Dupli Checker

PLAGIARISM SCAN REPORT

4%
Plagiarised

96%
Unique

Date	2021-04-16
Words	884
Characters	8288

Content Checked For Plagiarism

```
#importing the packages panda and numpy
import pandas as pd
import numpy as np
#Reading the dataset(csv file) into the dataframe
df = pd.read_csv(r'/Users/bhavya/Downloads/Coronavirus Tweets.CSV')
#Preprocessing the dataset
#Dropping columns that aren't required
to_drop = ['reply_to_status_id',
           'reply_to_user_id',
           'reply_to_screen_name',
           'country_code',
           'place_full_name',
           'place_type',
           'account_lang']
df.drop(to_drop, inplace=True, axis=1)
df = df.dropna(how = 'all')
#Filtering the dataset such that it contains tweets only of english language
df.drop(df[df['lang'] != "en"].index, inplace = True)
#After applying above filters the dataframe is converted to csv and it will be our dataset.
df.to_csv("File2.csv", index=False)

#importing the basic packages
import io
import random
import string
import warnings
import pandas as pd
import numpy as np
import advertools as adv
#importing package to generate wordcloud
from wordcloud import WordCloud
#In the context of the project this package is required for generating stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
#Required Package to measure similarities between texts
from sklearn.metrics.pairwise import cosine_similarity
#package to display warning messages
import warnings
```

```
warnings.filterwarnings('ignore')

#importing natural language processing toolkits
import nltk
#Package to break the sentences from the paragaraph of the tweet text into tokens
from nltk.tokenize import sent_tokenize
#importing words and stopwords from the corpus of nltk for wordcloud and sentiment analysis
from nltk.corpus import words
from nltk.corpus import stopwords
#package to break the sentences of the tweet text into seperate words
from nltk.tokenize import word_tokenize
#This package is imported to accept the list of tokenized words and stems it into root word.
from nltk.stem import WordNetLemmatizer
#This package is imported to accept the list of tokenized words and stems it into root word.
from nltk.stem import PorterStemmer
#importing these packages to perform Sentiment Analysis
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from nltk.sentiment.util import *


# sklearn imports
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics


# python imports
import re
import os
from collections import Counter
import datetime as dt


# Visualization
from matplotlib import pyplot as plt
from matplotlib import ticker
import seaborn as sns
from sklearn import feature_extraction, linear_model, model_selection, preprocessing
from wordcloud import WordCloud
from tqdm import tqdm_notebook

# Saving models
import pickle
import plotly.express as px
import chart_studio.plotly as py

#Location of our csv file
data_dir = 'NewFolder'
#initialising a list
tweets = []
#Extracting the tweets from the dataset and appending it to the tweets list and storing it to dataframe
for file in sorted(os.listdir(data_dir)):
    tweets.append(pd.read_csv(data_dir + '/' + file, lineterminator = '\n'))
df = pd.concat(tweets)
```

```
df.tail()

'''Code to analyse frequency of words from the tweet text'''
#Extracting only tweet text from dataframe
text_en = df['text']
#Dataset Preparation : Removing URLs,Stopwords,punctuations and hashtags and converting text into lower case
text_en_lr = text_en.apply(lambda x: re.sub(r"https\S+", "", str(x)))
text_en_lr_lc = text_en_lr.apply(lambda x: x.lower())
text_en_lr_lc_pr = text_en_lr_lc.apply(lambda x: x.translate(str.maketrans('', '', string.punctuation)))
stop_words = set(stopwords.words('english'))
stop_words.update(['#coronavirus', '#coronavirusoutbreak', '#coronavirusPandemic', '#covid19', '#covid_19', '#epitwitter',
'#ihavecorona', 'amp','coronavirus', 'covid19'])
#extracting the words and storing
text_en_lr_lc_pr_sr = text_en_lr_lc_pr.apply(lambda x: ' '.join([word for word in x.split() if word not in stop_words]))
word_list = [word for line in text_en_lr_lc_pr_sr for word in line.split()]
