**BIG DATA MANAGEMENT – ASSIGNMENT 3**

**Working With Speed Layer**

**PRONOY D’SA – 2952742**

**PROJECT REPORT – Individual Summary**

**Introduction:**

In the following BDM Assignment, we are required to establish an infrastructure that streams Social Media data like twitter data and stores it in a NoSQL database such as MongoDB. We are required to parse and process the streaming data stored in the text file which is in json format and this twitter data is also stored simultaneously in MongoDB. We are also able to perform some simple analytics such as text mining on this data and provide some results in the form of valuable information.

**Procedure:**

These are the steps which we are going to carry out:

1. TweetRead.py - Extract twitter data using tweepy and learn how to handle it using pandas storing it in MongoDb using pymongo.
2. TweetAnalysis.py - Do some basic statistics and visualizations with re, matplotlib and display visualizations with iPython.

A) Opening TweetRead.py on Spyder IDE: We make sure we have the Anaconda package installed which comes with Python environment, its Packages, Spyder IDE, Anaconda’s prompt. Using Spyder we open our TweetRead.py file.

B) Running MongoDB (Mongod.exe) in Background: After installing MongoDB and setting up its data directory, open cmd and type the below command to start the service.

"C:\Program Files\MongoDB\Server\3.6\bin\mongod.exe"

C) Now open Anaconda Prompt and make sure you ‘cd’ into the TweetRead.py directory, type the below command:

python TweetRead.py > twitter\_data.txt

It is used to save the recently streamed twitter data to the twitter\_txt file corresponding to the relevant twitter tags which simultaneously is stored in MongoDB after opening up a connection through the program.

D) We can stop the streaming process by clicking Ctrl+C and we can later view the streamed data gathered in the twitter\_data.txt text file, the same data is stored in mongodb.

E) We later performed simple text mining analysis on the tweets collected using our TweetAnalysis.py program which analyzed the tweets according to the properties of the tweets such language, country/location, and the relevancy of the tags with regards to each other.

1.Extracting twitter data:

To extract tweets for a posterior analysis, we need to access to our Twitter account and create an app. The website to do this is https://apps.twitter.com/. After creating an app in twitter, it will provide us with the following codes:

Consumer Key (API Key)

Consumer Secret (API Secret)

Access Token

Access Token Secret

from pymongo import MongoClient

import json

#Import the necessary methods from tweepy library

from tweepy.streaming import StreamListener

from tweepy import OAuthHandler

from tweepy import Stream

#Variables that contains the user credentials to access Twitter API

access\_token = "993861826910785536-SRwlwm6u1QnlMB8efcPYWURlniZi3o6"

access\_token\_secret = "YaxmO5FV7cVaLtb5xxxBMGtRr3HECFQSE7rAPrSwY3DSO"

consumer\_key = "YGxg6Pv3b5DNIB4O9vIxhO8it"

consumer\_secret = "dzqEyOncIN3TQ2ViOooPpEdKMxUz90Ub368RJw15vUXiIQamdi"

#This is a basic listener that just prints received tweets to stdout.

class StdOutListener(StreamListener):

def on\_data(self, data):

client = MongoClient('localhost', 27017)

db = client['twitter\_db']

collection = db['twitter\_collection']

tweet = json.loads(data)

collection.insert(tweet)

print(data)

return True

def on\_error(self, status):

print(status)

if \_\_name\_\_ == '\_\_main\_\_':

The Below code establishes a connection with the twitter and authenticates the twitter account with which we have created the twitter app using the consumer\_key, consumer\_secret, access\_token, access\_token\_secret.

#This handles Twitter authentication and the connection to Twitter Streaming API

l = StdOutListener()

auth = OAuthHandler(consumer\_key, consumer\_secret)

auth.set\_access\_token(access\_token, access\_token\_secret)

stream = Stream(auth, l)

#This line filter Twitter Streams to capture data by the keywords: 'trump', 'modi', 'putin'

stream.filter(track=['trump', 'modi', 'putin'])

2. Visualization and basic statistics

We are using pandas and matplotlib to do some analysis on the collected tweets and display the results in the form of graphs.

import json

import pandas as pd

import matplotlib.pyplot as plt

import re

#preprocessing and parsing of the text is done here,all the text is converted into lowercase letters

def word\_in\_text(word, text):

word = word.lower()

text = text.lower()

match = re.search(word, text)

if match:

return True

return False

#this function allows to extract all the links in the given file

def extract\_link(text):

regex = r'https?://[^\s<>"]+|www\.[^\s<>"]+'

match = re.search(regex, text)

if match:

return match.group()

return ''

def main():

# Reading Tweets

print('Reading Tweets\n')

tweets\_data\_path = "C:/Users/prono/.spyder-py3/twitter\_data.txt"

tweets\_data = []

tweets\_file = open(tweets\_data\_path, "r")

for line in tweets\_file:

try:

tweet = json.loads(line)

tweets\_data.append(tweet)

except:

continue

print(len(tweets\_data))

