PROJECT DOCUMENTATION ON ARTIFICIAL INTELLIGENCE

TOPIC: SENTIMENTAL ANALYSIS OF TWITTER DATA - USING DEEP LEARNING

PROJECT DONE BY:

S.BALA SUJITH

INDEX

TWITTER SENTIMENTAL ANALYSIS USING DEEP LEARNING

INTRODUCTION:

Overview

Sentiment analysis is the automated process of identifying and extracting the
subjective information that underlies a text. This can be either an opinion, a judgment,
or a feeling about a particular topic or subject. The most common type of sentiment
analysis is called ‘polarity detection’ and involves classifying a statement as ‘positive’,
‘negative’, or ‘neutral’. Twitter has grown in popularity during the past decades. It is now
used by millions of users who share information about their daily life and their feelings.
In order to automatically process and analyze these data, applications can rely on
analysis methods such as sentiment analysis and topic modeling. Developing a
program for sentiment analysis is an approach to be used to computationally measure
customers' perceptions.Twitter has been growing in popularity and nowadays, it is used
every day by people to express opinions about different topics, such as products,

movies, music, politicians, events, social events, among others. Twitter sentiment
classification aims to classify the sentiment polarity of a tweet as positive, negative or
neutral. In this project, we develop a deep learning system for Twitter sentiment
classification.

Purpose

Nearly 80% of the world's digital data is unstructured, and a large portion of that
includes social media data. Sentiment analysis tools use artificial intelligence and
natural language processing (NLP) to organize unstructured text data automatically.
Sentiment analysis algorithms are able to learn from data samples to detect the polarity
of Tweets in real-time. • **Business:** In marketing field companies use it to develop their strategies, to
understand customers' feelings towards products or brand, how people respond to
their campaigns or product launches and why consumers don't buy some
products.

• **Politics:** In political field, it is used to keep track of political view, to detect
consistency and inconsistency between statements and actions at the government

level. It can be used to predict election results as well!
• **Public Actions:** Sentiment analysis also is used to monitor and analyse social
phenomena, for the spotting of potentially dangerous situations and determining
the general mood of the blogosphere.

LITERATURE SURVEY:

This section summarizes some of the scholarly and the research works in the
field of deep learning to analyse sentiments on the Twitter and preparing prediction
model for various applications. As the available social platforms are shooting up, the
information is becoming vast and can be extracted to turn into business objectives
,social campaigns, marketing and other promotional strategies. The benefit of social
media to know public opinions and extract their emotions are considered.

Existing Problem:

Every day massive amount of data is being generated by social media users which can
be used to analyze their opinion about any event, movie, product or politics. Thus in
order to trace it, an automated process of analysing text data and sorting into positive,
negative or neutral is proposed.

