GM6sAbCWXvlVItF9xl'

class TweetsListener(StreamListener):

    def __init__(self, csocket):
        self.client_socket = csocket

    def on_data(self, data):
        try:
            msg = json.loads(data)
            if 'user' in msg:
                self.client_socket.send(msg['user']['screen_name'].encode('utf-8')+'||'+str(msg['user']['followers_count'])+'||'
            return True
        except BaseException as e:
            print("Error on_data: %s" % str(e))
        return True

    def on_error(self, status):
        print(status)
```

```
        return True

def sendData(c_socket):
    auth = OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_token, access_secret)

    twitter_stream = Stream(auth, TweetsListener(c_socket))
    #This below mentioned location coordinates covers the whole world data
    twitter_stream.filter(locations=[-180,-90,180,90])

if __name__ == "__main__":
    s = socket.socket()     # Create a socket object
    host = "192.168.244.136"      # Get local machine name
    port = 5552             # Reserve a port for your service.
    s.bind((host, port))    # Bind to the port

    print("Listening on port: %s" % str(port))

    s.listen(5)                 # Now wait for client connection.
    c, addr = s.accept()        # Establish connection with client.

    print( "Received request from: " + str( addr ) )

    sendData( c )
```

```
#Spark
from __future__ import print_function
import sys
from pyspark import SparkContext
from pyspark.streaming import StreamingContext
from pyspark.sql import Row, SparkSession
import io
import requests

def getSparkSessionInstance(sparkConf):
    if ('sparkSessionSingletonInstance' not in globals()):
        globals()['sparkSessionSingletonInstance'] = SparkSession\
            .builder\
            .config(conf=sparkConf)\
            .getOrCreate()
    return globals()['sparkSessionSingletonInstance']


if __name__ == "__main__":
    sc = SparkContext.getOrCreate()
    ssc = StreamingContext(sc, 5)

socket_stream = ssc.socketTextStream("192.168.244.136", 5552)

lines = socket_stream.window(60)

tweets = lines.flatMap(lambda line: line.split("\n"))
```

```
# Convert RDDs of the words DStream to DataFrame and run SQL query
def process(time, rdd):
        print("========= %s =========" % str(time))
        from pyspark.sql.types import NumericType
        try:
            # Get the singleton instance of SparkSession
            spark = getSparkSessionInstance(rdd.context.getConf())

            header = ["ScreenName","FollowersCount","Language","TweetsCount","CountryCode"]
            rowRdd = rdd.map(lambda row : row.split("||"))
            jsonDataFrame = spark.createDataFrame(rowRdd, header)
            jsonDataFrame = jsonDataFrame.withColumn("FollowersCount",jsonDataFrame["FollowersCount"].cast("Int"))
            jsonDataFrame = jsonDataFrame.withColumn("TweetsCount",jsonDataFrame["TweetsCount"].cast("Int"))


            # Creates a temporary view using the DataFrame.
            jsonDataFrame.createOrReplaceTempView("tweets")

            #4.Geo based analysis:
            #4.1 Country based tweets count.
            #4.3 List of top 15 countries in terms of tweet counts.
            CountryCountDF = \
            spark.sql("select CountryCode,TweetsCount  from tweets")
            data = CountryCountDF.groupby('CountryCode').agg({'TweetsCount': 'sum'})
            oldColumns = data.schema.names
            newColumns = ["CountryCode", "TweetVolume"]
            TweetVolumedf = reduce(lambda data, idx: data.withColumnRenamed(oldColumns[idx], newColumns[idx]), xrange(len(oldColu
            TweetVolumedf.createOrReplaceTempView("tweetsvol")
            vol_df = \
            spark.sql("select CountryCode, TweetVolume from tweetsvol order by TweetVolume desc limit 15")
            print("Tweets Volume by Country")
            vol_df.show()
            send_df_to_dashboard1(vol_df)
```

```
def send_df_to_dashboard1(df):
    # extract the hashtags from dataframe and convert them into array
    top_tags = [str(t.CountryCode) for t in df.select("CountryCode").collect()]
    # extract the counts from dataframe and convert them into array
    tags_count = [p.TweetVolume for p in df.select("TweetVolume").collect()]
    # initialize and send the data through REST API
    url = 'http://192.168.244.136:5001/updateData'
    request_data = {'label': str(top_tags), 'data': str(tags_count)}
    response = requests.post(url, data=request_data)

def send_df_to_dashboard2(df):
    # extract the hashtags from dataframe and convert them into array
    top_tags = [str(t.CountryCode) for t in df.select("CountryCode").collect()]
    # extract the counts from dataframe and convert them into array
    tags_count = [p.TweetCount for p in df.select("TweetCount").collect()]
    # initialize and send the data through REST API
    url = 'http://192.168.244.136:5002/updateData'
    request_data = {'label': str(top_tags), 'data': str(tags_count)}
    response = requests.post(url, data=request_data)

tweets.foreachRDD(process)

ssc.start()

ssc.awaitTermination()
```

Output – Tweet Volume:

```
========== 2017-10-07 17:06:00 ==========
+-----------+-----------+
|CountryCode|TweetVolume|
+-----------+-----------+
|         JP|    1177246|
|         BR|    1092785|
|         GB|     789087|
|         AR|     561868|
|         US|     521329|
|         ES|     441202|
|         PH|     423445|
|         KW|     280345|
|         FR|     269826|
|         TH|     251683|
|         CL|     248782|
|         UY|     228050|
|         DE|     202578|
|         MY|     196865|
|         PT|     171694|
+-----------+-----------+
```

Output – Tweet Count:

```
========= 2017-10-07 17:15:55 =========
+-----------+----------+
|CountryCode|TweetCount|
+-----------+----------+
|         US|        18|
|         JP|        12|
|         PH|        11|
|         BR|        11|
|         GB|        10|
|         ES|         7|
|         FR|         7|
|         ZA|         5|
|         SE|         3|
|         PT|         3|
|         AU|         3|
|         ID|         2|
|         MY|         2|
|         AT|         2|
|         IN|         2|
+-----------+----------+

========= 2017-10-07 17:16:00 =========
+-----------+----------+
|CountryCode|TweetCount|
+-----------+----------+
|         US|        44|
|         BR|        35|
|         JP|        33|
|         GB|        28|
|         PH|        24|
|         FR|        15|
|         ES|        15|
|         TH|        10|
|         MY|         9|
```

Insights

When we look at the overall tweet volume across the world, Japan has got the largest volume of tweets at real time.

When we look at the trends of tweet counts US has more number of tweets happening at the time this code was executed and India had the least within the first 15 countries.

5. FIND TOP TWEETING USER

5.1. Find User who is tweeting a lot.

5.2. Find top 50 across the world.

