Project Title: Analyzing Social Media Sentiment: Trends in Public Opinion Over Time

By Team Avengers Likhitha Gade Harichandana Mukkamala Chiranjeevi Uppala Reddy Krishna Reddy Yeddula

Social Media has opened a whole new world for people by connecting people around the globe. People are just a click away from getting a huge chunk of information. With information comes people's opinions, and with this comes people's positive and negative outlook regarding a topic. Sometimes, this also results in bullying and passing on hate comments about someone or something. By looking at large-scale sentiment trends that the datasets are accessible on, we may find patterns, link them to important world events, and comprehend the wider social influence of online talks. Our objective is to offer real-time insights into people's reactions to significant events, including social movements, product debuts, governmental decisions, and geopolitics.

Importing the necessary libraries

```
import seaborn as sns
import warnings
import re
import pandas as pd
import numpy as np
import string
import nltk
import matplotlib.pyplot as plt

warnings.filterwarnings("ignore", category = DeprecationWarning)

%matplotlib inline
```

Reading the train data:

- The first line will import the data using pandas
- In the second line we will make a backup/copy of the original data to keep it as it is.

```
train = pd.read_csv(r'C:\Users\PC\Downloads\train.csv')

train_orignal = train.copy()
```

```
train_original = train.copy()
print(train.shape)
```

```
(31962, 3)
```

```
print(train.columns)
```

```
Index(['id', 'label', 'tweet'], dtype='object')
```

Overview of the Training Data

```
train.head()
```

```
train.tail()
```

Reading the Test Data:

- First line Import Data
- Second Line backs up the original data

```
test = pd.read_csv(r'C:\Users\PC\Downloads\test.csv')

test_original = test.copy()
print(test.shape)
```

```
(17197, 2)
```

```
print(test.columns)
```

```
Index(['id', 'tweet'], dtype='object')
```

Overview of the test data:

```
test.head()
```

```
test.tail()
```

Data Pre-processing

Combining the datasets

```
combined_data = pd.concat([train, test], ignore_index=True, sort=True)
combined_data.head()
```

```
combined_data.tail()
```

Cleaning Data:

Removing the Usernames (@)

```
def remove_pattern(text,pattern):

    # re.findall() finds the pattern in the text and will put it in a list
    r = re.findall(pattern,text)

    # re.sub() will substitute all the @ with an empty character
    for i in r:
        text = re.sub(i,"",text)

    return text
```

Making a column for the cleaned Tweets

- We will use regex for and `np.vectorize()` for faster processing

```
combined_data['Cleaned_Tweets'] = np.vectorize(remove_pattern)(combined_data['tweet'],"@[

combined_data.head()
```

Now Removing punctuations, numbers and special characters

```
combined_data['Cleaned_Tweets'] = combined_data['Cleaned_Tweets'].str.replace("[^a-zA-Z#]

combined_data.head()
```

Removing Short Words:

- Words such as "hmm", "ok" etc. of length less than 3 are of no use

```
combined_data['Cleaned_Tweets'] = combined_data['Cleaned_Tweets'].apply(lambda x: ' '.joi

combined_data.head()
```

Tokenization:

- We will now tokenize the cleaned tweets as we will apply `Stemming` from `nltk`

```
tokenized_tweets = combined_data['Cleaned_Tweets'].apply(lambda x: x.split())

tokenized_tweets.head()
```

```
0    [when, father, dysfunctional, selfish, drags, ...
1    [thanks, #lyft, credit, can't, cause, they, do...
2                              [bihday, your, majesty]
3    [#model, love, take, with, time, urð±!!!, ð...
4                  [factsguide:, society, #motivation]
Name: Cleaned_Tweets, dtype: object
```

Stemming:

- Stemming is a step-based process of stripping the suffixes ("ing","ly",etc.) from a word

```
from nltk import PorterStemmer

ps = PorterStemmer()

tokenized_tweets = tokenized_tweets.apply(lambda x: [ps.stem(i) for i in x])

tokenized_tweets.head()
```

```
0    [when, father, dysfunct, selfish, drag, kid, i...
1    [thank, #lyft, credit, can't, caus, they, don'...
```

```
2                              [birthday, your, majesti]
3    [#model, love, take, with, time, urð±!!!, ð...
4                        [factsguide:, societi, #motiv]
Name: Cleaned_Tweets, dtype: object
```