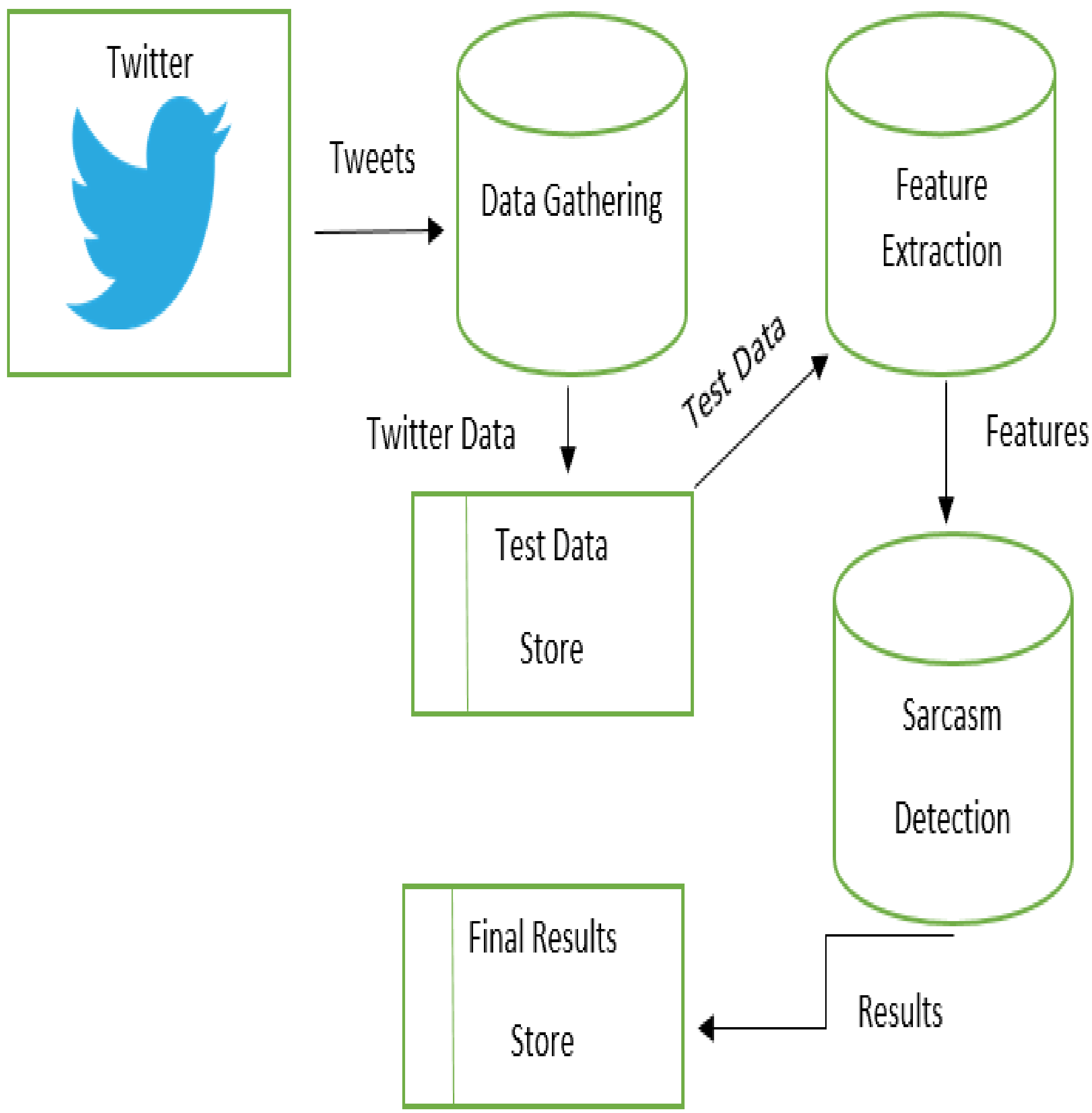
Different Classes of Sentiment Analysis

a. Positive Sentiments: These are the good words about the target in consideration. If the positive sentiments are increased, it is referred to be good. In case of product reviews, if the positive reviews about the product are more, it is bought by many customers. **b. Negative Sentiments**: These are the bad words about the target in consideration. If the negative sentiments are increased, it is discarded from the preference list. In case of product reviews, if the negative reviews about the product are more, no one intend to buy it.

