Sentiment Analysis for Twitter Data

June 6, 2020

1 Getting Ready for Analysis

Run the below cell to install all the required packages for this notebook.

[1]:
```
!pip3 install -r requirements.txt
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: emoji>=0.5.4 in /usr/local/lib/python3.6/dist-
packages (from -r requirements.txt (line 1)) (0.5.4)
Requirement already satisfied: numpy>=1.17.1 in /usr/local/lib/python3.6/dist-
packages (from -r requirements.txt (line 2)) (1.17.1)
Requirement already satisfied: pandas>=0.25.3 in /usr/local/lib/python3.6/dist-
packages (from -r requirements.txt (line 3)) (0.25.3)
Requirement already satisfied: textblob>=0.15.3 in
/usr/local/lib/python3.6/dist-packages (from -r requirements.txt (line 4))
(0.15.3)
Requirement already satisfied: scikit-learn>=0.22.1 in
/usr/local/lib/python3.6/dist-packages (from -r requirements.txt (line 5))
(0.22.1)
Requirement already satisfied: vaderSentiment>=3.3.2 in
/usr/local/lib/python3.6/dist-packages (from -r requirements.txt (line 6))
(3.3.2)
Requirement already satisfied: matplotlib>=3.1.2 in
/usr/local/lib/python3.6/dist-packages (from -r requirements.txt (line 7))
(3.1.2)
Requirement already satisfied: nltk>=3.4.5 in
/home/anurag/.local/lib/python3.6/site-packages (from -r requirements.txt (line
8)) (3.4.5)
Requirement already satisfied: wordcloud>=1.7.0 in
/usr/local/lib/python3.6/dist-packages (from -r requirements.txt (line 9))
(1.7.0)
Requirement already satisfied: python-dateutil>=2.6.1 in
/home/anurag/.local/lib/python3.6/site-packages (from pandas>=0.25.3->-r
requirements.txt (line 3)) (2.8.1)
Requirement already satisfied: pytz>=2017.2 in
/home/anurag/.local/lib/python3.6/site-packages (from pandas>=0.25.3->-r
requirements.txt (line 3)) (2019.3)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.6/dist-
```

```
packages (from scikit-learn>=0.22.1->-r requirements.txt (line 5)) (0.14.1)
Requirement already satisfied: scipy>=0.17.0 in
/home/anurag/.local/lib/python3.6/site-packages (from scikit-learn>=0.22.1->-r
requirements.txt (line 5)) (1.4.1)
Requirement already satisfied: requests in
/home/anurag/.local/lib/python3.6/site-packages (from vaderSentiment>=3.3.2->-r
requirements.txt (line 6)) (2.22.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
/home/anurag/.local/lib/python3.6/site-packages (from matplotlib>=3.1.2->-r
requirements.txt (line 7)) (1.1.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in
/home/anurag/.local/lib/python3.6/site-packages (from matplotlib>=3.1.2->-r
requirements.txt (line 7)) (2.4.5)
Requirement already satisfied: cycler>=0.10 in
/home/anurag/.local/lib/python3.6/site-packages (from matplotlib>=3.1.2->-r
requirements.txt (line 7)) (0.10.0)
Requirement already satisfied: six in /home/anurag/.local/lib/python3.6/site-
packages (from nltk>=3.4.5->-r requirements.txt (line 8)) (1.13.0)
Requirement already satisfied: pillow in /usr/local/lib/python3.6/dist-packages
(from wordcloud>=1.7.0->-r requirements.txt (line 9)) (6.2.0)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
/home/anurag/.local/lib/python3.6/site-packages (from
requests->vaderSentiment>=3.3.2->-r requirements.txt (line 6)) (1.24.3)
Requirement already satisfied: idna<2.9,>=2.5 in
/home/anurag/.local/lib/python3.6/site-packages (from
requests->vaderSentiment>=3.3.2->-r requirements.txt (line 6)) (2.8)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in
/home/anurag/.local/lib/python3.6/site-packages (from
requests->vaderSentiment>=3.3.2->-r requirements.txt (line 6)) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in
/home/anurag/.local/lib/python3.6/site-packages (from
requests->vaderSentiment>=3.3.2->-r requirements.txt (line 6)) (2019.11.28)
Requirement already satisfied: setuptools in /usr/local/lib/python3.6/dist-
packages (from kiwisolver>=1.0.1->matplotlib>=3.1.2->-r requirements.txt (line
7)) (41.2.0)
WARNING: You are using pip version 20.1; however, version 20.1.1 is
available.
You should consider upgrading via the '/usr/bin/python3 -m pip install --upgrade
pip' command.
```

2 Sentiment Analysis

I used two approaches in predicting the sentiment of the given data.

