TWITTER SENTIMENT ANALYSIS AND SARCASM DETECTION (USING EMOTICONS)

VIBHU DAGAR - 16BCE2141

SHITIJ GUPTA - 16BCE0510

FACULTY : PROF. SHARMILA BANU

GITHUB LINK FOR THE PROJECT :

https://github.com/vida97/Sentiment-Analysis---Sarcasm-Detection-NLP-Project-

GITHUB LINK FOR TASKS :

https://github.com/vida97/Sentiment-Analysis---Sarcasm-Detection-NLP-Project-/tree/master/class_tasks

RESEARCH PAPER LINK :

https://github.com/ajinkyachavan/Emoticon-Based-Sentiment-Analysis-Python/blob/master/emoticon-based-sentiment-analysis-of-twitter-data.pdf

http://www.cs.columbia.edu/~julia/papers/Agarwaletal11.pdf

https://arxiv.org/ftp/arxiv/papers/1711/1711.10377.pdf

DATASET LINK :

https://raw.githubusercontent.com/ajinkyachavan/Emoticon-Based-Sentiment-Analysis-Python/master/data/final_tweets.csv

POSITIVE AND NEGETIVE WORDS LINK :

https://github.com/ajinkyachavan/Emoticon-Based-Sentiment-Analysis-Python/blob/master/positive-words.txt

https://github.com/ajinkyachavan/Emoticon-Based-Sentiment-Analysis-Python/blob/master/negative-words.txt

FLOW WISE REPRESENTATION OF THE PROJECT :

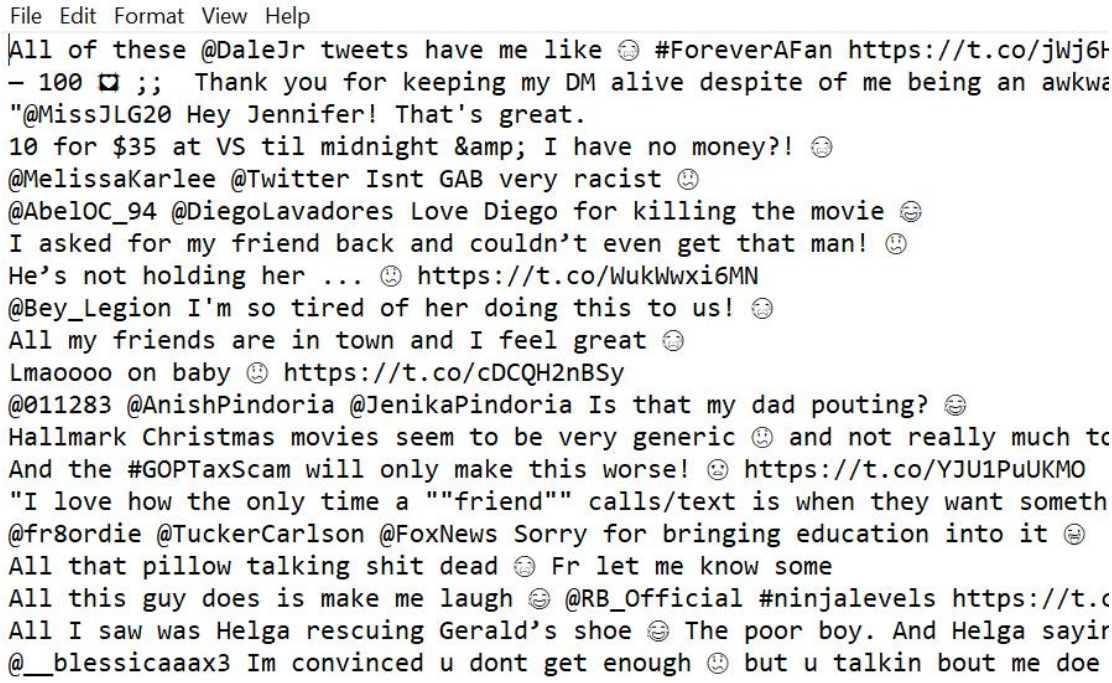
1. Importing all the libraries :

```
import nltk
import re
import enchant
from textblob import TextBlob
import matplotlib.pyplot as plt
import emoji
import pickle
import os
import sklearn
import numpy as np
```