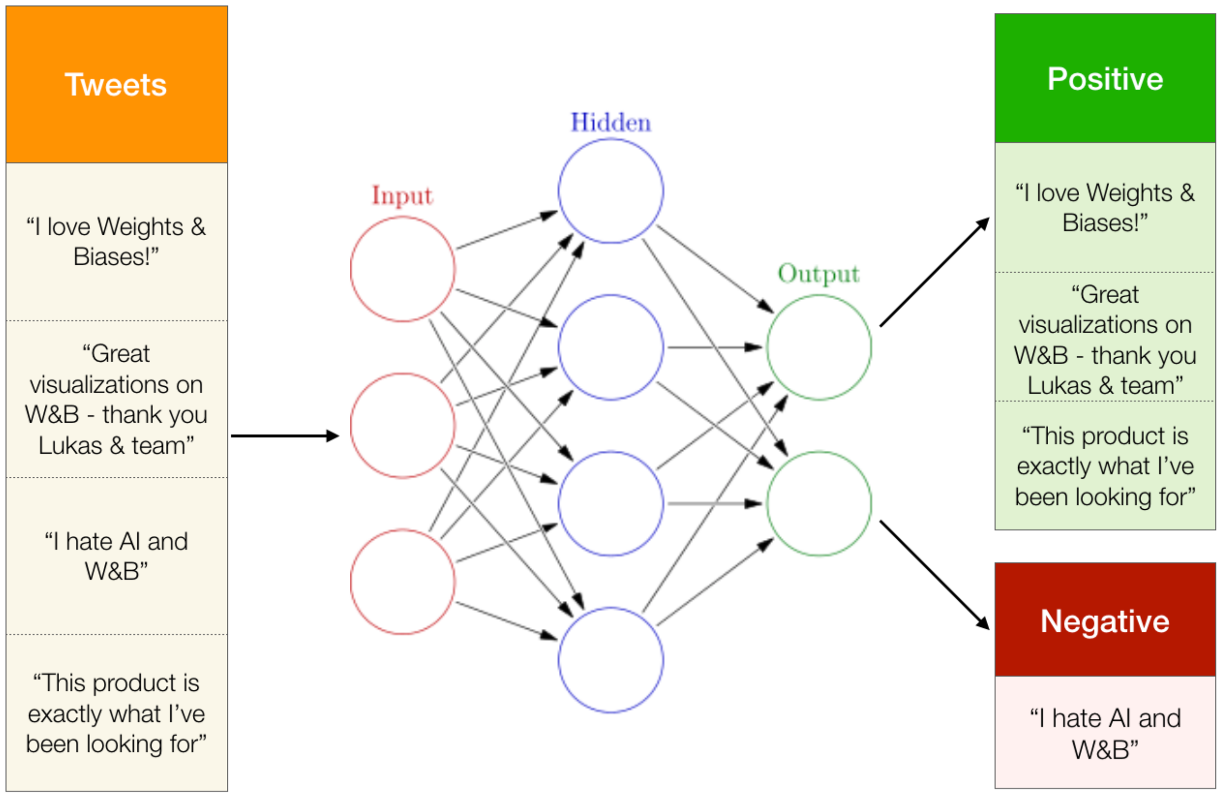
c. Neutral Sentiments: These are neither good nor bad words about the target. Hence it is neither preferred nor neglected.

THEORITICAL ANALYSIS:

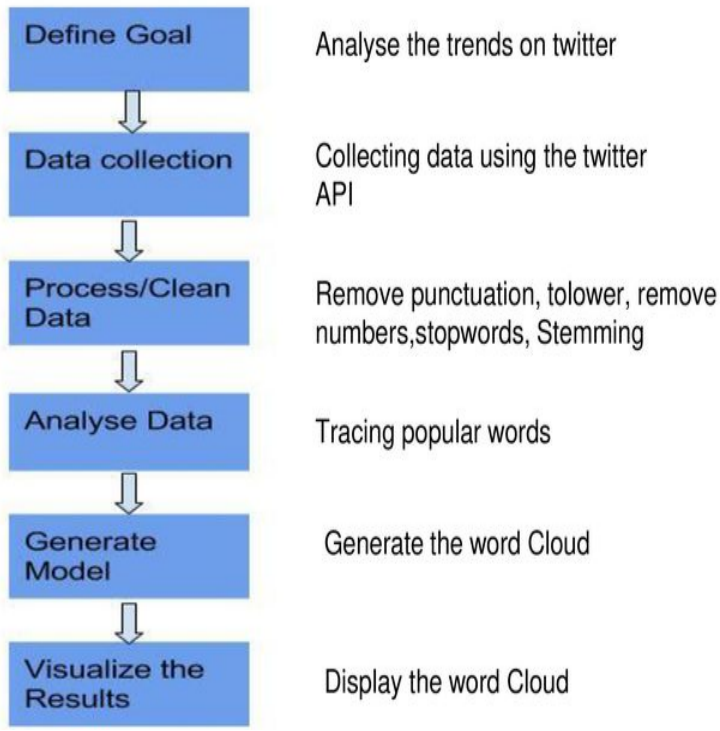
Block Diagram:



Twitter
Tweets
Data Gathering
Feature
Extraction
Test Data
Twitter Data
Features
Test Data
Store
Sarcasm
Detection
Final Results
Store
Results