# Structuring Tweets

print('Structuring Tweets\n')

tweets = pd.DataFrame()

tweets['text'] = list([tweet.get('text','') for tweet in tweets\_data])

tweets['lang'] = list([tweet.get('lang','') for tweet in tweets\_data])

tweets['country'] = list([tweet.get('place','').get('country','') if tweet.get('place') != None else None for tweet in tweets\_data])

# Analyzing Tweets by Language

print('Analyzing tweets by language\n')

tweets\_by\_lang = tweets['lang'].value\_counts()

fig, ax = plt.subplots()

ax.tick\_params(axis='x', labelsize=15)

ax.tick\_params(axis='y', labelsize=10)

ax.set\_xlabel('Languages', fontsize=15)

ax.set\_ylabel('Number of tweets', fontsize=15)

ax.set\_title('Top 5 languages', fontsize=15, fontweight='bold')

tweets\_by\_lang[:5].plot(ax=ax, kind='bar', color='red')

plt.savefig('tweet\_by\_lang', format='png')

# Analyzing Tweets by Country

print('Analyzing tweets by country\n')

tweets\_by\_country = tweets['country'].value\_counts()

fig, ax = plt.subplots()

ax.tick\_params(axis='x', labelsize=15)

ax.tick\_params(axis='y', labelsize=10)

ax.set\_xlabel('Countries', fontsize=15)

ax.set\_ylabel('Number of tweets', fontsize=15)

ax.set\_title('Top 5 countries', fontsize=15, fontweight='bold')

tweets\_by\_country[:5].plot(ax=ax, kind='bar', color='blue')

plt.savefig('tweet\_by\_country', format='png')

# Adding president columns to the tweets DataFrame

print('Adding president tags to the data\n')

tweets['trump'] = tweets['text'].apply(lambda tweet: word\_in\_text('trump', tweet))

tweets['modi'] = tweets['text'].apply(lambda tweet: word\_in\_text('modi', tweet))

tweets['putin'] = tweets['text'].apply(lambda tweet: word\_in\_text('putin', tweet))

# Analyzing Tweets by president: First attempt

print('Analyzing tweets by president: First attempt\n')

prg\_langs = ['trump', 'modi', 'putin']

tweets\_by\_prg\_lang = [tweets['trump'].value\_counts()[True], tweets['modi'].value\_counts()[True],

tweets['putin'].value\_counts()[True]]

x\_pos = list(range(len(prg\_langs)))

width = 0.8

fig, ax = plt.subplots()

plt.bar(x\_pos, tweets\_by\_prg\_lang, width, alpha=1, color='g')

ax.set\_ylabel('Number of tweets', fontsize=15)

ax.set\_title('Ranking: trump vs. modi vs. putin (Raw data)', fontsize=10, fontweight='bold')

ax.set\_xticks([p + 0.4 \* width for p in x\_pos])

ax.set\_xticklabels(prg\_langs)

plt.grid()

plt.savefig('tweet\_by\_president\_1', format='png')

# Targeting relevant tweets

print('Targeting relevant tweets\n')

tweets['president'] = tweets['text'].apply(lambda tweet: word\_in\_text('president', tweet))

tweets['best'] = tweets['text'].apply(lambda tweet: word\_in\_text('best', tweet))

tweets['relevant'] = tweets['text'].apply(

lambda tweet: word\_in\_text('president', tweet) or word\_in\_text('best', tweet))

# Analyzing Tweets by president: Second attempt

print('Analyzing tweets by president: First attempt\n')

import IPython

IPython.embed()

def get\_value\_counts(tweets, language):

try:

return tweets[tweets['relevant'] == True][language].value\_counts()[True]

except KeyError:

return 0

tweets\_by\_prg\_lang = [get\_value\_counts(tweets, 'trump'),

get\_value\_counts(tweets, 'modi'),

get\_value\_counts(tweets, 'putin')]

x\_pos = list(range(len(prg\_langs)))

width = 0.8

fig, ax = plt.subplots()

plt.bar(x\_pos, tweets\_by\_prg\_lang, width, alpha=1, color='g')

ax.set\_ylabel('Number of tweets', fontsize=15)

ax.set\_title('Ranking: trump vs. modi vs. putin (Relevant data)', fontsize=10, fontweight='bold')

ax.set\_xticks([p + 0.4 \* width for p in x\_pos])

ax.set\_xticklabels(prg\_langs)

plt.grid()

plt.savefig('tweet\_by\_president\_2', format='png')

# Extracting Links

tweets['link'] = tweets['text'].apply(lambda tweet: extract\_link(tweet))

tweets\_relevant = tweets[tweets['relevant'] == True]

tweets\_relevant\_with\_link = tweets\_relevant[tweets\_relevant['link'] != '']

print('\n Below are some Python links that we extracted\n')

print(tweets\_relevant\_with\_link[tweets\_relevant\_with\_link['trump'] == True]['link'].head())

if \_\_name\_\_ == '\_\_main\_\_':

main()