2. Retrieving data from text files (initiating word lemmatizer) :

```
lemma = nltk.wordnet.WordNetLemmatizer()
d=enchant.Dict("en_US")
use_stemmer=True

f=open("sample1.txt","r")
x=f.readlines()


positive_file=open("abc.txt","r")
q=positive_file.readlines()
z=[]
for i in q:
    z.append(i[0:-1])


negetive_file=open("def.txt","r")
y=negetive_file.readlines()
m=[]
for i in y:
    m.append(i[0:-1])
```

-- x the data set of all the tweets.



File Edit Format View Help

All of these @DaleJr tweets have me like 😳 #ForeverAFan https://t.co/jWj6H
– 100 🎟 ;; Thank you for keeping my DM alive despite of me being an awkwa
"@MissJLG20 Hey Jennifer! That's great.
10 for $35 at VS til midnight & I have no money?! 😳
@MelissaKarlee @Twitter Isnt GAB very racist 🙁
@AbelOC_94 @DiegoLavadores Love Diego for killing the movie 😂
I asked for my friend back and couldn’t even get that man! 🙁
He’s not holding her ... 🙁 https://t.co/WukWwxi6MN
@Bey_Legion I'm so tired of her doing this to us! 😳
All my friends are in town and I feel great 😳
Lmaoooo on baby 🙁 https://t.co/cDCQH2nBSy
@011283 @AnishPindoria @JenikaPindoria Is that my dad pouting? 😂
Hallmark Christmas movies seem to be very generic 😕 and not really much to
And the #GOPTaxScam will only make this worse! 😟 https://t.co/YJU1PuUKMO
"I love how the only time a ""friend"" calls/text is when they want someth
@fr8ordie @TuckerCarlson @FoxNews Sorry for bringing education into it 😁
All that pillow talking shit dead 😳 Fr let me know some
All this guy does is make me laugh 😂 @RB_Official #ninjalevels https://t.c
All I saw was Helga rescuing Gerald’s shoe 😂 The poor boy. And Helga sayir
@__blessicaaax3 Im convinced u dont get enough 😕 but u talkin bout me doe

-- m has all the negative words and z has all the positive words.

File Edit Format View Help

abound
abounds
abundance
abundant
accessable
accessible
acclaim
acclaimed
acclamation
accolade
accolades
accommodative
accomodative
accomplish
accomplished
accomplishment
accomplishments
accurate
accurately
achievable

File Edit Format View Help

```
2-faced
2-faces
abnormal
abolish
abominable
abominably
abominate
abomination
abort
aborted
aborts
abrade
abrasive
abrupt
abruptly
abscond
absence
absent-minded
absentee
absurd
```

3. Pre-processing the tweets and assigning scores:

```
T_score=[]
T_score1=[]

def pre_tweet(tweet):
    tweet=tweet.lower()
    tweet=re.sub(r'((www\.[\S]+)|(https?://[\S]+))', '', tweet)
    tweet=re.sub(r'@[\S]+', '', tweet)
    tweet1=tweet.split()
    tweet2=TextBlob(tweet)
    t_score=0
    sc=tweet2.sentiment.polarity
    #print(tweet2.tags)
    #print("Score1 : ",sc)
    T_score1.append(sc)
    for words in tweet1:
        if d.check(words)==True:
            word = lemma.lemmatize(words)
            if words in z:
                t_score+=0.5
            if words in m:
                t_score-=0.5
            #print(word,end=' ')
    #print("Score 2 : ",t_score)
    T_score.append(t_score)



l=[]
for i in x[0:1000]:
    l.append(pre_tweet(i))
```