Tweets
"I love Weights & Biases!"
"Great visualizations on W&B - thank you Lukas & team"
"I hate AI and W&B"
"This product is exactly what I've been looking for"
Input
Hidden
Output
Positive
"I love Weights & Biases!"
"Great visualizations on W&B - thank you Lukas & team"
"This product is exactly what I've been looking for"
Negative
"I hate AI and W&B"

Flowchart



EXPERIMENTAL INVESTIGATION:

Text Processing

Steps

1.Gathering data

2.Import the Dataset

3.Text Cleaning or Pre-processing

- Remove Punctuations, Numbers
- Convert each word into its lower case
- Stemming
- Splitting Data into Training and Test set

1.Gathering data

The data set which we have taken is Twitter Sentiment Analysis. The train set
consists of three columns namely ID, Tweet Statement and category.

2.Import the libraries

The dataset train.csv is imported using Pandas Library.Various libraries such as
numpy and matplotlib.pyplot

Sentimental Analysis of twitter data using deep learning

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
```

Importing Datasets

```
In [2]: train = pd.read_csv('dataset/train.csv', encoding='latin-1')
        test = pd.read_csv('dataset/test.csv', encoding='latin-1')
```

```
In [3]: train.head()
```

Out[3]:

	ItemID	Sentiment	SentimentText
0	1	0	is so sad for my APL frie...
1	2	0	I missed the New Moon trail...
2	3	1	omg its already 7:30 :O
3	4	0	.. Omgaga. Im sooo im gunna CRy. I'...
4	5	0	i think mi bf is cheating on me!!! ...

3.Text Cleaning or Pre-processing

“Re” is the library which is used to replace the selected special characters with
desired parameter. “NLTK” – Natural language Tool Kit is the library used for stemming
using a special class in the library.

```
In [4]: train.shape
Out[4]: (99989, 3)

In [5]: test.shape
Out[5]: (299989, 2)

In [6]: import re
        import nltk
        nltk.download('stopwords')
        from nltk.corpus import stopwords
        from nltk.stem.porter import PorterStemmer

        [nltk_data] Downloading package stopwords to
        [nltk_data]     C:\Users\hp\AppData\Roaming\nltk_data...
        [nltk_data]   Package stopwords is already up-to-date!

In [7]: sentiment = train["SentimentText"][0]
        sentiment = str(re.sub('[^a-zA-Z]', ' ', sentiment)).lower().strip()

In [8]: sentiment
Out[8]: 'is so sad for my apl friend'

In [9]: ps = PorterStemmer()
        sentiment = [ps.stem(word) for word in sentiment if not word in set(stopwords.words('english'))]

In [10]: sentiment
Out[10]: [' ', ' ', ' ', 'f', 'r', ' ', ' ', 'p', 'l', ' ', 'f', 'r', 'e', 'n']
```

Remove Punctuations, Numbers

Punctuations, Numbers doesn't help much in processing the given text, so we will be using **re** library to replace all the punctuations numbers with a space while excluding alphabets. As in the dataset the reviews are present in **Tweet** column, we are declaring a variable called tweet and assigning the second row of the column to declared variable. Then using **re** library we are substituting all the other special characters with a space excluding alphabets.

Convert each word into its lower case

Every word in the taken tweet should be lower cased, because if we have a word in different cases the machine will think both are different words.

Stemming

```
In [11]: data = []
         for i in range(train.shape[0]):
           sentiment = train["SentimentText"][i]
           sentiment = str(re.sub('[^a-zA-Z]', ' ', sentiment)).lower().strip().split()
           sentiment = [ps.stem(word) for word in sentiment if not word in set(stopwords.words('english'))]
           sentiment = ' '.join(sentiment)
           data.append(sentiment)
```

```
In [12]: data[:20]