```
#Python
import os
import tweepy
from tweepy import OAuthHandler
from tweepy import Stream
from tweepy.streaming import StreamListener
import socket
import json
import requests
import requests_oauthlib

consumer_key = 'SntKtNL6CuaXQdz1Oa7ImyY5D'
consumer_secret = 'DI89nzONJClhLKxoHm9pUAk13aqstKHjTtIvKB0y7z0SHUzm16'
access_token = '911915238173188097-KH2VkrtKmo5UmAjehwxAZFMVDWzouuC'
access_secret = 'CAblWrnbvFLGNCPRDp8uvG3ZyD0GM6sAbCWXvlVItF9xl'

class TweetsListener(StreamListener):

    def __init__(self, csocket):
        self.client_socket = csocket

    def on_data(self, data):
        try:
            msg = json.loads(data)
            if 'user' in msg:
                self.client_socket.send(msg['user']['screen_name'].encode('utf-8')+'||'+str(msg['user']['followers_count'])+'||'
            return True
        except BaseException as e:
            print("Error on_data: %s" % str(e))
        return True
```

```
    def on_error(self, status):
        print(status)
        return True

def sendData(c_socket):
    auth = OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_token, access_secret)

    twitter_stream = Stream(auth, TweetsListener(c_socket))
    #This below mentioned location coordinates covers the whole world data
    twitter_stream.filter(locations=[-180,-90,180,90])

if __name__ == "__main__":
    s = socket.socket()     # Create a socket object
    host = "192.168.244.136"      # Get local machine name
    port = 5559             # Reserve a port for your service.
    s.bind((host, port))    # Bind to the port

    print("Listening on port: %s" % str(port))

    s.listen(5)                 # Now wait for client connection.
    c, addr = s.accept()        # Establish connection with client.

    print( "Received request from: " + str( addr ) )

    sendData( c )
```

```
Listening on port: 5559
Received request from: ('192.168.244.136', 56914)
```

```
#Below coding is the Spark logic and I am not repeating connection attributes
tweets = lines.flatMap(lambda line: line.split("\n"))
# Convert RDDs of the words DStream to DataFrame and run SQL query
def process(time, rdd):
        print("========= %s =========" % str(time))
        from pyspark.sql.types import NumericType
        try:
            # Get the singleton instance of SparkSession
            spark = getSparkSessionInstance(rdd.context.getConf())

            header = ["ScreenName","FollowersCount","Language","TweetsCount","CountryCode"]
            rowRdd = rdd.map(lambda row : row.split("||"))
            jsonDataFrame = spark.createDataFrame(rowRdd, header)
            jsonDataFrame = jsonDataFrame.withColumn("FollowersCount",jsonDataFrame["FollowersCount"].cast("Int"))
            jsonDataFrame = jsonDataFrame.withColumn("TweetsCount",jsonDataFrame["TweetsCount"].cast("Int"))


            # Creates a temporary view using the DataFrame.
            jsonDataFrame.createOrReplaceTempView("tweets")
            #5. Find Top tweeting user:
            #5.1 Find User who is tweeting a lot.
            #5.2 Find top 50 across the world.

            TweetCountDF = \
            spark.sql("select ScreenName,TweetsCount  from tweets order by TweetsCount desc limit 50")
            TweetCountDF.show()
            send_df_to_dashboard(TweetCountDF)
```

```
def send_df_to_dashboard(df):
    # extract the hashtags from dataframe and convert them into array
    top_tags = [str(t.ScreenName) for t in df.select("ScreenName").collect()]
    # extract the counts from dataframe and convert them into array
    tags_count = [p.TweetsCount for p in df.select("TweetsCount").collect()]
    # initialize and send the data through REST API
    url = 'http://192.168.244.136:5001/updateData'
    request_data = {'label': str(top_tags), 'data': str(tags_count)}
    response = requests.post(url, data=request_data)
tweets.foreachRDD(process)
ssc.start()
ssc.awaitTermination()
```

```
========= 2017-10-08 13:19:05 =========
+---------------+-----------+
|     ScreenName|TweetsCount|
+---------------+-----------+
|          511NY|    2542288|
| 511NYMidHudson|     452360|
|   SvobodaRadio|     207597|
|    TonyMatsuda|     197591|
|     goisutokki|     182338|
|      g008c1088|     162981|
|   SeanathonOAK|     144895|
|    saraltareen|     117613|
| Jasonkessler16|      80792|
|  Caligraphy215|      74486|
|mili_sagardoyOk|      63985|
|    Nikkopayton|      61319|
|   endzoneblog_|      57070|
|       melkeior|      56654|
|    MrOddFuture|      56581|
|       Dire27Li|      56558|
|Marceldesrosie1|      54436|
| iPreferPurpHoe|      51690|
| shaafiqahramli|      49305|
|    1_libriana_|      41807|
|     marocasouz|      41019|
|       _iAmBuck|      33659|
|takatayoshitake|      31823|
|        LGJRDNS|      29529|
|   Tina7Dubovec|      24037|
|   silvarafaelb|      23594|
|   ReivieBasten|      21525|
|Meli_guadalupee|      21466|
|      CuddyBruh|      21090|
|    uekiyaryoen|      19895|
|oksanacherednik|      18298|
|         horozz|      17799|
|    lizzydeLoca|      17290|
|    MeluDiaz_Ok|      17284|
|      asrofuu24|      15320|
|      evaraujos|      15028|
|       mayra909|      14399|
|     prynxhenry|      12339|
|    Peterlowery|      12267|
|      jonahuhdg|      11371|
|   PeacefulPure|      10623|
|      xxtaniaxx|      10510|
|       ryujisea|      10413|
|     judeeflick|      10148|
| abdullahgotah1|       9945|
|  rasheed222012|       9771|
|       Swedrick|       9283|
|  AlbionnKurtaj|       9219|
|astridalcocermn|       7782|
|     girlsonbws|       7018|
+---------------+-----------+
```

Insights

We can here see the list of 50 users with their corresponding tweet counts across the world. As the number of tweets by these users increases, this output tends to keep changing in runtime.

User analysis alongside region based, concept based and language based analysis would give a multitude of perspectives on the opinions of people across the world.

6. LANGUAGE BASED ANALYSIS

6.1. Find counts of tweets in different languages.

6.2. Get top 7 languages in terms of volume.

