Name: Om Kadam

Class: BE EXTCA

Roll No: 28

PID: 213035

Date: 22/03/2025

Understanding Emotion in Text using Natural Language Processing

Import Functions & Data

In [6]:
```
import nltk
import pandas as pd
import matplotlib.pyplot as plt
import random
import re
import string
import numpy as np

from os import getcwd
from nltk.corpus import twitter_samples
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.tokenize import TweetTokenizer

nltk.download('twitter_samples')
nltk.download('stopwords')
```

```
[nltk_data] Downloading package twitter_samples to
[nltk_data]     /home/tetroner/nltk_data...
[nltk_data]   Package twitter_samples is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     /home/tetroner/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[6]:
```
True
```

In [7]:
```
def process_tweet(tweet):
    stemmer = PorterStemmer()
    stopwords_english = stopwords.words('english')

    # Remove stock market tickers like SGE
    tweet = re.sub(r'\$\w*', '', tweet)

    # Remove old style retweet text "RT"
    tweet = re.sub(r'^RT[\s]+', '', tweet)

    # Remove hyperlinks
    tweet = re.sub(r'https?://[^\s\n\r]+', '', tweet)

    # Remove hashtags
    tweet = re.sub(r'#', '', tweet)

    # Tokenize tweets
    tokenizer = TweetTokenizer(preserve_case=False, strip_handles=True, redu
    tweet_tokens = tokenizer.tokenize(tweet)

    tweets_clean = []
    for word in tweet_tokens:
        if word not in stopwords_english and word not in string.punctuation:
            stem_word = stemmer.stem(word) # Stemming words
            tweets_clean.append(stem_word)

    return tweets_clean
```

In [8]:
```
def build_freqs(tweets, ys):
    yslist = np.squeeze(ys).tolist()
    freqs = {}

    for y, tweet in zip(yslist, tweets):
        for word in process_tweet(tweet):
            pair = (word, y)

            if pair in freqs:
                freqs[pair] += 1
            else:
                freqs[pair] = 1
    return freqs
```

Prepare Data

In [9]:
```
# Select the set of positive & negative tweets
all_positive_tweets = twitter_samples.strings('positive_tweets.json')
all_negative_tweets = twitter_samples.strings('negative_tweets.json')

# Split the data into two pieces, one for training and one for testing(valid
test_pos = all_positive_tweets[4000:]
train_pos = all_positive_tweets[:4000]
test_neg = all_negative_tweets[4000:]
train_neg = all_negative_tweets[:4000]

train_x = train_pos + train_neg
test_x = test_pos + test_neg

# Combine positive & negative labels
train_y = np.append(np.ones((len(train_pos), 1)), np.zeros((len(train_neg),
test_y = np.append(np.ones((len(test_pos), 1)), np.zeros((len(test_neg), 1))

# Print the shape train & test sets
print("train_y.shape = " + str(train_y.shape))
print("test_y.shape = " + str(test_y.shape))

train_y.shape = (8000, 1)
test_y.shape = (2000, 1)

# Create a frequency dictionary
freqs = build_freqs(train_x, train_y)

# Check the output
print("type(freqs) = " + str(type(freqs)))
print("len(freqs) = " + str(len(freqs.keys())))
```

```
train_y.shape = (8000, 1)
test_y.shape = (1000, 2)
type(freqs) = <class 'dict'>
len(freqs) = 11397
```

Process Tweet

In [10]:
```
# Test the function below
print('This is an example of positive tweet: \n', train_x[0])
print('\n This is an example of the processed version of the tweet: \n', prc
```

```
This is an example of positive tweet:
 #FollowFriday @France_Inte @PKuchly57 @Milipol_Paris for being top engaged
members in my community this week :)

 This is an example of the processed version of the tweet:
 ['followfriday', 'top', 'engag', 'member', 'commun', 'week', ':)']
```

In [11]:
```
# UNQ_C1 Graded Function: Sigmoid
def sigmoid(z):
    # Calculate the sigmoid of z
    return 1 / (1 + np.exp(-z))