Out[12]: ['sad apl friend',
          'miss new moon trailer',
          'omg alreadi',
          'omgaga im sooo im gunna cri dentist sinc supos get crown put min',
          'think mi bf cheat',
          'worri much',
          'juuuuuuuuuuuuuuuuussssst chillin',
          'sunni work tomorrow tv tonight',
          'hand uniform today miss alreadi',
          'hmmmm wonder number',
          'must think posit',
          'thank hater face day',
          'weekend suck far',
          'jb isnt show australia',
          'ok that win',
          'lt way feel right',
          'awhh man complet useless rt funni twitter http myloc hx',
          'feel strang fine gonna go listen semison celebr',
          'huge roll thunder scari',
          'cut beard grow well year gonna start shaunamanu happi meantim']
```

Stemming is the process of producing morphological variants of a root/base word.

Stemming programs are commonly referred to as stemming algorithms or stemmers.

Importing Count vectorizer:

```
In [13]: from sklearn.feature_extraction.text import CountVectorizer
         cv = CountVectorizer(max_features = 1000, stop_words='english')
         cv.fit(data)
         X = cv.transform(data).toarray()
```

```
In [14]: X.shape

Out[14]: (99989, 1000)
```

```
In [15]: cv.vocabulary_

Out[15]: {'sad': 726,
          'friend': 353,
          'miss': 570,
          'new': 596,
          'moon': 579,
          'omg': 619,
          'alreadi': 29,
          'im': 449,
          'sooo': 798,
          'cri': 227,
          'sinc': 775,
          'min': 567,
          'think': 861,
          'worri': 970,
          'sunni': 837,
          'work': 968,
          'tomorrow': 876,
          'tv': 898,
          'tonight': 877,
          'hand': 397,
```

Splitting Data into Training and Test set

For this, we need class train_test_split from

sklearn.cross_validation. Split can be
made 70/30 or 80/20 or 85/15 or 75/25, here we choose 80/20 via “test_size”. X is the
bag of words; y is 0 or 1 (positive or negative).**Model Building**

```
In [16]: y = train.iloc[:, 1].values

In [17]: y.shape
Out[17]: (99989,)

In [18]: from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 7)

In [19]: print("X_train : ", X_train.shape, "\nX_test : ", X_test.shape, "\ny_train : ", y_train.shape, "\ny_test : ", y_test.shape)

         X_train :  (79991, 1000)
         X_test :  (19998, 1000)
         y_train :  (79991,)
         y_test :  (19998,)
```

Importing Libraries:

The first step is to define the functions and classes we intend to use in this. We
will use two classes from the Keras library to define our model.

Model Building

```
In [22]: import tensorflow
         import tensorflow.keras
         from tensorflow.keras.models import Sequential
         from tensorflow.keras.layers import Dense

In [23]: model = Sequential()

In [27]: model.add(Dense(units = 128, kernel_initializer = "uniform", activation = 'relu'))
         model.add(Dense(units = 64, kernel_initializer = 'uniform', activation = 'relu'))
         model.add(Dense(units = 64, kernel_initializer = 'uniform', activation = 'relu'))
         model.add(Dense(units = 1, kernel_initializer = 'uniform', activation = 'sigmoid'))

In [28]: model.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])

In [29]: model.fit(X_train, y_train, epochs = 100, batch_size = 100)
```

Initializing the model :

Keras has 2 ways to define a neural network**:**

- Sequential
- Function API

We will use the **Sequential** constructor to create a model, which will then have
layers added to it using the **add()** method.

Adding Input layer:
This step is to add a dense layer (input layer) where you will be specifying the
number of inputs to the neural network, activation function and weights initializer and
number of connection to the hidden layer as the arguments. We use add() method to
add dense layers.**Adding Hidden layer:**
This step is to add a dense layer (Hidden layer) where you will be specifying the
number neurons to the next layer, activation function and weight initializer as the
arguments.
Adding an Output Layer :
This step is to add a dense layer (output layer) where you will be specifying the
number of classes your dependent variable has, activation function and weight initializer
as the arguments.
Configuring the learning process:
Compilation requires 3 arguments: an optimizer, a loss function, and a list of
metrics.

