**Program:**

|  |
| --- |
| import tweepy |
|  | from tweepy import API |
|  | from tweepy import Cursor |
|  | from tweepy import OAuthHandler |
|  |  |
|  | from textblob import TextBlob |
|  | from wordcloud import WordCloud |
|  | import numpy as np |
|  | import pandas as pd |
|  | import matplotlib.pyplot as plt |
|  |  |
|  | plt.style.use('fivethirtyeight') |
|  |  |
|  | import re |
|  |  |
|  | from sklearn.linear\_model import SGDClassifier |
|  | from sklearn.pipeline import Pipeline |
|  | from sklearn.feature\_extraction.text import CountVectorizer |
|  | from sklearn.feature\_extraction.text import TfidfTransformer |
|  | from sklearn.model\_selection import train\_test\_split |
|  | from sklearn.metrics import classification\_report, accuracy\_score, confusion\_matrix |
|  |  |
|  |  |
|  | # # --- AUTHENTICATION --- # # |
|  | CONSUMER\_KEY = "HtpQB7yavzzcAZuNTogKQLiBK" |
|  | CONSUMER\_SECRET = "NnMLsXpvH1KOGCNvApnBEZm6dV9cueqErlk5XN8P8ZFimTFXnH" |
|  | ACCESS\_TOKEN = "1082924245217411072-9dve9pDQOgFgTWeS9BDM4a779CV6H6" |
|  | ACCESS\_TOKEN\_SECRET = "jXIAkDiYIpchTVJW5iTULu1eza6XDeAnA0GUw1tkLPqnx" |
|  |  |
|  |  |
|  | # # --- TWITTER AUTHENTICATOR - Authentication Twitter credencials --- # # |
|  | class TwitterAuthenticator(): |
|  |  |
|  | # Get the autenthication keys and authenticate in Twitter API. |
|  | def authenticate\_twitter\_app(): |
|  | auth = OAuthHandler(CONSUMER\_KEY, CONSUMER\_SECRET) |
|  | auth.set\_access\_token(ACCESS\_TOKEN, ACCESS\_TOKEN\_SECRET) |
|  | api = tweepy.API(auth, wait\_on\_rate\_limit=True) |
|  | return api |
|  |  |
|  |  |
|  | # # --- TWEET ANALYZER - Funcionality for analyzing and categorizing content from tweets --- # # |
|  | class TweetAnalyzer(): |
|  |  |
|  | # Clean the tweet texs, removing unnecessary characters. |
|  | def clean\_tweet(self, tweet): |
|  | return ' '.join(re.sub("RT (@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)", " ", tweet).split()) |
|  |  |
|  | # Create the DataFrame with the searched tweets . |
|  | def tweets\_to\_data\_frame(self): |
|  | tweet\_df=pd.DataFrame() |
|  | api = TwitterAuthenticator.authenticate\_twitter\_app() |
|  | for hashtag in hash\_tag\_list: |
|  | tweets = tweepy.Cursor(api.search, |
|  | q = hashtag, |
|  | lang = "en", |
|  | since = date\_since).items(100) |
|  |  |
|  | data = pd.DataFrame(data=[tweet.text for tweet in tweets], columns = ["text"]) |
|  | data["hashtag"] = np.array([hashtag for tweet in data["text"]]) |
|  | data["text"] = data["text"].apply(lambda x: self.clean\_tweet(x)) |
|  |  |
|  | tweet\_df=pd.concat([tweet\_df, data]) |
|  | return tweet\_df |
|  |  |
|  |  |
|  | # # --- TRAIN SET CREATOR --- # # |
|  | class TrainSet(): |
|  |  |
|  | # Split the DataFrame to create a train set. |
|  | def createTrainSet(df): |
|  | train\_set = pd.DataFrame() |
|  | count = 0 |
|  | row = [] |
|  | columns = 0 |
|  | while count < len(df): |
|  | row.append(columns) |
|  | columns += 2 |
|  | count += 2 |
|  | origin = df |
|  | data = origin.iloc[row] |
|  | train\_set=pd.concat([train\_set, data]) |
|  |  |
|  | return train\_set |
|  |  |
|  |  |
|  | # # --- TEST SET CREATOR --- # # |
|  | class TestSet(): |
|  |  |
|  | # Split the DataFrame to create a test set. |
|  | def createTestSet(df): |
|  | test\_set = pd.DataFrame() |
|  | count = 1 |
|  | row = [] |
|  | columns = 1 |
|  | while count < len(df): |
|  | row.append(columns) |
|  | columns += 2 |
|  | count += 2 |
|  | origin = df |
|  | data = origin.iloc[row] |
|  | test\_set=pd.concat([test\_set, data]) |
|  |  |
|  | return test\_set |
|  |  |
|  |  |
|  | # # --- TWEET CLASSIFICATOR - functionality to classify and analyze accuracy --- # # |
|  | class TweetClassificator(): |
|  |  |
|  | # This function execute the SGDClassifier. |
|  | def sdg(tweet\_df): |
|  | X = tweet\_df.text |
|  | y = tweet\_df.hashtag |
|  | X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state = 42) |
|  |  |
|  | sgd = Pipeline([('vect', CountVectorizer()), |
|  | ('tfidf', TfidfTransformer()), |
|  | ('clf', SGDClassifier(loss='hinge', penalty='l2',alpha=1e-3, random\_state=42, max\_iter=5, tol=None)), |
|  | ]) |
|  | sgd.fit(X\_train, y\_train) |
|  |  |
|  | y\_pred = sgd.predict(X\_test) |
|  |  |
|  | return {"y\_pred": y\_pred, "y\_test": y\_test} |
|  |  |
|  |  |
|  | # # --- SENTIMENT ANALYSIS --- # # |