# Testing function
if sigmoid(0) == 0.5:
    print('SUCCESS')
else:
    print('FAILURE')

if sigmoid(4.92) == 0.9927537604041685:
    print('SUCCESS')
else:
    print('FAILURE')
```

```
SUCCESS
SUCCESS
```

In [12]:
```
# Verify that when the model predicts close to 1, but the actual label is 0,
-1*(1-0)*np.log(1-0.9999) #loss is about 9.2
```

Out[12]:
```
np.float64(9.210340371976294)
```

In [13]:
```
# UNQ_C2 Graded Function: gradientdescent
def gradientdescent(x, y, theta, alpha, num_iters):
    # Get 'm' (no. of rows in matrix x)
    m = x.shape[0]

    for i in range(0, num_iters):
        # Get 'z' (dot product of x & theta)
        z = np.dot(x, theta)

        # Get the sigmoid of z
        h = sigmoid(z)

        # Calculate the cost function
        # Ensure y & (1-y) are numpy arrays before calling transpose
        J = (-1./m) * (np.dot(np.array(y).transpose(), np.log(h)) + np.dot(r

        # Calculate the weights theta
        theta = theta - (alpha/m) * (np.dot(x.transpose(), (h-y)))
        J = float(J)

    return J, theta
```

In [15]:
```
# Construct a synthetic test case using numpy PRING Function
np.random.seed(1)

# X input is 10 x 3 with ones for the bias terms
tmp_X = np.append(np.ones((10,1)), np.random.rand(10,2)*2000, axis=1)

# Y Labels are 10 x 1
tmp_Y = (np.random.rand(10,1)>0.35).astype(float)
tmp_J, tmp_theta = gradientdescent(tmp_X, tmp_Y, np.zeros((3,1)), 1e-8, 700)

print(f"The cost after training is {tmp_J:.8f}.")
print(f"The resulting vector of weights is {[round(t, 8) for t in np.squeeze
```

```
The cost after training is 0.67094970.
The resulting vector of weights is [np.float64(4.1e-07), np.float64(0.000356
58), np.float64(7.309e-05)]
```

```
/tmp/ipykernel_77082/2495877286.py:19: DeprecationWarning: Conversion of an
array with ndim > 0 to a scalar is deprecated, and will error in future. Ens
ure you extract a single element from your array before performing this oper
ation. (Deprecated NumPy 1.25.)
  J = float(J)
```

In [17]:
```
def extract_features(tweet, freqs, pricess_tweet=process_tweet):
    word_l = process_tweet(tweet)
    x = np.zeros((1,3))
    x[0,0] =1

    for word in word_l:
        x[0,1] += freqs.get((word,1),0)
        x[0,2] += freqs.get((word,0),0)
    assert(x.shape == (1,3)) # This line was not indented properly. Now insi
    return x

tmp1 = extract_features(train_x[0], freqs)
print(tmp1)
```

```
tmp2 = extract_features('blorb bleeeeb bloooob', freqs)
print(tmp2)
```

```
[[1.000e+00 3.133e+03 6.100e+01]]
[[1. 0. 0.]]
```

Training the Model

In [19]:
```
# Collect the features 'x' and stack them into a matrix 'x'
x = np.zeros((len(train_x), 3))

for i in range(len(train_x)):
    x[i, :] = extract_features(train_x[i], freqs) # Changed X to x

# Training labels corresponding to x
Y= train_y
J, theta = gradientdescent(x, Y, np.zeros((3,1)), 1e-9, 1500) # Changed X to

print(f"The cost after training is {J:.8f}.")
print(f"The resulting vector of weights is {[round(t,8) for t in np.squeeze(
```

```
/tmp/ipykernel_77082/2495877286.py:19: DeprecationWarning: Conversion of an
array with ndim > 0 to a scalar is deprecated, and will error in future. Ens
ure you extract a single element from your array before performing this oper
ation. (Deprecated NumPy 1.25.)
  J = float(J)
The cost after training is 0.22524410.
The resulting vector of weights is [np.float64(6e-08), np.float64(0.0005378
6), np.float64(-0.00055885)]
```

Predicting

In [20]:
```
def predict_tweet(tweet, freqs, theta):
    # Extract the feature from tweets and store in x
    x = extract_features(tweet,freqs)