```
#Python
#I have removed all repititive code and mentioning only the logical part here
def on_data(self, data):
        try:
            msg = json.loads(data)
            if 'user' in msg:
                self.client_socket.send(msg['user']['screen_name'].encode('utf-8')+'||'+str(msg['user']['followers_count'])+'||'
            return True
        except BaseException as e:
            print("Error on_data: %s" % str(e))
        return True
```

```
Listening on port: 5559
Received request from: ('192.168.244.136', 59738)
```

```
#Spark
#6.1. Find counts of tweets in different languages.

def process(time, rdd):
        print("========= %s =========" % str(time))
        from pyspark.sql.types import NumericType
        try:
            # Get the singleton instance of SparkSession
            spark = getSparkSessionInstance(rdd.context.getConf())

            header = ["ScreenName","FollowersCount","Language","TweetsCount","CountryCode"]
            rowRdd = rdd.map(lambda row : row.split("||"))
            jsonDataFrame = spark.createDataFrame(rowRdd, header)
            jsonDataFrame = jsonDataFrame.withColumn("FollowersCount",jsonDataFrame["FollowersCount"].cast("Int"))
            jsonDataFrame = jsonDataFrame.withColumn("TweetsCount",jsonDataFrame["TweetsCount"].cast("Int"))

            # Creates a temporary view using the DataFrame.
            jsonDataFrame.createOrReplaceTempView("tweets")

            LangDF = \
            spark.sql("select Language,TweetsCount  from tweets")
            #4.Geo based analysis:
            #4.2 Top countries in terms of Tweets Volume
            data = LangDF.groupby('Language').agg({'TweetsCount': 'count'})
            oldColumns = data.schema.names
            newColumns = ["Language", "TweetCount"]
            TweetVolumedf = reduce(lambda data, idx: data.withColumnRenamed(oldColumns[idx], newColumns[idx]), xrange(len(oldColu
            TweetVolumedf.createOrReplaceTempView("tweetscnt")
            vol_df = \
            spark.sql("select Language, TweetCount from tweetscnt order by TweetCount desc limit 15")
            vol_df.show()
            send_df_to_dashboard(vol_df)
```

```
def send_df_to_dashboard(df):
    # extract the hashtags from dataframe and convert them into array
    top_tags = [str(t.Language) for t in df.select("Language").collect()]
    # extract the counts from dataframe and convert them into array
    tags_count = [p.TweetCount for p in df.select("TweetCount").collect()]
    # initialize and send the data through REST API
    url = 'http://192.168.244.136:5001/updateData'
    request_data = {'label': str(top_tags), 'data': str(tags_count)}
    response = requests.post(url, data=request_data)

tweets.foreachRDD(process)
ssc.start()
ssc.awaitTermination()
```

Output:

```
========= 2017-10-07 18:32:10 =========
+--------+----------+
|Language|TweetCount|
+--------+----------+
|      en|       586|
|      pt|       132|
|      ja|       102|
|      es|        81|
|      tr|        35|
|      ar|        23|
|      fr|        19|
|      ru|        16|
|      th|        13|
|      it|        12|
|      id|        10|
|      de|         9|
|      nl|         7|
|   en-gb|         5|
|      ca|         4|
+--------+----------+
```

```
========= 2017-10-07 18:32:15 =========
+--------+----------+
|Language|TweetCount|
+--------+----------+
|      en|       669|
|      pt|       154|
|      ja|       117|
|      es|        94|
|      tr|        40|
|      ar|        27|
|      fr|        21|
|      ru|        18|
|      th|        14|
|      id|        12|
|      it|        12|
|      nl|         9|
|      de|         9|
|   en-gb|         8|
|      ca|         4|
+--------+----------+
```

```
#Spark
#6.2 Get top 7 languages in terms of volume.
def process(time, rdd):
        print("========= %s =========" % str(time))
        from pyspark.sql.types import NumericType
        try:
            # Get the singleton instance of SparkSession
            spark = getSparkSessionInstance(rdd.context.getConf())

            header = ["ScreenName","FollowersCount","Language","TweetsCount","CountryCode"]
            rowRdd = rdd.map(lambda row : row.split("||"))
            jsonDataFrame = spark.createDataFrame(rowRdd, header)
            jsonDataFrame = jsonDataFrame.withColumn("FollowersCount",jsonDataFrame["FollowersCount"].cast("Int"))
            jsonDataFrame = jsonDataFrame.withColumn("TweetsCount",jsonDataFrame["TweetsCount"].cast("Int"))


            # Creates a temporary view using the DataFrame.
            jsonDataFrame.createOrReplaceTempView("tweets")

            LangDF = \
            spark.sql("select Language,TweetsCount  from tweets")
            #4.Geo based analysis:
            #4.2 Top countries in terms of Tweets Volume
            data = LangDF.groupby('Language').agg({'TweetsCount': 'sum'})
            oldColumns = data.schema.names
            newColumns = ["Language", "TweetVolume"]
            TweetVolumedf = reduce(lambda data, idx: data.withColumnRenamed(oldColumns[idx], newColumns[idx]), xrange(len(oldColu
            TweetVolumedf.createOrReplaceTempView("tweetsvol")
            vol_df = \
            spark.sql("select Language, TweetVolume from tweetsvol order by TweetVolume desc limit 7")
            vol_df.show()
            send_df_to_dashboard(vol_df)
```

```
========= 2017-10-07 18:59:55 =========
+--------+-----------+
|Language|TweetVolume|
+--------+-----------+
|      en|   12455222|
|      ja|    5026451|
|      pt|    3489876|
|      es|    1298686|
|      ar|     457732|
|      fr|     421425|
|      it|     415039|
+--------+-----------+

========= 2017-10-07 19:00:00 =========
+--------+-----------+
|Language|TweetVolume|
+--------+-----------+
|      en|   15804144|
|      ja|    5889889|
|      pt|    4536939|
|      es|    1603492|
|      fr|     846913|
|      ar|     523186|
|      it|     415046|
+--------+-----------+
```

Insights

Maximum number of tweets during the code execution time were being tweeted in English followed by Portuguese, Japanese and other languages such as Arabic and Spanish. Volume of tweets also follows more or less similar kind of Trend.

7. MOST POPULAR TWEETS

7.1. Most popular tweets means here is the tweet which has been re-tweeted maximum number of times.
7.2. Get top 100 most re-tweeted tweets in last 1 hour related to “iPhone”.