#Using seaborn library for styling
sns.set(style="darkgrid")
#making use of counter package to count the frequency of words
counts = Counter(word_list).most_common(50)
counts_df = pd.DataFrame(counts)
counts_df
counts_df.columns = ['word', 'frequency']
#The bargraph is plotted against frequency and the word
#plots displayed using matlplots and seaborn
fig, ax = plt.subplots(figsize = (12, 12))
ax = sns.barplot(y="word", x='frequency', ax = ax, data=counts_df)
plt.savefig('wordcount_bar.png')
tweets = pd.read_csv("File1.csv")
#Function to get the hashtags
#if a particular hashtag is present in the text appending the hashtag to the list
def get_hashtag(row):
    tweet = []
    text = row["text"].lower()
    #Check for hashtag coronavirus
    if "#coronavirus" in text :
        tweet.append("#coronavirus")
    #Check for hashtag covid19
    if "#covid19" in text :
        tweet.append("#covid19")
    #check for hashtag lockdown
    if "#lockdown" in text :
        tweet.append("#lockdown")
    #check for hashtag stayhomestaysafe
    if "#stayhomestaysafe" in text  :
        tweet.append("#stayhomestaysafe")


    return ",".join(tweet)
#Function call
#Using the apply method to work for every row and column
tweets["hashtags"] = tweets.apply(get_hashtag,axis=1)
#finding the frequency of each hashtag
counts = tweets["hashtags"].value_counts()
#plotting the count as a bargraph
plt.bar(range(len(counts)), counts)
```

```
plt.show()
print(counts)

#Storing the status id that used the particular hashtag
cl_tweets = tweets["status_id"][tweets["hashtags"] == "#coronavirus"]
sa_tweets = tweets["status_id"][tweets["hashtags"] == "#covid19"]
tr_tweets = tweets["status_id"][tweets["hashtags"] == "#lockdown"]
a=tweets["status_id"][tweets["hashtags"] == "#stayhomestaysafe"]
#plotting the histogram
plt.hist([
        cl_tweets,
        sa_tweets,
        tr_tweets,
        a
    ],
     stacked=True,
     label=["coronavirus", "covid", "lockdown","stayhomestaysafe"])
#Giving labels to the plot
plt.legend()
plt.title("Tweets mentioning each hashtag")
plt.xlabel("Status id")
plt.ylabel("# of tweets")
plt.show()

#using the advertools method to extract mentions and hashtags
[x for x in dir(adv) if x.startswith('extract')]
#methods to analyse hashtags
hashtag_summary = adv.extract_hashtags(tweets['text'])
hashtag_summary.keys()
#number of hashtags in the dataset
hashtag_summary['overview']
#To get hashtags used by people
hashtag_summary['hashtags']
#Hashtags limited to a count of 10 sets
#Multidimensional list of the hashtags
hashtag_summary['hashtags'][:10]
#Single dimensional list of the first 10 hashtags extracted
hashtag_summary['hashtags_flat'][:10]
#Counting the hashtags used overall and hashtags used per tweet
hashtag_summary['hashtag_counts'][:20]
hashtag_summary['hashtag_freq'][:15]
#plotting the hashtags per tweet against the total number of tweets
plt.figure(facecolor='#ebebeb', figsize=(11, 8))
plt.bar([x[0] for x in hashtag_summary['hashtag_freq'][:15]],
        [x[1] for x in hashtag_summary['hashtag_freq'][:15]])
#Labelling the plot
plt.title('Hashtag frequency', fontsize=18)
plt.xlabel('Hashtags per tweet', fontsize=12)
plt.ylabel('Number of tweets', fontsize=12)
#Setting up the dimensions
plt.xticks(range(16))
plt.yticks(range(0,28000, 1000))
plt.grid(alpha=0.5)
plt.gca().set_frame_on(False)
fig = px.bar(x=freq[2:], y=top_hashtags[2:], orientation='h')
```

```
fig.update_layout(
    height=600, width=700,
    title_text='Most Popular Hashtags',
    xaxis = {'title': 'Frequency'},
    yaxis = {'autorange': "reversed", 'title':''}
)
fig.show()
mention_summary = adv.extract_mentions(tweets['text'])
mention_summary.keys()
#extracting the number of mentions from tweet set
mention_summary['overview']
#Multidimensional list of mentions
mention_summary['mentions'][:15]
#single dimensional list of mentions
```

Matched Source

Similarity 5%

Title: EDA ON #SSR | Kaggle

'lengthoftweet': The total length of the tweet posted by the user (words). ... ['word', 'frequency'] fig, ax = plt.subplots(figsize = (12, 12)) ax = sns.barplot(y="word", ...

https://www.kaggle.com/suyashpratapsingh/eda-on-ssr