    # Make prediction using x & theta
    return sigmoid(np.dot(x, theta))
```

In [21]:
```
# Test the function
for tweet in ['I am happy', 'I am bad', 'this movie should have been great.'
    print( '%s -> %f' % (tweet, predict_tweet(tweet, freqs, theta)))
```

```
I am happy -> 0.519259
I am bad -> 0.494338
this movie should have been great. -> 0.515962
great -> 0.516052
great great -> 0.532070
great great great -> 0.548023
great great great great -> 0.563877
/tmp/ipykernel_77082/839043443.py:3: DeprecationWarning: Conversion of an ar
ray with ndim > 0 to a scalar is deprecated, and will error in future. Ensur
e you extract a single element from your array before performing this operat
ion. (Deprecated NumPy 1.25.)
  print( '%s -> %f' % (tweet, predict_tweet(tweet, freqs, theta)))
```

In [22]:
```
def test_logistic_regression(test_x, test_y, freqs, theta, predict_tweet=pre
    '''
    Input:
    test_x: a list of tweets
    test_y: (m,1) vector with the corresponding labels for the list of tweet
    freqs: a dictionary with the frequency of each pair (or tuple)
    theta: weight vector of dimension (3,1)
    Output:
    accuracy: (# of tweets classified correctly)/total # of tweets
    '''

    # The list for storing predictions
    y_hat = []
    for tweet in test_x:
        y_pred = predict_tweet(tweet, freqs, theta)

        if y_pred > 0.5:
            y_hat.append(1)
        else:
            y_hat.append(0)

    accuracy = (y_hat==np.squeeze(test_y)).sum()/len(test_x)
    return accuracy
```

In [23]:
```
tmp_accuracy = test_logistic_regression(test_x, test_y, freqs, theta)
print(f"Logistic regression model's accuracy = {tmp_accuracy:.4f}")
```

```
Logistic regression model's accuracy = 0.5005
```

In [24]:
```
# Some error analysis
print('Label Predicted Tweet')

for x,y in zip(test_x,test_y):
    y_hat = predict_tweet(x, freqs, theta)