-- Tscore and Tscore1 lists are created to hold the scores of each tweet.
-- Function pre_tweets() is called to perform pre-processing
-- Website links and user mentions are handled.

```
['all', 'of', 'these', '#mention', 'tweets', 'have', 'me', 'like', 'happy', '??'
, '#foreverafan', 'URL']
['—', '100', '??', ';;', 'thank', 'you', 'for', 'keeping', 'my', 'dm', 'alive',
'despite', 'of', 'me', 'being', 'an', 'awkward', 'species', '??', 'lets', 'talk'
, 'more', 'in', 'the', 'future…', 'URL']
['"#mention', 'hey', 'jennifer!', "that's", 'great.']
['10', 'for', '$35', 'at', 'vs', 'til', 'midnight', '&amp;', 'i', 'have', 'no',
'money?!', '??']
['#mention', '#mention', 'isnt', 'gab', 'very', 'racist', '??']
['#mention', '#mention', 'diego', 'for', 'killing', 'the', 'movie', '??']
['i', 'asked', 'for', 'my', 'friend', 'back', 'and', 'couldn’t', 'even', 'get',
'that', 'man!', '??']
['he’s', 'not', 'holding', 'her', '...', '??', 'URL']
['#mention', "i'm", 'so', 'tired', 'of', 'her', 'doing', 'this', 'to', 'us!', '?
?']
['all', 'my', 'friends', 'are', 'in', 'town', '??']
['lmaoooo', 'on', 'baby', '??', 'URL']
```

-- For Tscore, all the words of a tweet are matched to check if they are positive or negative and accordingly scores are assigned to that particular tweet.
-- For Tsocre1 the score retrieved from TextBlob are considered for each tweet.

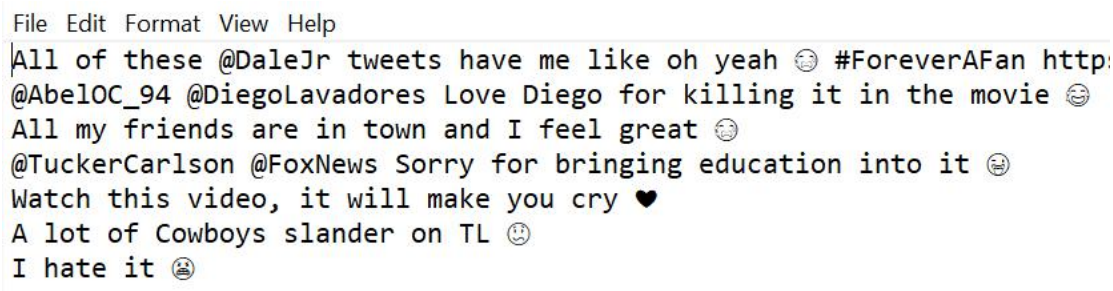
```
[1.0, 0.0, 0, 0, -0.5, -0.5, 0, 0, -0.5, 0, 0, 0, 1.0, 0, 0.5
0, 0.5, 0.5, 0, 0, 0.5, 0, 0, 0, 1.0, 0.5, 0, 0, -0.5, 0, 0,
5, 0, 0.0, 0, -0.5, 0.5, 0, 0, 0, 0, 0.0, 0, 0, -0.5, -1.0, (
-0.5, 0.5, 0, 0.5, 0, 0.5, 0, -0.5, 0.5, 0, 0, 0, 0, 0, 0, -(
.0, -0.5, 1.0, 1.0, 0, -0.5, 0.5, 0, -1.0, -0.5, 0, 0, 0, -0.
.0, 0, -0.5, -0.5, 0, 0.5]
[0.8, 0.0, 0.8, 0.0, 0.2, 0.0, 0.0, 0.0, -0.5, 0.0, 0.0, 0.0,
3, -0.25, 0.25, -0.5, -0.2, 0.3, 0.2333333333333333, 0.0, 0.4
.0, 0.075, 0.5, -0.8, 0.0, 0.0, 0.39999999999999997, 0.4, 0.9
0, 0.8, 0.0, 0.0, 0.0, 0.17142857142857143, 0.0, 0.0, 0.0, 0.
.5, 0.5, 0.0, 0.0, 0.06666666666666667, -0.5, 0.0, 0.0, -0.5,
, -0.2, 0.0, -0.3, 0.0, -0.5, 0.4, 0.0, 0.0, 0.0, 0.0, 0.1072
, -0.05, -0.17708333333333334, 0.0, 0.0, 0.25, 0.0, -0.316666
0000000000044, 0.2, 0.0, 0.06818181818181818, 0.0, 0.1875, 0.
0.0, -0.5, 0.06190476190476191, 0.0, -0.25, -0.03333333333333
, -0.4, -0.3125, 0.0, 0.22348484848484848, 0.0, 0.0, 0.0, 0.0
34, 0.0, 0.2]
```