- Lexicon based Sentiment Method

- Machine Learning based Sentiment Method

In the lexicon-based method, I have used two third-party libraries to predict the sentiments.

- TextBlob
- vaderSentiment

Both of these methods used a pre-defined valance score of the words to predict tweets' sentiment.

Coming to Machine Learning based prediction, we cannot use any traditional method of training the model and predicting on test data.

But we are provided with the unlabelled data. So, we cannot carry out the supervised learning strategy

So, I have used the IMDB Movie review dataset to train a neural network model and then use that model to predict the given tweets data.

This gave a very similar result compared to the lexicon-based methods.

Why these modules?

- *re:* It is used in preprocessing the data.
- *emoji:* To remove all the emoji's (part of preprocessing step)
- *numpy:* For doing some numerical calculations
- *pandas:* For loading the data into dataframe's
- *sklearn:* Splitting the dataset into train and test sets
- *tensorflow:* To build the machine learning
- *matplotlib:* For visualization of graphs and plots
- *textblob:* To compute lexicon based sentiment(method-1)
- *vaderSentiment:* To compute lexicon based sentiment(method-2)

[2]:
```
# Required modules

import re
import emoji
import numpy as np
import pandas as pd
import tensorflow as tf
from matplotlib import pyplot as plt

from textblob import TextBlob
from sklearn.model_selection import train_test_split
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

%matplotlib inline
plt.rcParams['figure.figsize'] = (12, 7)
```

```
/usr/local/lib/python3.6/dist-
packages/tensorflow/python/framework/dtypes.py:523: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/usr/local/lib/python3.6/dist-
packages/tensorflow/python/framework/dtypes.py:524: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/usr/local/lib/python3.6/dist-
packages/tensorflow/python/framework/dtypes.py:525: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/usr/local/lib/python3.6/dist-
packages/tensorflow/python/framework/dtypes.py:526: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
/usr/local/lib/python3.6/dist-
packages/tensorflow/python/framework/dtypes.py:527: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
/usr/local/lib/python3.6/dist-
packages/tensorflow/python/framework/dtypes.py:532: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
/usr/local/lib/python3.6/dist-
packages/tensorboard/compat/tensorflow_stub/dtypes.py:541: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/usr/local/lib/python3.6/dist-
packages/tensorboard/compat/tensorflow_stub/dtypes.py:542: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/usr/local/lib/python3.6/dist-
packages/tensorboard/compat/tensorflow_stub/dtypes.py:543: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/usr/local/lib/python3.6/dist-
packages/tensorboard/compat/tensorflow_stub/dtypes.py:544: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
```

```
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
/usr/local/lib/python3.6/dist-
packages/tensorboard/compat/tensorflow_stub/dtypes.py:545: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
/usr/local/lib/python3.6/dist-
packages/tensorboard/compat/tensorflow_stub/dtypes.py:550: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
```

[3]:

```
# Loading the data

data = pd.read_csv('./data/tweets.zip', compression='zip')
data.head()
```