Now lets combine the data back:

```
for i in range(len(tokenized_tweets)):
    tokenized_tweets[i] = ' '.join(tokenized_tweets[i])

combined_data['Clean_Tweets'] = tokenized_tweets
combined_data.head()
```

Data Visualization:

We will visualize the data using WordCloud

```
from wordcloud import WordCloud,ImageColorGenerator
from PIL import Image
import urllib
import requests
```

Storing all the non-sexist/racist words

```
positive_words = ' '.join(text for text in combined_data['Cleaned_Tweets'][combined_data[


# Generating images
Mask = np.array(Image.open(requests.get('http://clipart-library.com/image_gallery2/Twitte

# We will use the ImageColorGenerator to generate the color of the image
image_color = ImageColorGenerator(Mask)

# Now we will use the WordCloud function of the wordcloud library
wc = WordCloud(background_color='black',height=1500,width=4000,mask=Mask).generate(positi


# Size of the image generated
plt.figure(figsize=(10,20))

# Here we recolor the words from the dataset to the image's color
# interpolation is used to smooth the image generated
```

```
plt.imshow(wc.recolor(color_func=image_color),interpolation="hamming")

plt.axis('off')
plt.show()
```

Now lets store the words with label '1':

```
negative_words = ' '.join(text for text in combined_data['Clean_Tweets'][combined_data['l


# Combining Image with Dataset
Mask = np.array(Image.open(requests.get('http://clipart-library.com/image_gallery2/Twitte

image_colors = ImageColorGenerator(Mask)

# Now we use the WordCloud function from the wordcloud library
wc = WordCloud(background_color='black', height=1500, width=4000,mask=Mask).generate(nega


# Size of the image generated
plt.figure(figsize=(10,20))

# Here we recolor the words from the dataset to the image's color
# recolor just recolors the default colors to the image's blue color
# interpolation is used to smooth the image generated
plt.imshow(wc.recolor(color_func=image_colors),interpolation="gaussian")

plt.axis('off')
plt.show()
```

Now Extracting hastags from tweets:

```
def extractHashtags(x):
    hashtags = []

    # Loop over the words in the tweet
    for i in x:
        ht = re.findall(r'#(\w+)',i)
        hashtags.append(ht)

    return hashtags


positive_hashTags = extractHashtags(combined_data['Cleaned_Tweets'][combined_data['label'
```

```
positive_hashTags
```

```
[['run'],
 ['lyft', 'disapointed', 'getthanked'],
 [],
 ['model'],
 ['motivation'],
 ['allshowandnogo'],
 [],
 ['school',
  'exams',
  'hate',
  'imagine',
  'actorslife',
  'revolutionschool',
  'girl'],
 ['allin', 'cavs', 'champions', 'cleveland', 'clevelandcavaliers'],
 ['gr8'],
 ['ireland', 'blog', 'silver', 'gold', 'forex'],
 ['orlando',
  'standwithorlando',
  'pulseshooting',
  'orlandoshooting',
  'biggerproblems',
  'selfish',
  'heabreaking',
  'values',
  'love'],
 ['80days', 'gettingfed'],
 ['got7', 'junior', 'yugyoem', 'omg'],
 ['thankful', 'positive'],
 ['friday', 'cookies'],
 [],
 ['euro2016'],
 ['badday', 'coneofshame', 'cats', 'pissed', 'funny', 'laughs'],
 ['wine', 'weekend'],
 ['tgif', 'gamedev', 'indiedev', 'indiegamedev', 'squad'],
 ['upsideofflorida', 'shopalyssas', 'love'],
 ['smiles', 'media', 'pressconference', 'antalya', 'turkey', 'throwback'],
 ['ica16'],
 [],
 ['rip', 'orlando'],
 [],
 ['alohafriday', 'time', 'not', 'exist', 'positivevibes', 'hawaiian'],
 [],
 ['goodnight', 'badmonday'],
 ['taylorswift1989'],
 ['travelingram', 'dalat', 'ripinkylife'],
 ['photoshop',
  'enoughisenough',
  'dontphotoshopeverything',
  'wheresallthenaturalphotos'],
 ['cedarpoint'],
 ['thankful', 'positive'],
```

```
 ['bookworm', 'ontothenexthovel'],
 [],
 [],
 [],
 ['flowers',
  'grow',
```