```
#Python
import os
import tweepy
from tweepy import OAuthHandler
from tweepy import Stream
from tweepy.streaming import StreamListener
import socket
import json
import requests
import requests_oauthlib

consumer_key = 'SntKtNL6CuaXQdz1Oa7ImyY5D'
consumer_secret = 'DI89nzONJClhLKxoHm9pUAk13aqstKHjTtIvKB0y7z0SHUzm16'
access_token = '911915238173188097-KH2VkrtKmo5UmAjehwxAZFMVDWzouuC'
access_secret = 'CAblWrnbvFLGNCPRDp8uvG3ZyD0GM6sAbCWXvlVItF9xl'

class TweetsListener(StreamListener):

    def __init__(self, csocket, num_tweets_to_grab, retweet_count=10000):
        self.client_socket = csocket
        self.counter = 0
        self.num_tweets_to_grab = num_tweets_to_grab
        self.retweet_count = retweet_count
```

```
class TweetsListener(StreamListener):

    def __init__(self, csocket, num_tweets_to_grab, retweet_count=10000):
        self.client_socket = csocket
        self.counter = 0
        self.num_tweets_to_grab = num_tweets_to_grab
        self.retweet_count = retweet_count

    def on_data(self, data):
        try:
            msg = json.loads(data)
            self.counter += 1
            retweet_count = msg["retweeted_status"]["retweet_count"]
            text_tweet = msg['text'][3:15]
            print(text_tweet)
            if retweet_count >= self.retweet_count:
                #print(msg["text"], retweet_count)
                #self.client_socket.send( msg['text'].encode('utf-8') + '|||' + str(retweet_count) + '\n' )
                 self.client_socket.send( text_tweet + '|||' + str(retweet_count) + '\n' )
            return True

        except BaseException as e:
            print("Error on_data: %s" % str(e))
        return True

    def on_error(self, status):
        print(status)
        return True
```

```
def sendData(c_socket):
    auth = OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_token, access_secret)

    twitter_stream = Stream(auth, TweetsListener(c_socket,num_tweets_to_grab=100, retweet_count=10000))
    #7.1 Most popular tweets means here is the tweet which has been re-tweeted maximum number of times
    twitter_stream.sample()
    #7.2 Get top 100 most re-tweeted tweets in last 1 hour related to "iphone".
    twitter_stream.filter(track="iphone")

if __name__ == "__main__":
    s = socket.socket()     # Create a socket object
    host = "192.168.244.136"      # Get local machine name
    port = 5299             # Reserve a port for your service.
    s.bind((host, port))    # Bind to the port

    print("Listening on port: %s" % str(port))

    s.listen(5)                 # Now wait for client connection.
    c, addr = s.accept()        # Establish connection with client.

    print( "Received request from: " + str( addr ) )

    sendData( c )
```

```
Listening on port: 5299
Received request from: ('192.168.244.136', 57138)
```

```
#Spark Code
words = lines.map(lambda row : row.split("|||"))
# Convert RDDs of the words DStream to DataFrame and run SQL query
def process(time, rdd):
        print("========= %s =========" % str(time))
        from pyspark.sql.types import NumericType

        try:
            header = ["Tweet","RetweetCount"]
            spark = getSparkSessionInstance(rdd.context.getConf())
            jsonDataFrame = spark.createDataFrame(rdd, header)
            jsonDataFrame = jsonDataFrame.withColumn("RetweetCount",jsonDataFrame["RetweetCount"].cast("Int"))
            jsonDataFrame.createOrReplaceTempView("tweets")
            TrendTweetDF = \
            spark.sql("select Tweet, RetweetCount from tweets order by RetweetCount desc")
            TrendTweetDF.show()
            send_df_to_dashboard(TrendTweetDF)

        except:
            pass
def send_df_to_dashboard(df):
    # extract the hashtags from dataframe and convert them into array
    top_tags = [str(t.Tweet) for t in df.select("Tweet").collect()]
    # extract the counts from dataframe and convert them into array
    tags_count = [p.RetweetCount for p in df.select("RetweetCount").collect()]
    # initialize and send the data through REST API
    url = 'http://192.168.244.136:5001/updateData'
    request_data = {'label': str(top_tags), 'data': str(tags_count)}
    response = requests.post(url, data=request_data)
words.foreachRDD(process)
ssc.start()
ssc.awaitTermination()
```

7.1 Most popular tweets means here is the tweet which has been re-tweeted maximum number of times.

```
========= 2017-10-07 22:22:30 =========
+------------+------------+
|       Tweet|RetweetCount|
+------------+------------+
|@1800SADDAD:|       47179|
|@sac480: Uni|       44257|
|@EpicSexyFac|       30911|
|@Panama_1703|       23078|
|@DanielPeach|       18548|
|@eemoneee: L|       15985|
|@hoesuueee: |       14899|
|@StreetFashi|       13724|
|@aalgarviaa:|       11166|
|@rafael_bati|       11047|
+------------+------------+

========= 2017-10-07 22:22:35 =========
+------------+------------+
|       Tweet|RetweetCount|
+------------+------------+
|@1800SADDAD:|       47179|
|@sac480: Uni|       44257|
|@EpicSexyFac|       30911|
|@Panama_1703|       23078|
|@eemoneee: L|       15985|
|@StreetFashi|       13724|
|@aalgarviaa:|       11166|
|@rafael_bati|       11047|
+------------+------------+
```

7.2 Get top 100 most re-tweeted tweets in last 1 hour related to “iphone”

