#=====================================================================================================================================

# PREPROCESSING

#=====================================================================================================================================

# Flask for python server and Cross Origin Resource Sharing (CORS)

from flask import Flask

from flask\_cors import CORS

# Tweepy - Python's official Twitter API to extract tweet information & user details

import tweepy

# NewsAPI - Official newsAPI of 100s of news channel from legitimate sources

from newsapi.newsapi\_client import NewsApiClient

# GNnews - Google Search API for verifying from legitimate sources

import json

import requests

# Text Analysis,Cleaning & Ntural Language Processing

import re

import emoji

import string

from rake\_nltk import Rake

from textblob import TextBlob

from nltk.corpus import stopwords

from nltk.corpus import wordnet as wn

from nltk.tokenize import word\_tokenize

# Machine Learning

import pandas as pd

# ================================================= SETUP FOR TWEEPY ===============================================================

consumer\_key = "HeN41iSC2AmyFBOW6vX8sKTUs"

consumer\_secret = "MZjHSXATBkLnfZwSI5aoaGN3zCSSh3aCzcKjri3OoWuRggifpO"

access\_key = "1010555953647902720-RYonQIwDZ5VmhepIvIrka4fTWzKt5K"

access\_secret = "cEN42A0S5f7u8JVW6fS1oXyLfut44rOqXIXplM2oiEmvn"

# Authorization to consumer key and consumer secret

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

# Access to user's access key and access secret

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

# Calling api

api = tweepy.API(auth)

# ============================================ TEXT ANALYSIS FOR NEWS API ==========================================================

# Count & REMOVE EMOJIS from text

emoji\_count = 0 # Used for checking formality of content later

def give\_emoji\_free\_text(text):

allchars = [str for str in text]

emoji\_list = [c for c in allchars if c in emoji.UNICODE\_EMOJI]

e\_count = len(emoji\_list)

clean\_text = ' '.join([str for str in text.split() if not any(i in str for i in emoji\_list)])

return (clean\_text,e\_count)

def newsSentiment(mylist,tweet\_senti):

if len(mylist) < 1:

return "f" # no related news found , return FAKE

all\_senti = list(map(getSentiment,[news for news in mylist]))

if all\_senti.count(tweet\_senti)/len(all\_senti) >= 0.6 :

return "hc" # if all or more than 60% news have the same sentiment as that of tweet

def getval(num):

if num > 0 :

return 1

elif num < 0 :

return -1

return 0

news\_senti = getval(sum(all\_senti))

# above line adds up news sentiments of each article in message\_list to find dominating sentiment

if news\_senti == tweet\_senti :

return "c" # most of the news support the tweet so SEEMS CREDIBLE

else:

return "sf" # most of the news doesn't support the tweet so SEEMS FAKE

def getNewsLabel(NLabel,GLabel):

if GLabel == NLabel : # If both agree on same Label

return GLabel

elif NLabel == "f": # ML is confident that news is FAKE

return "sf" # No matter if newsAPI says even HC, credibiliy can't be promised

# RESULT : Seems Fake

elif NLabel == "sf" or NLabel == "c":

# ML is neither confident that news is FAKE nor HIGHLY CREDIBLE

if GLabel == "hc": # If newsAPI says HC, then might be credible to some extent

return "c" # RESULT : Credible

else : # If newsAPI says less credible or seems fake

if NLabel == "c":

return "c"

return "sf" # RESULT : Seems Fake

else: # ML is confident that news is HIGHLY CREDIBLE no doubt

return "c"

# =========================================== TEXT ANALYSIS OF TWEET CONTENT =======================================================

def isFormal(sentence):

err=0

if emoji\_count > 2: # more than 2 emoji count means less formal

err += emoji\_count - 2

w = ['think','believe','feel'] ; c=0

wh = ['what','when','where','which','who','whom','why','how']

puncts = [1 for s in sentence if s in string.punctuation]

whitesp = [1 for s in sentence if s in string.whitespace]

totchars = len(sentence)-len(puncts)-len(whitesp)

if totchars > 70 : # size limited to 70 characters is a concise news

err+=1

firstword = sentence.split()[0] # Most Official news begins with Subject / Noun first

for x in wn.synsets(firstword):

if x.pos() == 'n':

c+=1

else:

c-=1

if c <= 0 :

err+=1

if firstword in wh: # Official news may contain but, never begins with a WH-Question

err+=1

if sentence[0] != sentence.capitalize()[0]: # First letter of first word should be capital

err+=1

err += sentence.count("i ")+ sentence.count(" i ")-3 # Not more than 3 times personal pronoun usage

err += sentence.count("we ")+ sentence.count(" we ")-3

err += sentence.count("my ")+ sentence.count("our ")-3

y = [sentence.count(word) for word in w if word in sentence]

if 'I' in sentence and sum(y): # Avoid verbs like 'think', believe', 'feel' with "I"

err+=sum(y)

if sentence.count("!") > 2 : # atmost 2 or no exclamations & exaggerations

err+=sentence.count("!")-2

if err > 4: # Temporary threshold

return False

else:

return True

def getSentiment(text):

t = TextBlob(text)

if t.sentiment.polarity > 0:

return 1

elif t.sentiment.polarity < 0 :

return -1

else:

return 0

def detectURL(t):

url = re.findall('https?://t\.co/\S+',t)

if (len(url) >0):

return 1

else:

return 0

# ============================================ TRAINING OF MACHINE LEARNING MODEL =================================================