4. Handling emoticons :

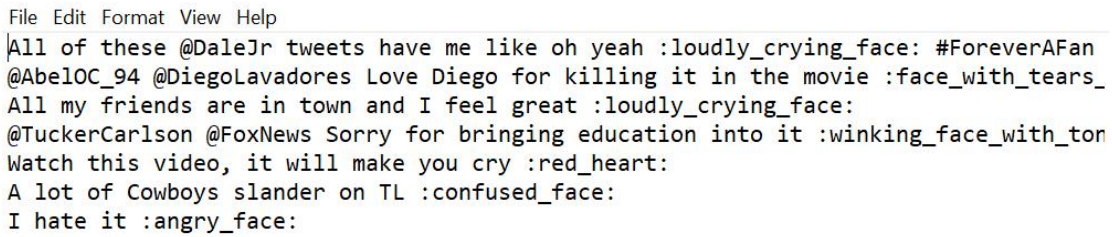
```
def preprocessing():
    with open('emoji.txt', 'r', encoding='utf16', errors='ignore') as file:
        text = file.read().split('\n')
    for i in range(len(text)):
        text[i] = emoji.demojize(text[i])
    with open('final_tweets.txt', 'w+', encoding='utf8', errors='ignore') as fil
        new_text = ""
        for i in range(len(text)):
            new_text += text[i] + "\n"
        file.write(new_text)

preprocessing()
```

-- Opening the file containing emoticons and the demojizing all the emojis found using emoji.demojize(text[])
-- Saving the tweets with handled emoticons in the file final_tweets

Before emoticons are handled :



After emoticons are handled :



5. Calculating the scores and visual representation of the scores :

```
print(T_score)
print(T_score1)
p,q,n=0,0,0
for i in T_score:
    if i>0:
        p+=1
    if i<0:
        q+=1
    if i==0:
        n+=1
ts1=[p,q,n]
labels=["positive","negetive","neutral"]
colors=["green","red","blue"]
patches, texts = plt.pie(ts1, colors=colors, shadow=True, startangle=90)
plt.legend(patches, labels, loc="best")
plt.axis('equal')
plt.tight_layout()
plt.show()
```

```
p,q,n=0,0,0
for i in T_score1:
    if i>0:
        p+=1
    if i<0:
        q+=1
    if i==0:
        n+=1
ts2=[p,q,n]
labels=["positive","negetive","neutral"]
colors=["green","red","blue"]
patches, texts = plt.pie(ts2, colors=colors, shadow=True, startangle=90)
plt.legend(patches, labels, loc="best")
plt.axis('equal')
plt.tight_layout()
plt.show()
print("\n")
print("Percentages Score 1 :")
print("  Positive : ",ts1[0]/(ts1[0]+ts1[1]+ts1[2])*100)
print("  Negetive : ",ts1[1]/(ts1[0]+ts1[1]+ts1[2])*100)
print("  Neutral : ",round(ts1[2]/(ts1[0]+ts1[1]+ts1[2])*100,2))
print("Percentages Score 2 : (TextBlob)")
print("  Positive : ",ts2[0]/(ts2[0]+ts2[1]+ts2[2])*100)
print("  Negetive : ",ts2[1]/(ts2[0]+ts2[1]+ts2[2])*100)
print("  Neutral : ",ts2[2]/(ts2[0]+ts2[1]+ts2[2])*100)
print("\n")
```

-- Calculating the percentages of positive, negative or neutral tweets for both the cases.