```
========= 2017-10-07 22:51:55 =========
+------------+------------+
|       Tweet|RetweetCount|
+------------+------------+
|@kennagq: Do|      125008|
|@KelseyNews3|       73923|
|@_jxzminemxr|       72103|
|@faddelg7: B|       60353|
|@edbankzz: t|       48659|
|@edbankzz: t|       48644|
|@finah: ever|       36327|
|@MeLlamoTrev|       29544|
|@MeLlamoTrev|       29540|
|@LILAFRIMANE|       28360|
|@zxxchh: no |       27510|
|@Proyecto4Pa|       22802|
|@finah: i ju|       21002|
|@LebaenesePa|       19195|
|@LebaenesePa|       19193|
|@ejdickson: |       16570|
|@liliaaannee|       14786|
|@liliaaannee|       14786|
|@JalenSkutt:|       14488|
|@NecatiAkcay|       11444|
+------------+------------+
```

Insights

The above trend shows the retweet counts of various messages across the world and also the number of retweets on iPhone during the execution time of this code. These observations would provide multitude of insights when the complete tweet text that is retweeted maximum number of times is analyzed in detail.

8. PLOT ALL ABOVE ANALYSIS

This is a real time dashboard and it gets updated according to data streaming in Spark during real time. Flask package in python, Chart.js and html coding is used to create these dashboards.

```
#Dashboard
from flask import Flask,jsonify,request
from flask import render_template
import ast
app = Flask(__name__)
labels = []
values = []
@app.route("/")
def get_chart_page():
    global labels,values
    labels = []
    values = []
    return render_template('chart.html', values=values, labels=labels)
@app.route('/refreshData')
def refresh_graph_data():
    global labels, values
    print("labels now: " + str(labels))
    print("data now: " + str(values))
    return jsonify(sLabel=labels, sData=values)
@app.route('/updateData', methods=['POST'])
def update_data():
    global labels, values
    if not request.form or 'data' not in request.form:
        return "error",400
    labels = ast.literal_eval(request.form['label'])
    values = ast.literal_eval(request.form['data'])
    print("labels received: " + str(labels))
    print("data received: " + str(values))
    return "success",201
if __name__ == "__main__":
    app.run(host='192.168.244.136', port=5001)
```

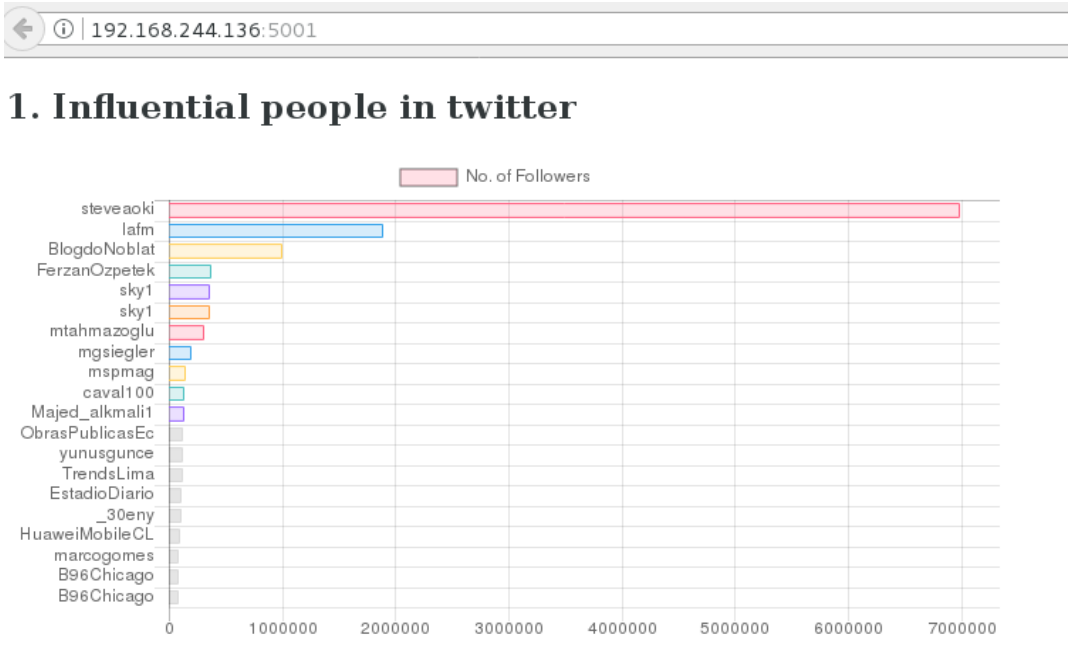
```
 * Running on http://192.168.244.136:5001/ (Press CTRL+C to quit)
192.168.244.136 - - [07/Oct/2017 01:45:44] "POST /updateData HTTP/1.1" 201 -
```

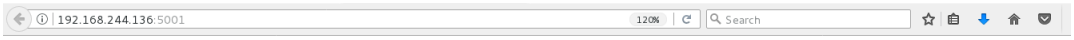
```
# HTML
<!DOCTYPE html>
<html>
      <head>
          <meta charset="utf-8"/>
          <title>1. Influential people in twitter</title>
          <script src='static/Chart.js'></script>
          <script src="//ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"></script>

      </head>
      <body>
              <h2>1. Influential people in twitter</h2>
              <div style="width:700px;height=500px">
                  <canvas id="chart"></canvas>
              </div>
      </body>
<script>
   var ctx = document.getElementById("chart");
   var myChart = new Chart(ctx, {
          type: 'horizontalBar',
          data: {
              labels: [{% for item in labels %}
                        "{{item}}",
                       {% endfor %}],
              datasets: [{
                  label: 'No. of Followers',
                  data: [{% for item in values %}
                             {{item}},
                           {% endfor %}],
```