Now unnesting the list:

```
positive_hastags_unnested = sum(positive_hashTags,[])
positive_hastags_unnested
```

```
['run',
 'lyft',
 'disapointed',
 'getthanked',
 'model',
 'motivation',
 'allshowandnogo',
 'school',
 'exams',
 'hate',
 'imagine',
 'actorslife',
 'revolutionschool',
 'girl',
 'allin',
 'cavs',
 'champions',
 'cleveland',
 'clevelandcavaliers',
 'gr8',
 'ireland',
 'blog',
 'silver',
 'gold',
 'forex',
 'orlando',
 'standwithorlando',
 'pulseshooting',
 'orlandoshooting',
 'biggerproblems',
 'selfish',
 'heabreaking',
 'values',
 'love',
 '80days',
 'gettingfed',
 'got7',
 'junior',
 'yugyoem',
 'omg',
 'thankful',
 'positive'
```

```
 'positive',
 'friday',
 'cookies',
 'euro2016',
 'badday',
 'coneofshame',
 'cats',
 'pissed',
 'funny',
 'laughs',
 'wine',
 'weekend',
 'tgif',
 'gamedev',
 'indiedev',
 'indiegamedev',
 'squad',
```

Now storing the negative hastags:

```
negative_hashtags = extractHashtags(combined_data['Cleaned_Tweets'][combined_data['label'
```

```
negative_hashtags_unnest = (sum(negative_hashtags,[]))
negative_hashtags_unnest
```

```
['cnn',
 'michigan',
 'tcot',
 'australia',
 'opkillingbay',
 'seashepherd',
 'helpcovedolphins',
 'thecove',
 'helpcovedolphins',
 'neverump',
 'xenophobia',
 'love',
 'peace',
 'race',
 'identity',
 'medâ',
 'altright',
 'whitesupremacy',
 'linguistics',
 'race',
 'power',
 'raciolinguistics',
 'brexit',
 'people',
 'trump',
 'republican',
 'michelleobama',
 'knicks',
```

```
 'golfâ',
 'jewishsupremacist',
 'libtard',
 'sjw',
 'liberal',
 'politics',
 'trash',
 'hate',
 'â',
 'stereotyping',
 'prejudice',
 'hope',
 'hate',
 'conflictâ',
 'pols',
 'bluelivesmatter',
 'draintheswamp',
 'ferguson',
 '2016',
 'antisemitism',
 'hocoschools',
 'columbiamd',
 'hocomd',
 'nazi',
 'hatred',
 'bigotry',
 'libtard',
 'sjw',
 'liberal',
 'politics',
```

Plotting Bar Plots:

- Word Frequencies:

```
positive_word_freq = nltk.FreqDist(positive_hastags_unnested)

positive_word_freq
```

```
FreqDist({'love': 1531, 'positive': 874, 'healthy': 570, 'smile': 548, 'thankful':
491, 'fun': 434, 'life': 405, 'summer': 367, 'model': 364, 'affirmation': 363, ...})
```

Now creating a dataframe of the most frequently used words in hashtags :

```
positive_df = pd.DataFrame({'Hashtags': list(positive_word_freq.keys()),'Count' : list(po
positive_df
```

Plotting the bar plot for 20 most frequent words:

```
positive_df_plot = positive_df.nlargest(20,columns='Count')

sns.barplot(data=positive_df_plot,y='Hashtags',x='Count')
sns.despine()
```

Negative Word Frequency:

```
negative_word_freq = nltk.FreqDist(negative_hashtags_unnest)

negative_word_freq
```

```
FreqDist({'trump': 133, 'politics': 94, 'allahsoil': 92, 'libtard': 76, 'liberal':
75, 'sjw': 74, 'retweet': 57, 'black': 44, 'miamiâ': 38, 'hate': 32, ...})
```

Creating a dataset of the most frequent words:

```
negative_df = pd.DataFrame({'Hashtags':list(negative_word_freq.keys()),'Count':list(negat

negative_df
```

Plotting the bar plot for the 20 most frequent negative words:

```
negative_df_plot = negative_df.nlargest(20,columns='Count')

sns.barplot(data=negative_df_plot,y='Hashtags',x='Count')
sns.despine()
```

Feature Extraction from Cleaned Tweets:

- Applying Bag of Words method to embed the data
- using `Count Vectorizer` package

```
from sklearn.feature_extraction.text import CountVectorizer
```

```
bow_vecotrizer = CountVectorizer(max_df=0.90, min_df = 2, max_features = 1000, stop_words

bow = bow_vecotrizer.fit_transform(combined_data['Cleaned_Tweets'])

bow_df = pd.DataFrame(bow.todense())

bow_df
```

TF-IDF Features:

Term-Frequency (TF):

The first computes the normalized Term Frequency (TF), aka. the number of times a word appears in a document, divided by the total number of words in that document. The Term Frequency is calculated as:

image.png

Inverse-Document Frequency (IDF):

The second term is the Inverse Document Frequency (IDF), computed as the logarithm of the number of the documents in the corpus divided by the number of documents where the specific term appears. The IDF is calulated as:

image.png

Now lets apply this to our dataset

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(max_df=0.90,min_df=2,max_features=1000,stop_words='english')

tfidf_matrix = tfidf.fit_transform(combined_data['Cleaned_Tweets'])

tfidf_df = pd.DataFrame(tfidf_matrix.todense())

tfidf_df
```

```
train_bow = bow[:31962]
```

```
train_bow = bow[:31962]

train_bow.todense()
```

```
matrix([[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]], shape=(31962, 1000))
```

```
train_tfidf_matrix = tfidf_matrix[:31962]

train_tfidf_matrix.todense()
```

```
matrix([[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]], shape=(31962, 1000))
```

Splitting data into training data and test data:

```
from sklearn.model_selection import train_test_split
```

Bag of Words Features:

```
x_train_bow, x_valid_bow, y_train_bow, y_valid_bow = train_test_split(train_bow,train['la
```

TF-IDF Features:

```
x_train_tfidf, x_valid_tfidf, y_train_tfidf, y_valid_tfidf = train_test_split(train_tfidf
```

Applying ML Models:

The model we will be using is:

- **Logistic Regression**

```
from sklearn.metrics import f1_score
```

Logistic Regression:

```
from sklearn.linear_model import LogisticRegression


log_Reg = LogisticRegression(random_state=0,solver='lbfgs')
```

Fitting Bag of Words Features:

```
log_Reg.fit(x_train_bow,y_train_bow)
```

```
predict_bow = log_Reg.predict_proba(x_valid_bow)
predict_bow
```

```
array([[9.68652048e-01, 3.13479521e-02],
       [9.99852344e-01, 1.47656003e-04],
       [9.14995367e-01, 8.50046333e-02],
       ...,
       [6.37072882e-01, 3.62927118e-01],
       [9.45998334e-01, 5.40016663e-02],
       [9.68524880e-01, 3.14751200e-02]], shape=(9589, 2))
```

Calculating the F1-Score:

```
# If prediction is more than or equal to 0.3 then 1 else 0
prediction_int = predict_bow[:,1] >=0.3

# Converting to integer type
prediction_int = prediction_int.astype(int)
prediction_int

# Calculating f1 score
log_bow = f1_score(y_valid_bow, prediction_int)
log_bow
```

```
0.5510887772194305
```

Fitting TF-IDF Features:

```
log_Reg.fit(x_train_tfidf,y_train_tfidf)
```

```
predict_tfidf = log_Reg.predict_proba(x_valid_tfidf)
predict_tfidf
```

```
array([[0.97688537, 0.02311463],
       [0.96611218, 0.03388782],
       [0.93767463, 0.06232537],
       ...,
       [0.93478755, 0.06521245],
       [0.92462338, 0.07537662],
       [0.97813239, 0.02186761]], shape=(9589, 2))
```

```
prediction_int = predict_tfidf[:,1]>=0.3

prediction_int = prediction_int.astype(int)
prediction_int

log_tfidf = f1_score(y_valid_tfidf,prediction_int)
log_tfidf
```

```
0.5759162303664922
```

Predicting the test_data and storing it:

```
test_tfidf = tfidf_matrix[31962:]
test_pred = log_Reg.predict_proba(test_tfidf)

test_pred_int = test_pred[:,1] >= 0.3
test_pred_int = test_pred_int.astype(int)

test['label'] = test_pred_int

submission = test[['id','label']]
submission.to_csv('result.csv', index=False)
```

Results after prediction:

For a negative label : 1

For a positive label : 0

```
res = pd.read_csv('result.csv')
```

```
res
```

Summary:

F-1 Score of Model: 0.5315391084945332 (Bag of Words) & 0.5558534405719392 (TF-IDF) using Logistic Regression

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.