|  | class SentimentAnalysis(): |
|  |  |
|  | # Function to get the subjectivity |
|  | def getSubjectivity(text): |
|  | return TextBlob(text).sentiment.subjectivity |
|  |  |
|  | # Function to get the polarity |
|  | def getPolarity(text): |
|  | return TextBlob(text).sentiment.polarity |
|  |  |
|  | # Get the polarity and return classification. |
|  | def getAnalysis(score): |
|  | if score > 0: |
|  | return "Positive" |
|  | elif score == 0: |
|  | return "Neutral" |
|  | else: |
|  | return "Negative" |
|  |  |
|  |  |
|  | # # --- SENTIMENT ANALYSIS GRAPH--- # # |
|  | class GraphAnalysis(): |
|  |  |
|  | # Get the sentiment analysis and create bar graphs for each hashtag . |
|  | def getGraph(dataframe, hash\_tag\_list): |
|  | for hashtag in hash\_tag\_list: |
|  | df = pd.DataFrame() |
|  | df=pd.concat([dataframe, df]) |
|  | dfc = df[df['hashtag'].str.contains(hashtag)] |
|  |  |
|  | plt.title("Sentiment Analysis - %s" % hashtag) |
|  | plt.xlabel('Sentiment') |
|  | plt.ylabel('Counts') |
|  | dfc['Analysis'].value\_counts().plot(kind = 'bar') |
|  | plt.show() |
|  |  |
|  |  |
|  | # # --- WORDCLOUD --- # # |
|  | class CloudOfWord(): |
|  | from wordcloud import WordCloud |
|  |  |
|  | # Get tweet text and create a word cloud with most used words ofr each hashtag. |
|  | def getWordCloud(dataframe, hash\_tag\_list): |
|  | for hashtag in hash\_tag\_list: |
|  | df = pd.DataFrame() |
|  | df=pd.concat([dataframe, df]) |
|  | dfc = df[df['hashtag'].str.contains(hashtag)] |
|  |  |
|  | allWords = ' '.join([twts for twts in dfc['text']]) |
|  | wordCloud = WordCloud(width=500, height=300, random\_state=21, max\_font\_size=110).generate(allWords) |
|  | plt.imshow(wordCloud, interpolation="bilinear") |
|  | plt.axis('off') |
|  | plt.show() |
|  |  |
|  |  |
|  | # # --- MAIN CLASS --- # # |
|  | if \_\_name\_\_=="\_\_main\_\_": |
|  |  |
|  | # # ---CREATE DATAFRAME --- # # |
|  | tweet\_analyzer = TweetAnalyzer() |
|  | hash\_tag\_list = ["Covid-19", "Ireland", "Data Mining", "Holiday"] |
|  | date\_since = "2020-04-01" |
|  | tweet\_df = tweet\_analyzer.tweets\_to\_data\_frame() |
|  |  |
|  | # # --- PRINT DATAFRAME--- # # |
|  | pd.set\_option("max\_colwidth", 40) |
|  | print("\n-------------DataFrame-------------") |
|  | print(tweet\_df.head(400)) |
|  |  |
|  | # # --- CREATE AND PRINT TRAIN SET --- # # |
|  | train\_set = TrainSet.createTrainSet(tweet\_df) |
|  | pd.set\_option("max\_colwidth", 40) |
|  | print("\n-------------Train Set-------------") |
|  | print(train\_set.head(200)) |
|  |  |
|  | # # --- CREATE AND PRINT TEST SET --- # # |
|  | test\_set = TestSet.createTestSet(tweet\_df) |
|  | pd.set\_option("max\_colwidth", 40) |
|  | print("\n-------------Test Set-------------") |
|  | print(test\_set.head(200)) |
|  |  |
|  | # # --- ANALYSIS --- # # |
|  | tweetClassificator = TweetClassificator.sdg(tweet\_df) |
|  | y\_pred = tweetClassificator["y\_pred"] |
|  | y\_test = tweetClassificator["y\_test"] |
|  |  |
|  | print("\n-------------Confusion Matrix-------------\n") |
|  | results = confusion\_matrix(y\_test, y\_pred) |
|  | print(results) |
|  |  |
|  | print("\n-------------Accurancy-------------\n") |
|  | print('accuracy %s' % accuracy\_score(y\_pred, y\_test)) |
|  |  |
|  | print("\n-------------Classification Report-------------\n") |
|  | print(classification\_report(y\_test, y\_pred, target\_names=hash\_tag\_list)) |
|  |  |
|  | # # --- SENTIMENT ANALYSIS --- # # |
|  | tweet\_df['Subjectivity'] = tweet\_df['text'].apply(SentimentAnalysis.getSubjectivity) |
|  | tweet\_df['Polarity'] = tweet\_df['text'].apply(SentimentAnalysis.getPolarity) |
|  | tweet\_df['Analysis'] = tweet\_df['Polarity'].apply(SentimentAnalysis.getAnalysis) |
|  | pd.set\_option("max\_colwidth", 40) |
|  | print("\n-------------Sentiment Analysis Set-------------") |
|  | print(tweet\_df.head(400)) |
|  |  |
|  | # # --- SENTIMENT ANALYSIS GRAPH --- # # |
|  | graph = GraphAnalysis.getGraph(tweet\_df, hash\_tag\_list) |
|  | print("\n-------------Sentiment Analysis Graph-------------\n") |
|  | print("\nGraphs successfully created.\n") |
|  |  |
|  | # # --- WORD CLOUD --- # # |
|  | cloud = CloudOfWord.getWordCloud(tweet\_df, hash\_tag\_list) |
|  | print("\n-------------Word Cloud-------------\n") |
|  | print("\nWord Clouds successfully created.\n") |