```
800/800 [==============================] - 5s 6ms/step - loss: 0.0957 - accuracy: 0.9474
Epoch 93/100
800/800 [==============================] - 5s 6ms/step - loss: 0.0949 - accuracy: 0.9480
Epoch 94/100
800/800 [==============================] - 5s 6ms/step - loss: 0.0955 - accuracy: 0.9475
Epoch 95/100
800/800 [==============================] - 5s 6ms/step - loss: 0.0985 - accuracy: 0.9468
Epoch 96/100
800/800 [==============================] - 5s 6ms/step - loss: 0.0978 - accuracy: 0.9473
Epoch 97/100
800/800 [==============================] - 5s 6ms/step - loss: 0.0946 - accuracy: 0.9478: 0s - loss: 0.0947 -
Epoch 98/100
800/800 [==============================] - 5s 7ms/step - loss: 0.0941 - accuracy: 0.9482
Epoch 99/100
800/800 [==============================] - 5s 6ms/step - loss: 0.0944 - accuracy: 0.9480
Epoch 100/100
800/800 [==============================] - ETA: 0s - loss: 0.0964 - accuracy: 0.94 - 5s 6ms/step - loss: 0.0964 - accuracy:
0.9476

Out[29]: <tensorflow.python.keras.callbacks.History at 0x20b10c76eb0>
```

Training the model and saving the model :

Training begins by calling the **fit()** method. The arguments are batch size as you
are using "**adam**" (bath gradient descent and epochs: no: of times the model should get
trained.

Saving the model

```
In [30]: import pickle
         pickle.dump(cv, open('cv.pkl', 'wb'))

In [31]: model.save('model.h5')
```

Making Predictions:

Making predictions

```
In [32]: predictions = [ 1 if x > 0.5 else 0 for x in model.predict(X_test)]

In [33]: predictions[:20]

Out[33]: [1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0]

In [34]: from sklearn.metrics import confusion_matrix, classification_report

In [35]: print(confusion_matrix(y_test, predictions))

         [[5262 3550]
          [2776 8410]]

In [36]: print(classification_report(y_test, predictions))

                       precision    recall  f1-score   support

                   0       0.65      0.60      0.62      8812
                   1       0.70      0.75      0.73     11186

            accuracy                           0.68     19998
           macro avg       0.68      0.67      0.68     19998
        weighted avg       0.68      0.68      0.68     19998
```

Building HTML Page:

This is the basic HTML page for our Project. H1 tag is used to give heading to
the project. user has to enter the tweet , so we have to add 1 (one) text input fields in
the web page.A button is used to send these values to the model files this functionality
will be written in the python file app.py. the model predicts the value and is displayed on
the {{ **y_pred** }}field and respected emoji will be displayed.

FILE STRUCTURE:

PYTHON CODE:

app.py file:

```python

import os
from flask import Flask, request, jsonify, render_template
from brain import brain

app = Flask(__name__)
@app.route('/')
def home():
    err = "Saved Model Doesn't Exist"
    if os.path.isfile('./model.h5'): err = ''
    return render_template('sentimental analysis.html', result = 'https://i.pinimg.com/or
@app.route('/',methods=['POST'])
def y_predict():
    sentiment = request.form["Message"]
    err, res = str(brain(sentiment)), ''
    if err == '0' or err == '1' :
        res, err = err, res
    if res=='1':
            return render_template('sentimental analysis.html', result = 'https://i.pini
    if res=='0':
             return render_template('sentimental analysis.html', result = 'https://image
if __name__=="__main__":
    app.run(debug = True)


```

brain.py file:

```python
import os

import pickle
#import scipy as sp
#import numpy as np
from tensorflow.keras.models import load_model

def brain(x):
    if not os.path.isfile('./model.h5'): return "Unable to find Saved Model"
    model = load_model('model.h5')
    with open('cv.pkl', 'rb') as file:
        cv = pickle.load(file)
        x = cv.transform([x])