-- Visualizing the polarities using pie charts.

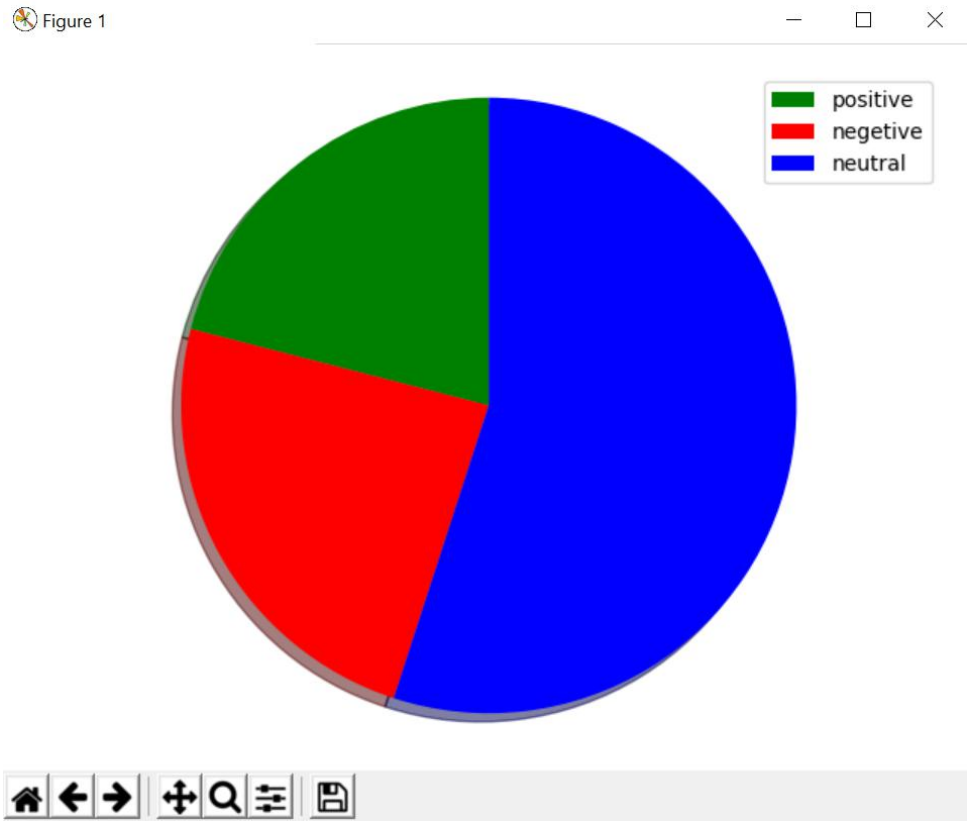


Figure 1
positive
negetive
neutral

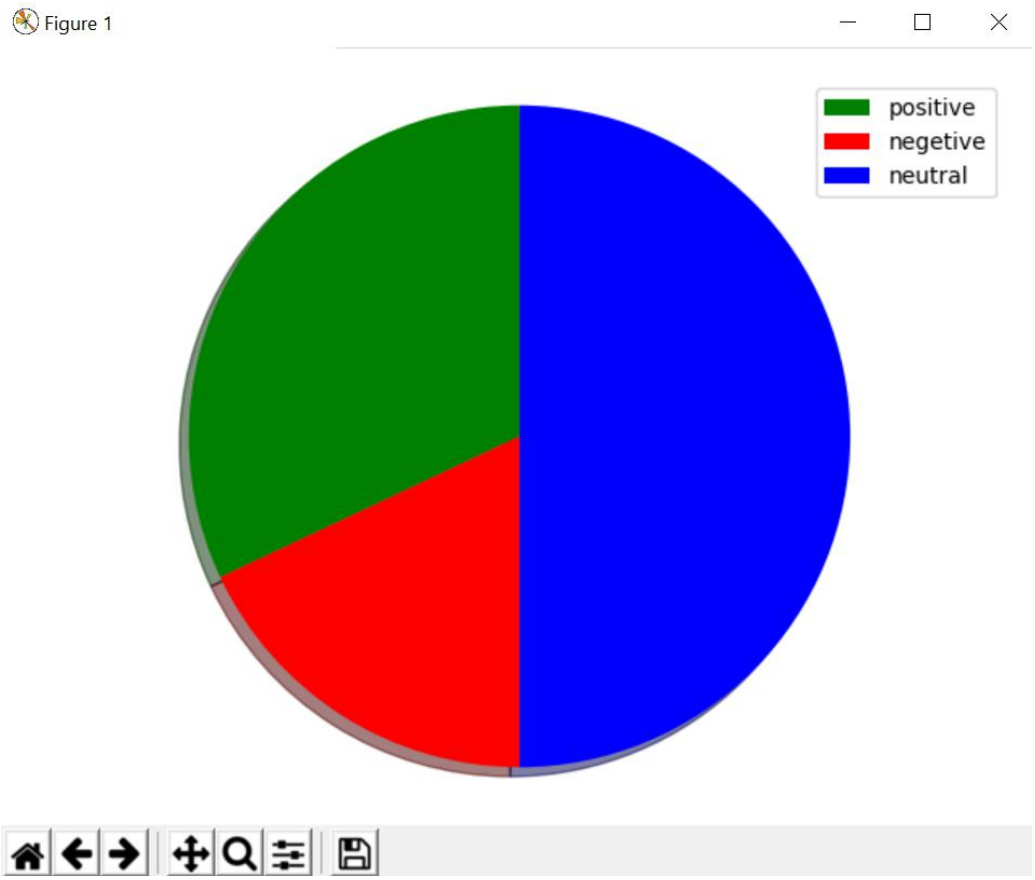


Figure 1
positive
negetive
neutral

6. Scoring the emoticons and text :

```
f=open('final_tweets.txt','r')
zz=f.readlines()
score_emo=0
ZZ=[]
for i in zz:
    ZZ.append(i[0:-1])
sc1=[]
sc2=[]
for i in ZZ:
    score_emo=0
    twe=TextBlob(i)
    dc=twe.sentiment.polarity
    sc1.append(dc*0.1)
    zzz=[]
    zzz=i.split(" ")
    for i in zzz:
        if i==":loudly_crying_face:":
            score_emo+=-0.093
        if i==":face_with_tears_of_joy:":
            score_emo+=0.221
        if i==":winking_face_with_tongue:":
            score_emo+=0.445
        if i==":confused_face:":
            score_emo+=0.001
        if i==":red_heart:":
            score_emo+=0.746
        if i==":angry_face:":
            score_emo+=-0.397
    sc2.append(score_emo)
print(sc1)
print(sc2)
```

-- Emoticons are given scores based on the research already done.
-- Their score hold more weightage than that of text present in the tweets.
-- Text in the tweets are scored through TextBlob Library.

7. Recognizing sarcasm :

```
fsc=[]
for i in range(len(sc1)):
    fsc.append(sc1[i]+sc2[i])
#print(fsc)
SC1=np.sign(sc1)
SC2=np.sign(sc2)
SC=[]
Sc=[]
for i in range(len(sc1)):
    if SC1[i]==SC2[i]:
        SC.append(SC1[i])
        Sc.append("no")
    else:
        SC.append(SC2[i])
        Sc.append("yes")
print(SC)
print(Sc)
```

-- The scores of the emoticons and text in the tweets are taken into consideration together hence generating the final score.
-- Also to identify the sarcasm in the statements/tweets the polarity of text is matched with that of the tweets.

```
[0.0, 0.05, 0.08000000000000002, -0.05, 0.0, 0.0, -0.08000000000000002]
[-0.093, 0.221, -0.093, 0.445, 0.746, 0.001, -0.397]
[-1.0, 1.0, -1.0, 1.0, 1.0, 1.0, -1.0]
['yes', 'no', 'yes', 'yes', 'yes', 'yes', 'no']
```