```
                  {% endfor %}],
              backgroundColor: [
                  'rgba(255, 99, 132, 0.2)',
                  'rgba(54, 162, 235, 0.2)',
                  'rgba(255, 206, 86, 0.2)',
                  'rgba(75, 192, 192, 0.2)',
                  'rgba(153, 102, 255, 0.2)',
                  'rgba(255, 159, 64, 0.2)',
                  'rgba(255, 99, 132, 0.2)',
                  'rgba(54, 162, 235, 0.2)',
                  'rgba(255, 206, 86, 0.2)',
                  'rgba(75, 192, 192, 0.2)',
                  'rgba(153, 102, 255, 0.2)'
              ],
              borderColor: [
                  'rgba(255,99,132,1)',
                  'rgba(54, 162, 235, 1)',
                  'rgba(255, 206, 86, 1)',
                  'rgba(75, 192, 192, 1)',
                  'rgba(153, 102, 255, 1)',
                  'rgba(255, 159, 64, 1)',
                  'rgba(255,99,132,1)',
                  'rgba(54, 162, 235, 1)',
                  'rgba(255, 206, 86, 1)',
                  'rgba(75, 192, 192, 1)',
                  'rgba(153, 102, 255, 1)'
              ],
              borderWidth: 1
          }]
      },
```

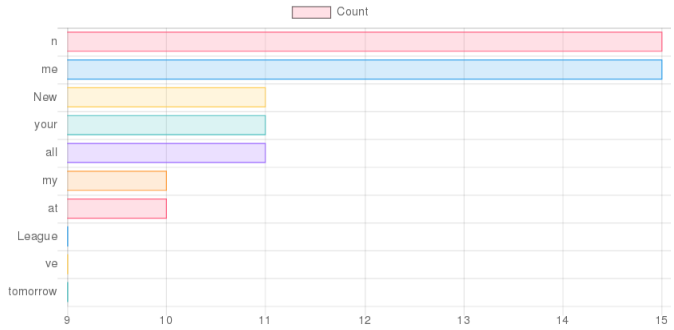
```
                ],