        pred = model.predict(x)
        if(pred > 0.5):
            return 1
        else:
            return 0

```

Running flask web app on local server using command line:

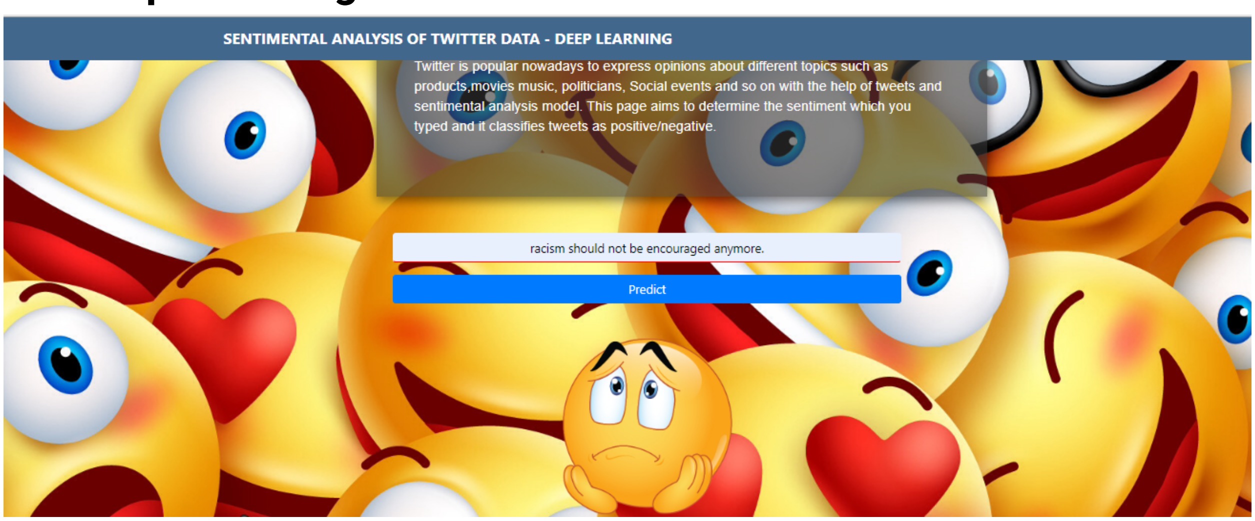
```
 * Debugger is active!
 * Debugger PIN: 126-892-279
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

WEBPAGE:

Training Data

1.Example for Postitive Sentiment:

2.Example for Negative Sentiment:



RESULT:

Sentiment Analysis is an interesting way to think about the applicability of Natural
Language Processing in making automated conclusions about text. It is being utilized in
social media trend analysis and, sometimes, for marketing purposes. The results from
sentiment analysis help businesses understand the conversations and discussions
taking place about them.They can quickly identify any negative sentiments being
expressed and turn poor customer experiences into good ones.

ADVANTAGES:

- Sentiment analysis is a useful tool for any organization or group for which public
sentiment or attitude towards them is important for their success - whichever way
that success is defined.
- The results from sentiment analysis help businesses understand the
conversations and discussions taking place about them, and helps them react
and take action accordingly.
- By listening to and analysing comments on Facebook and Twitter, local
government departments can gauge public sentiment towards their department
and the services they provide, and use the results to improve

services such as
parking and leisure facilities, local policing, and the condition of roads.

DISADVANTAGES:

- Misspellings and grammatical mistakes may cause the analysis to overlook important words or usage.
- Sarcasm and irony may be misinterpreted.Analysis is language-specific.
- Discriminating jargon, nomenclature, memes, or turns of phrase may not be recognized.
- Sentiment analysis tools can identify and analyse many pieces of text automatically and quickly. But computer programs recognizing things –the sort of things a person would have little trouble identifying. So sentiment analysis tool do a really great job of analysing text for opinion and attitude ,but they are not perfect.**APPLICATIONS:**

- Social media monitoring
- Customer Experience Management and voice of customer
- People analytics

CONCLUSION

We develop a deep learning system for message-level Twitter sentiment

classification in this project. The effectiveness of this project has been verified in both
positive/negative/neutral classification of tweets. **Sentiment analysis** allows businesses
to identify customer **sentiment** toward products, brands or services in online
conversations and feedback. Thus this tool can be used in a wide range of applications.

FUTURE SCOPE:

Sentiment analysis is already evolving rapidly from a very simple (positive,
negative, neutral) to more granular and deep understanding. These new classifier really
dimensionalize the nuances of human expression in meaningful ways. There is also a
move away from document/record level analysis of the text towards entity/facet level -
meaning every expression of opinion is captured so that we can really understand the
root cause drivers of opinions. This requires machine learning approaches that are
superceding more traditional rules based approaches.