# Importing the dataset

dataset = pd.read\_csv('TweetSet\_100.csv')

X = dataset.iloc[:,3:10] # remove .values to open the object in variable explorer

y = dataset.iloc[:, 10]

# Splitting the dataset into the Training set and Test set

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.20, random\_state = 0)

# Feature Scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X\_train = sc.fit\_transform(X\_train)

X\_test = sc.transform(X\_test)

# Fitting Naive Bayes to the Training set ACCURACY : 17 correct , 3 incorrect

from sklearn.naive\_bayes import GaussianNB

classifier = GaussianNB()

classifier.fit(X\_train, y\_train)

# Predicting the Test set results

y\_pred = classifier.predict(X\_test)

# Making the Confusion Matrix

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test, y\_pred)

def getFinalLabel(newsLabel,mlLabel):

d\_nums = {"f":0, "sf":1, "c":2, "hc":3}

l = {"f":["f","sf","sf","c"],"sf":["sf","sf","sf","c"],"c":["sf","sf","c","c"],"hc":["sf","c","c","hc"]}

d = {"f":"is Fake", "sf":"seems Fake", "c":"seems Credible", "hc":"is Highly Credible"}

return d[l[mlLabel][d\_nums[newsLabel]]]

#=====================================================================================================================================

# REAL TIME IMPLEMENTATION

#=====================================================================================================================================

# ============================================ SETTING UP FLASK SERVER ===============================================================

app = Flask(\_\_name\_\_)

CORS(app)

@app.route('/tweet/<id>')

def tweet(id):

# ============================= EXTRACTION OF TWEET CONTENT & PARAMETERS USING TWEEPY ================================================

id\_of\_tweet = id # Put Tweet\_ID here

status = api.get\_status(id\_of\_tweet, tweet\_mode = 'extended')

# if no tweet\_mode is specified, gives truncated tweets upto 140 chars only

# Parameters to pass to ML model for analysis

verified = status.user.verified # Account verified or not

followers = status.user.followers\_count # No. of Followers

likes = status.favorite\_count # Number of likes

RT = status.retweet\_count # Number of Retweets

tweettext = status.full\_text # status.text gives truncated tweet wihout tweet\_mode

# =============================== TWEET CLEANING FOR GENERATING QUERY FOR GNEWS API ==================================================

(text,emoji\_count) = give\_emoji\_free\_text(tweettext)

# BASIC CLEANING TEXT

text2 = re.sub('[^a-zA-Z]',' ',text)

text2 = text2.lower()

# TOKENIZE into words

word\_tokens = word\_tokenize(text2)

# REMOVE STOP WORDS like a,an,the,etc....

stop\_words = set(stopwords.words('english'))

content2 = [w for w in word\_tokens if not w in stop\_words]

# Find relevant keywords & phrases using RAKE (Rapid Automatic Keyword Extraction)

r = Rake() # uses standard stopwards of nltk by default or pass stopwordlist file

r.extract\_keywords\_from\_text(" ".join(content2))

phrases = r.get\_ranked\_phrases()

query = phrases[0].split() # The 1st ranked phrase is most relevant & enough for now

if len(query) > 8: # limited keywords avoid empty results returned

query = query[:8]

query = ' '.join(query)

# ================ =======================ANALYSIS OF TWEET CONTENT FOR ML MODEL ====================================================

formal = isFormal(text)

sentiment = getSentiment(text)

URL = detectURL(text)

# ===============================NewsAPI & GNEWS API FOR RETRIVAL & GET CLASSIFICATION ==================================================

#NewsAPI

newsapi = NewsApiClient(api\_key='97220b4495bb442688ef7595889118eb')

data = newsapi.get\_everything(q=query,language='en')

articles\_array = data['articles']

message\_list=[]

for article in articles\_array:

message\_list.append(article['description'])

# GNews

url="https://gnews.io/api/v2/"

token="c7e821d7d236323b18350a160ee68a83"

response = requests.get(url+'?q='+query+'&token='+token) # call GNews using query

results = json.loads(response.text) # returns atmost 10 best results

NLabel = newsSentiment(message\_list,sentiment)

GLabel = newsSentiment([x['title'] for x in results['articles']],sentiment)

newsLabel = getNewsLabel(NLabel,GLabel) # Pass this to final color decider

# ============================ ==============MACHINE LEARNING - NAIVE BAYES =========================================================

# Running the model with the input from extension :

m=["f","sf","c","hc"]

attribs = ['verified','followers','formal','sentiment','likes','RT','URL']

x\_custom\_input = pd.DataFrame([[verified, followers, formal, sentiment, likes, RT, URL]], columns=attribs)

x\_custom\_input = sc.transform(x\_custom\_input)

y\_custom\_input = classifier.predict(x\_custom\_input)

mlLabel = m[list(y\_custom\_input)[0]] # class label to pass to final label decider code

# ========================================== FINAL LABEL & COLOR DECISION ==========================================================

return "This tweet "+getFinalLabel(newsLabel,mlLabel)

# ============================================ RUN THE FLASK PYTHON SERVER ===========================================================

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

app.run()