    if np.abs(y - (y_hat > 0.5)) > 0:
        print('THE TWEET IS:', x)
        print('THE PROCESSED TWEET IS:', process_tweet(x))
        print('%d\t%0.8f\t%s' % (y, y_hat, ' '.join(process_tweet(x)).encode
Label Predicted Tweet
THE TWEET IS: @heyclaireee is back! thnx God!!! i'm so happy :)
THE PROCESSED TWEET IS: ['back', 'thnx', 'god', 'happi', ':)']
0       0.84399696      b'back thnx god happi :)'
THE TWEET IS: @HumayAG 'Stuck in the centre right with you. Clowns to the ri
ght, jokers to the left...' :) @orgasticpotency @ahmedshaheed @AhmedSaeedGah
aa
THE PROCESSED TWEET IS: ['stuck', 'centr', 'right', 'clown', 'right', 'joke
r', 'left', '...', ':)']
0       0.82347548      b'stuck centr right clown right joker left ... :)'
THE TWEET IS: @Sazzi91 we are following you now :) x
THE PROCESSED TWEET IS: ['follow', ':)', 'x']
0       0.84129184      b'follow :) x'
THE TWEET IS: @FindBenNeedham it's my birthday today so for my birthday wish
I hope there's good news about Ben soon :-)
THE PROCESSED TWEET IS: ['birthday', 'today', 'birthday', 'wish', 'hope', "t
here'", 'good', 'news', 'ben', 'soon', ':-)']
0       0.59266261      b"birthday today birthday wish hope there' good news
ben soon :-)"
THE TWEET IS: @LouiseR97054900 Happy Friday for you too :) @toonstra65 @_eme
raldeye_  @lisamarti76 @Dahlizma  @miss_steele89 @LouMWrites @ASeguda
THE PROCESSED TWEET IS: ['happi', 'friday', ':)']
0       0.84708628      b'happi friday :)'
THE TWEET IS: @EllieVond @SkeletonSweets @Justin_Naito @justcallmerizzo No a
ctually, you don't. Bye bye indeed. Go take your drama elsewhere. :)
THE PROCESSED TWEET IS: ['actual', 'bye', 'bye', 'inde', 'go', 'take', 'dram
a', 'elsewher', ':)']
0       0.82478326      b'actual bye bye inde go take drama elsewher :)'
THE TWEET IS: @Tim_A_Roberts @Pinter_Quotes works for me :)
THE PROCESSED TWEET IS: ['work', ':)']
0       0.82946022      b'work :)'
THE TWEET IS: @ninebonso04 thank you :)
THE PROCESSED TWEET IS: ['thank', ':)']
0       0.86046645      b'thank :)'
THE TWEET IS: Yoohoo! Shattering all records #BajrangiBhaijaanStorm #SuperHa
ppy :D http://t.co/wQSegYjWil
THE PROCESSED TWEET IS: ['yoohoo', 'shatter', 'record', 'bajrangibhaijaansto
rm', 'superhappi', ':d']
0       0.57011298      b'yoohoo shatter record bajrangibhaijaanstorm superh
appi :d'
THE TWEET IS: Happy Friday :-) http://t.co/B0l9zha8cl
THE PROCESSED TWEET IS: ['happi', 'friday', ':-)']
0       0.60297733      b'happi friday :-)'
THE TWEET IS: @charlesjonesss F off :)
THE PROCESSED TWEET IS: ['f', ':)']
0       0.83043373      b'f :)'
THE TWEET IS: @EJWoolf hi emma. :-) can I ask is your #BellyButton an #Innie
or an #Outie?
THE PROCESSED TWEET IS: ['hi', 'emma', ':-)', 'ask', 'bellybutton', 'inni',
'outi']
0       0.59107629      b'hi emma :-) ask bellybutton inni outi'
THE TWEET IS: @BarbieDevotees lyka followback :)
THE PROCESSED TWEET IS: ['lyka', 'followback', ':)']
0       0.83057924      b'lyka followback :)'
THE TWEET IS: @Gurmeetramrahim #OurDaughtersOurPride dhan dhan satguru tera
hi aasra...many congratulations Pita G...Keep them blessed as always :-)
```