                borderWidth: 1
            }]
        },
        options: {
            scales: {
                yAxes: [{
                    ticks: {
                        beginAtZero:true
                    }
                }]
            }
        }
    });
    var src_Labels = [];
    var src_Data = [];
    setInterval(function(){
            $.getJSON('/refreshData', {
            }, function(data) {
                src_Labels = data.sLabel;
                src_Data = data.sData;
            });
            myChart.data.labels = src_Labels;
            myChart.data.datasets[0].data = src_Data;
            myChart.update();
    },1000);
</script>
</html>
```

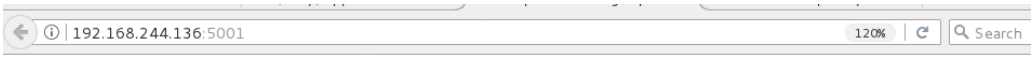
Snapshots of Real time Dashboard images:



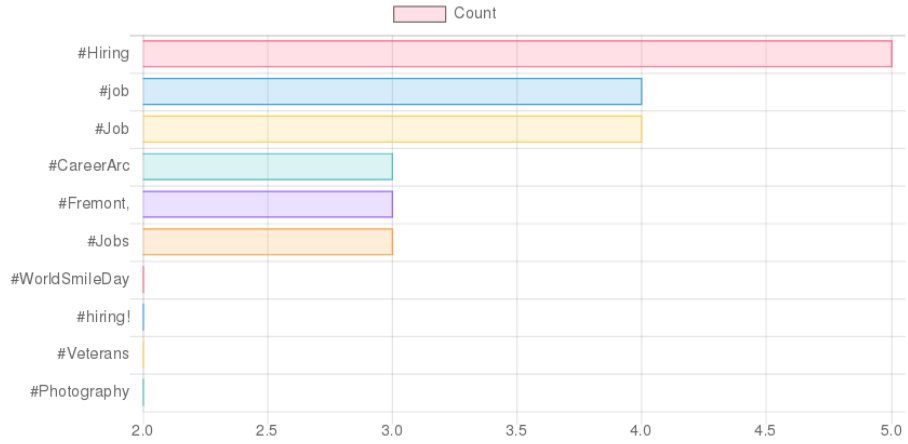


2.1 word from the tweet which is occurring most of the times in the whole tweets corpus

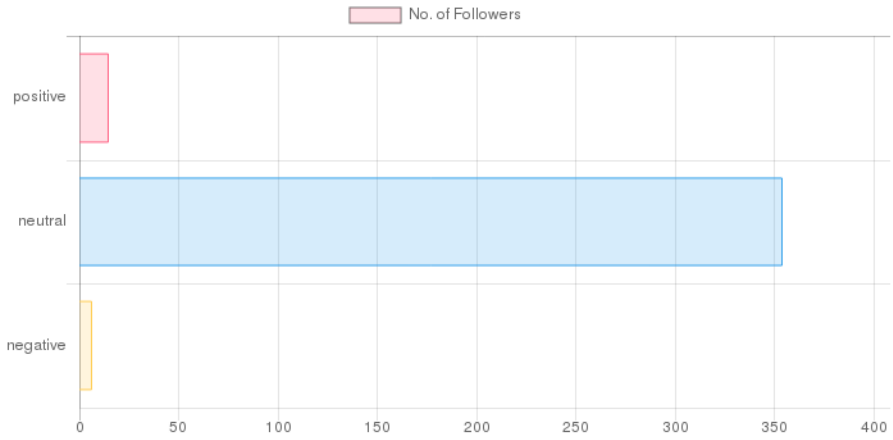




2.2 Top 10 trending topic

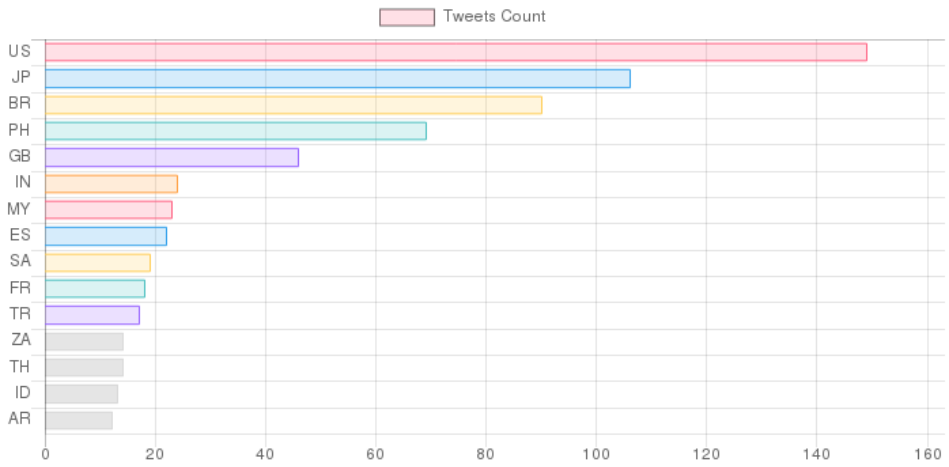


3. Sentiment Analysis - iPhone

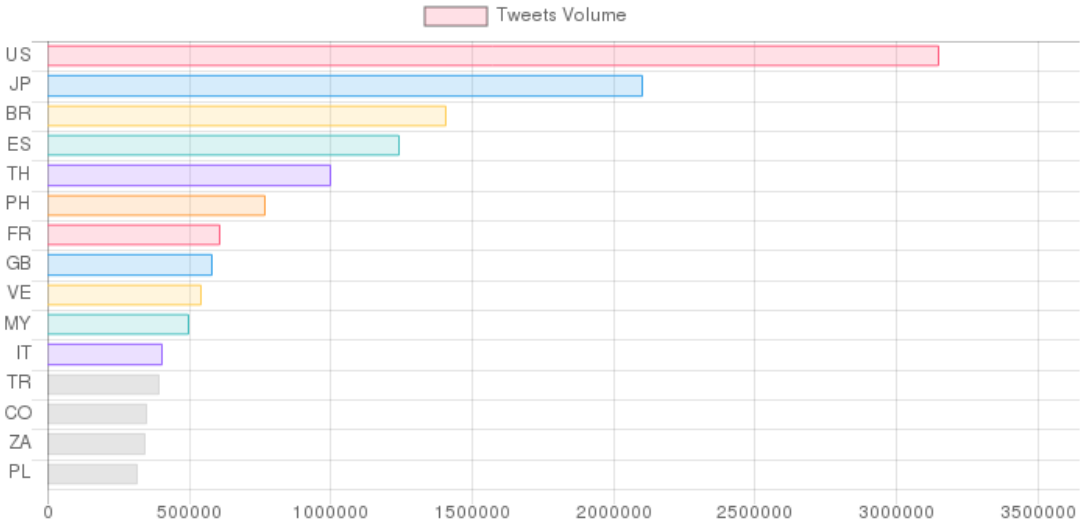


192.168.244.136:5001

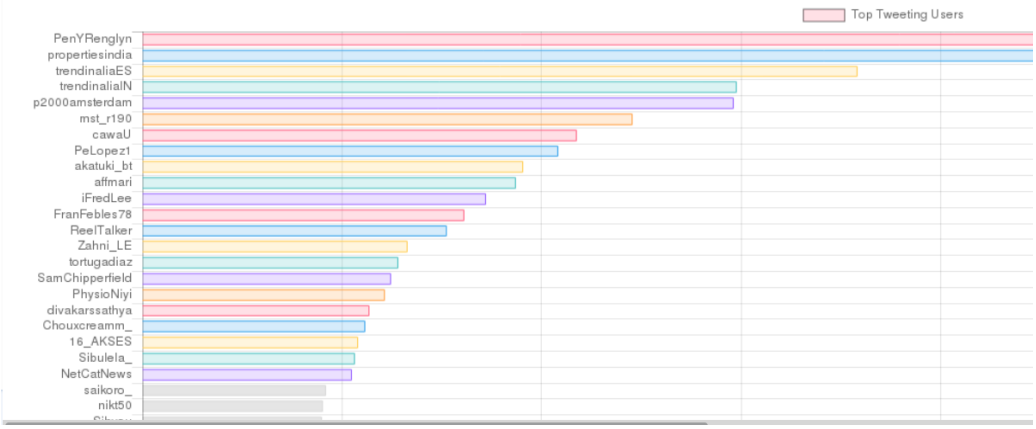
4.1 & 4.3. List of top 15 countries in terms of tweet counts:

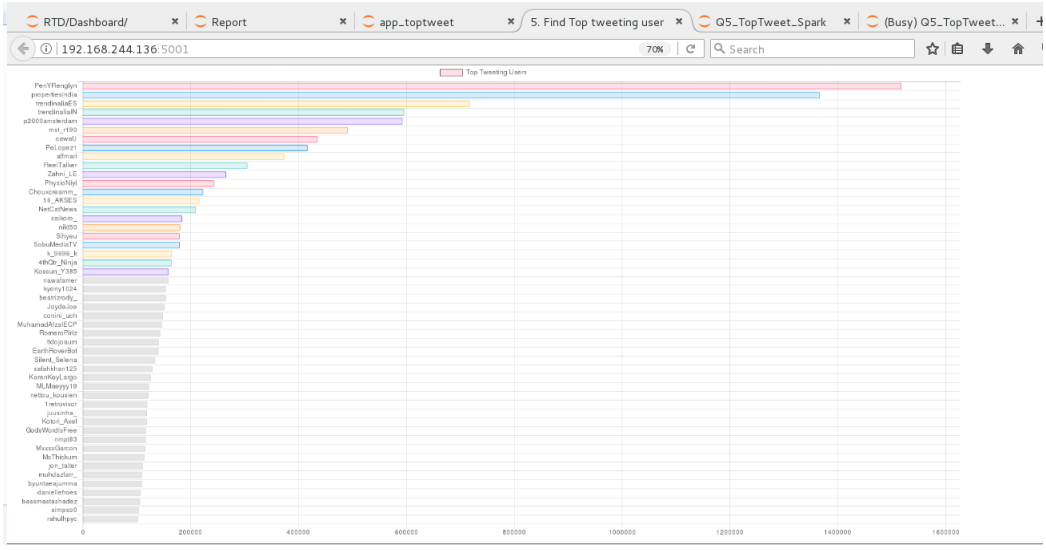


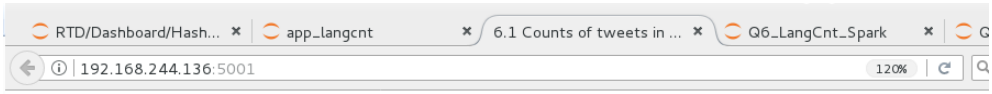
4.2 Top countries in terms of Tweets Volume:



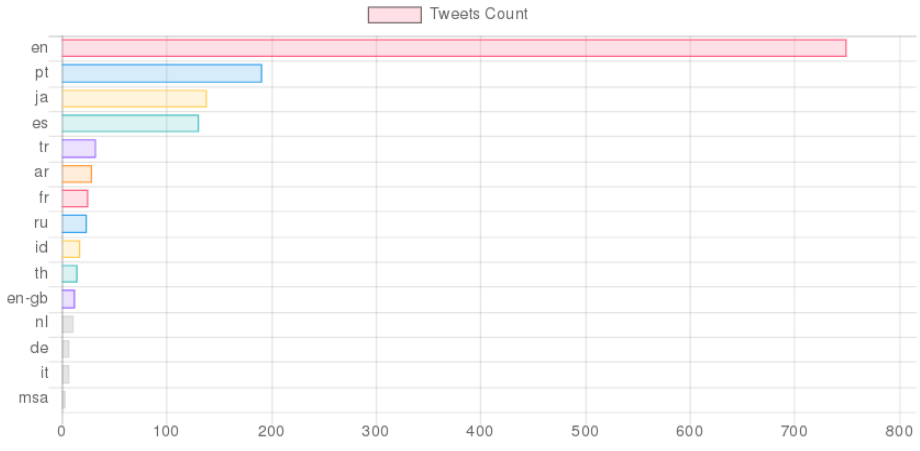
5. Find Top tweeting user



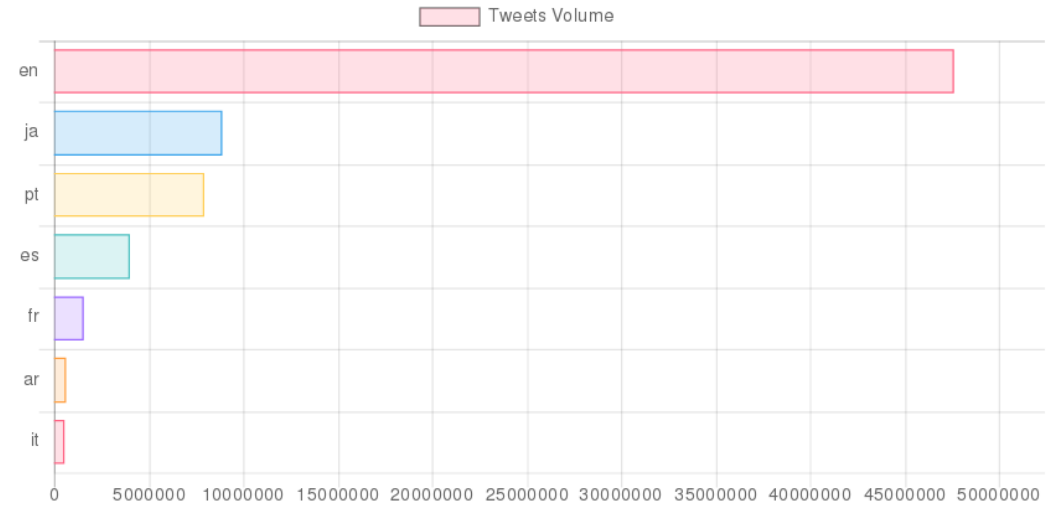




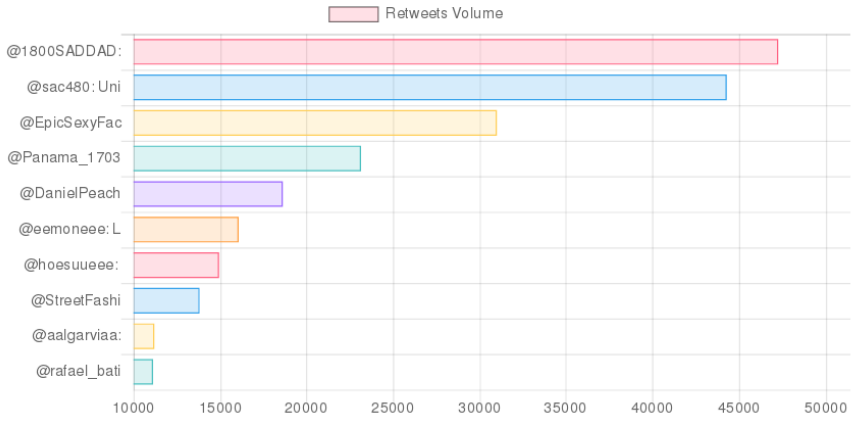
6.1 Counts of tweets in different languages:



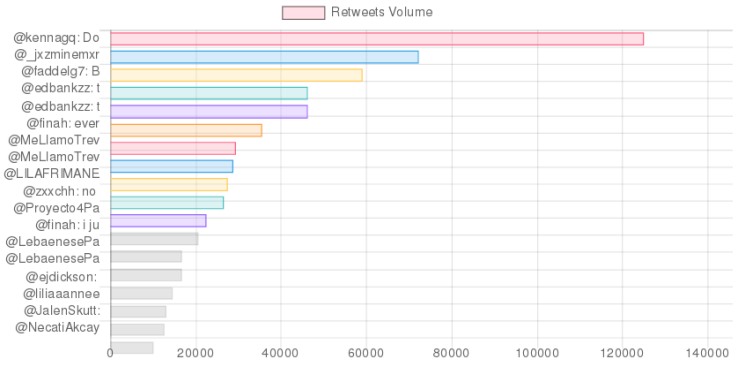
6.2 Top 7 languages in terms of volume:



7.1 Most popular tweets



7.2 Get top 100 most re-tweeted tweets in last 1 hour related to “iphone”



Insights

The above dashboard is linked to Spark code and needs to be executed along with Spark and Python programs. This dashboard shows the same insights as we discussed in the Insights section of questions 1 to 7.

STEPS TO EXECUTE CODE

For each question, Code needs to be executed in the following order:

Unzip the file contents:

1. Keep the folder structure intact
2. Change Ip address, port number and twitter credentials in the code
3. Execute Q<No_Name.>_Python.ipynb
4. Execute Q< No_Name.>_Spark.ipynb
5. Execute Q< No_Name.>_App.ipynb
6. For questions 4 and 6, there will be 1 python script and 2 spark scripts. Execute the same python for both spark scripts.