```
THE PROCESSED TWEET IS: ['ourdaughtersourprid', 'dhan', 'dhan', 'satguru',
'tera', 'hi', 'aasra', '...', 'mani', 'congratul', 'pita', 'g', '...', 'kee
p', 'bless', 'alway', ':-)']
0       0.57820389      b'ourdaughtersourprid dhan dhan satguru tera hi aasr
a ... mani congratul pita g ... keep bless alway :-)'
THE TWEET IS: Keeo guessing whats behind that white cover :) special for yo
u.. happy bday once again my darling… https://t.co/TyFcTnM59u
THE PROCESSED TWEET IS: ['keeo', 'guess', 'what', 'behind', 'white', 'cove
r', ':)', 'special', '..', 'happi', 'bday', 'darl', '…']
0       0.84284836      b'keeo guess what behind white cover :) special .. h
appi bday darl '
THE TWEET IS: That awkward moment when your name is 'Akarshan', But you end
up staying 'Single'.
:D
#ForeverAlone
THE PROCESSED TWEET IS: ['awkward', 'moment', 'name', 'akarshan', 'end', 'st
ay', 'singl', ':d', 'foreveralon']
0       0.56639175      b'awkward moment name akarshan end stay singl :d for
everalon'
THE TWEET IS: @TomRPI Don't worry :)  I know how much stress you were under
and I had no problems with the site! I can't wait till the event!
THE PROCESSED TWEET IS: ['worri', ':)', 'know', 'much', 'stress', 'problem',
'site', "can't", 'wait', 'till', 'event']
0       0.82271447      b"worri :) know much stress problem site can't wait
till event"
THE TWEET IS: @V4Violetta *highfive* You are probably ahead of me there, sin
ce I am less artsy than verbal :D
THE PROCESSED TWEET IS: ['highfiv', 'probabl', 'ahead', 'sinc', 'less', 'art
si', 'verbal', ':d']
0       0.56712618      b'highfiv probabl ahead sinc less artsi verbal :d'
THE TWEET IS: @PARKCHAN92_bTH just kidding kaaaa :p
THE PROCESSED TWEET IS: ['kid', 'kaaa', ':p']
0       0.51362804      b'kid kaaa :p'
THE TWEET IS: @bookmyshow Wahoo Doing Team :-) #MasaanToday
THE PROCESSED TWEET IS: ['wahoo', 'team', ':-)', 'masaantoday']
0       0.57558917      b'wahoo team :-) masaantoday'
THE TWEET IS: DAT RP THO. thank you so much you guys for celebrating one mon
th of partnership with me!!! ty @MadMorphTV for the raid! :D
THE PROCESSED TWEET IS: ['dat', 'rp', 'tho', 'thank', 'much', 'guy', 'celeb
r', 'one', 'month', 'partnership', 'ty', 'raid', ':d']
0       0.61200951      b'dat rp tho thank much guy celebr one month partner
ship ty raid :d'
THE TWEET IS: @clarelea101 At least it's Friday :D
THE PROCESSED TWEET IS: ['least', 'friday', ':d']
0       0.58065913      b'least friday :d'
THE TWEET IS: @LemonyLimeUK thanks for the follow have a great day :)
THE PROCESSED TWEET IS: ['thank', 'follow', 'great', 'day', ':)']
0       0.87966991      b'thank follow great day :)'
THE TWEET IS: @readcreatelove Great to hear you had such a nice day, thanks
for visiting :)
THE PROCESSED TWEET IS: ['great', 'hear', 'nice', 'day', 'thank', 'visit',
':)']
0       0.87595099      b'great hear nice day thank visit :)'
THE TWEET IS: drama korea 49 days.:)
THE PROCESSED TWEET IS: ['drama', 'korea', '49', 'day', ':)']
0       0.83593749      b'drama korea 49 day :)'
```

```
ck soon'
THE TWEET IS: first time to go to school without my bracelets :(( it feels o
dd
THE PROCESSED TWEET IS: ['first', 'time', 'go', 'school', 'without', 'bracel
et', ':(', 'feel', 'odd']
1       0.10309226      b'first time go school without bracelet :( feel odd'
THE TWEET IS: @ayyedolans IM NOT UNTIL THE TWINS FOLLOW ME BACK BYLFNNZ :(
THE PROCESSED TWEET IS: ['im', 'twin', 'follow', 'back', 'bylfnnz', ':(']
1       0.11894136      b'im twin follow back bylfnnz :('
THE TWEET IS: @LucyAndLydia @georgiamerryyy I want one :(
THE PROCESSED TWEET IS: ['want', 'one', ':(']
1       0.10575212      b'want one :('
THE TWEET IS: Hmmm 10 mins to get my train and I'm currently about 15 mins a
way :( #failsatlife
THE PROCESSED TWEET IS: ['hmmm', '10', 'min', 'get', 'train', 'current', '1
5', 'min', 'away', ':(', 'failsatlif']
1       0.11224128      b'hmmm 10 min get train current 15 min away :( fails
atlif'
THE TWEET IS: I want it to be my birthday already :(
THE PROCESSED TWEET IS: ['want', 'birthday', 'alreadi', ':(']
1       0.10797903      b'want birthday alreadi :('
THE TWEET IS: Im super duper tired :(
THE PROCESSED TWEET IS: ['im', 'super', 'duper', 'tire', ':(']
1       0.10868714      b'im super duper tire :('
THE TWEET IS: ill be on soon, I PROMISE :(
waaah
THE PROCESSED TWEET IS: ['ill', 'soon', 'promis', ':(', 'waaah']
1       0.11323199      b'ill soon promis :( waaah'
THE TWEET IS: MY PUPPY BROKE HER FOOT :(
THE PROCESSED TWEET IS: ['puppi', 'broke', 'foot', ':(']
1       0.11321671      b'puppi broke foot :('
THE TWEET IS: But but Mr Ahmad Maslan cooks too :( https://t.co/ArCiD31Zv6
THE PROCESSED TWEET IS: ['mr', 'ahmad', 'maslan', 'cook', ':(']
1       0.11354535      b'mr ahmad maslan cook :('
```

In [26]:

```python
my_tweet = 'This is a ridiculously bright movie. The plot was terrible and I

print(process_tweet(my_tweet))
y_hat = predict_tweet(my_tweet, freqs, theta)

print(y_hat)
if y_hat > 0.5:
    print('Positive sentiment')
else:
    print('Negative sentiment')
```

```
['ridicul', 'bright', 'movi', 'plot', 'terribl', 'sad', 'end']
[[0.48122777]]
Negative sentiment
```

Visualizing Naive Bayes

In [28]:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import re
import string
import nltk

from nltk.corpus import stopwords, twitter_samples
from nltk.tokenize import TweetTokenizer
from nltk.stem import PorterStemmer
from nltk.tokenize import TweetTokenizer
from os import getcwd
```

In [29]:
```
def process_tweet(tweet):
    stemmer = PorterStemmer()
    stopwords_english = stopwords.words('english')
    # Remove stock market tickers like SGE
    tweet = re.sub(r'\$\w*', '', tweet)

    # Remove old style retweet text "RT"
    tweet = re.sub(r'^RT[\s]+', '', tweet)

    # Remove hyperlinks
    tweet = re.sub(r'https?://[^\s\n\r]+', '', tweet)

    # Remove hashtags
    # Only removing the hash # sign from the word
    tweet = re.sub(r'#', '', tweet)

    # Tokenize tweets
    tokenizer = TweetTokenizer(preserve_case=False, strip_handles=True, redu
    tweet_tokens = tokenizer.tokenize(tweet)
    tweets_clean = []

    for word in tweet_tokens:
        if word not in stopwords_english and  word not in string.punctuatior
            stem_word = stemmer.stem(word) # Stemming words
            tweets_clean.append(stem_word)

    return tweets_clean
```

In [31]:
```
def lookup(freqs, word, label):
    n = 0
    pair = (word, label)

    if pair in freqs:
        n = freqs[pair]
    return n
```

In [32]:
```
def test_lookup(func):
    freqs = {('sad', 0): 4,
             ('happy', 1): 12,
             ('opperessed',0):7}
    word='happy'
    label=1

    if func(freqs, word, label) == 12:
        return 'S'
    else:
        return False

print(test_lookup(lookup))
```

S

In [33]:
```
data = pd.read_csv('bayes_features.csv')
data.head(5)
```

Out[33]:

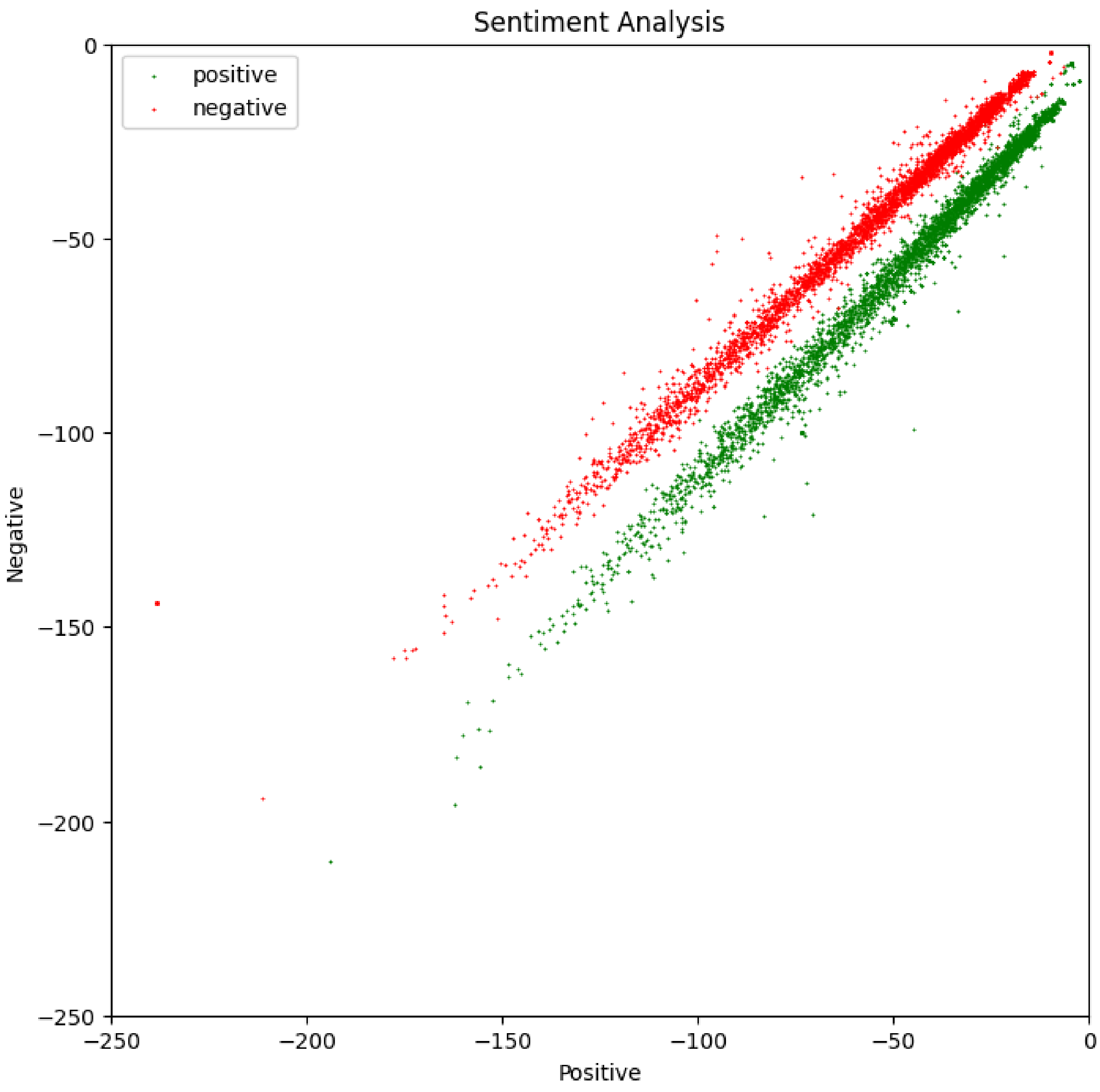
	positive	negative	sentiment
0	-45.763393	-63.351354	1.0
1	-105.491568	-114.204862	1.0
2	-57.028078	-67.216467	1.0
3	-10.055885	-18.589057	1.0
4	-125.749270	-138.334845	1.0

In [34]:
```
fig, ax = plt.subplots(figsize = (8,8))
colors = ['red','green']
sentiments = ['negative','positive']
index = data.index

for sentiment in data.sentiment.unique():
    ix = index[data.sentiment == sentiment]
    ax.scatter(data.iloc[ix].positive, data.iloc[ix].negative, c=colors[int(

ax.legend(loc='best')

plt.xlim(-250,0)
plt.ylim(-250,0)
plt.xlabel('Positive')
plt.ylabel('Negative')
plt.title('Sentiment Analysis')
plt.show()
```


Sentiment Analysis
positive
negative
Negative
Positive
0
−50
−100
−150
−200
−250
−250
−200
−150
−100
−50
0

```
In [ ]:
```