[3]:

```
                    id      conversation_id     created_at        date  \
0  1262787913311387649  1262787913311387649  1589907074000  2020-05-19
1  1262787786152620040  1262787786152620040  1589907044000  2020-05-19
2  1262787219498000384  1262787219498000384  1589906909000  2020-05-19
3  1262786998592434176  1262786998592434176  1589906856000  2020-05-19
4  1262786970163441669  1262786970163441669  1589906849000  2020-05-19

       time timezone              user_id         username  \
0  16:51:14      UTC  1250079805980045318       dramaflick
1  16:50:44      UTC   807843238648299520       knowpuneet
2  16:48:29      UTC  1085426639570235392  narasinhpurohit
3  16:47:36      UTC  1104213868467806208        ka_trolls
4  16:47:29      UTC            392180204   rajendrabohora

                name place  … geo source user_rt_id user_rt  retweet_id  \
0     The Drama Flick   NaN  … NaN    NaN        NaN     NaN         NaN
1       TravelTrainee   NaN  … NaN    NaN        NaN     NaN         NaN
2    Narasinh Purohit   NaN  … NaN    NaN        NaN     NaN         NaN
3  Humans Of Hindutva   NaN  … NaN    NaN        NaN     NaN         NaN
4      rajendrabohora   NaN  … NaN    NaN        NaN     NaN         NaN

                                            reply_to  retweet_date translate  \
0  [{'user_id': '1250079805980045318', 'username'…          NaN       NaN
1  [{'user_id': '807843238648299520', 'username':…          NaN       NaN
2  [{'user_id': '1085426639570235392', 'username'…          NaN       NaN
3  [{'user_id': '1104213868467806208', 'username'…          NaN       NaN
4  [{'user_id': '392180204', 'username': 'rajendr…          NaN       NaN

  trans_src trans_dest
```

```
0       NaN        NaN
1       NaN        NaN
2       NaN        NaN
3       NaN        NaN
4       NaN        NaN

[5 rows x 34 columns]
```

[4]:

```python
# Inspecting the data

data.info()
data.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 124384 entries, 0 to 124383
Data columns (total 34 columns):
id                 124384 non-null int64
conversation_id    124384 non-null int64
created_at         124384 non-null int64
date               124384 non-null object
time               124384 non-null object
timezone           124384 non-null object
user_id            124384 non-null int64
username           124384 non-null object
name               124384 non-null object
place              9143 non-null object
tweet              124384 non-null object
mentions           124384 non-null object
urls               124384 non-null object
photos             124384 non-null object
replies_count      124384 non-null int64
retweets_count     124384 non-null int64
likes_count        124384 non-null int64
hashtags           124384 non-null object
cashtags           124384 non-null object
link               124384 non-null object
retweet            124384 non-null bool
quote_url          13033 non-null object
video              124384 non-null int64
near               0 non-null float64
geo                0 non-null float64
source             0 non-null float64
user_rt_id         0 non-null float64
user_rt            0 non-null float64
retweet_id         0 non-null float64
reply_to           124384 non-null object
retweet_date       0 non-null float64
```

```
translate          0 non-null float64
trans_src          0 non-null float64
trans_dest         0 non-null float64
dtypes: bool(1), float64(10), int64(8), object(15)
memory usage: 31.4+ MB
```

[4]:

```
                 id  conversation_id    created_at       user_id  \
count  1.243840e+05     1.243840e+05  1.243840e+05  1.243840e+05
mean   1.248684e+18     1.248494e+18  1.586544e+12  4.351165e+17
std    2.229079e+16     2.304835e+16  5.369331e+09  5.240636e+17
min    8.660923e+08     8.660923e+08  1.216819e+12  3.160000e+03
25%    1.243007e+18     1.242870e+18  1.585191e+12  2.617597e+08
50%    1.247014e+18     1.246851e+18  1.586146e+12  2.940158e+09
75%    1.256459e+18     1.256427e+18  1.588398e+12  9.912536e+17
max    1.263035e+18     1.263035e+18  1.589966e+12  1.262808e+18

       replies_count  retweets_count    likes_count          video  near  geo  \
count  124384.000000   124384.000000  124384.000000  124384.000000   0.0  0.0
mean        0.667626        2.077044       7.881858       0.061953   NaN  NaN
std        11.845734       28.424693     121.299772       0.241072   NaN  NaN
min         0.000000        0.000000       0.000000       0.000000   NaN  NaN
25%         0.000000        0.000000       0.000000       0.000000   NaN  NaN
50%         0.000000        0.000000       1.000000       0.000000   NaN  NaN
75%         0.000000        0.000000       2.000000       0.000000   NaN  NaN
max      2044.000000     3710.000000   19929.000000       1.000000   NaN  NaN

       source  user_rt_id  user_rt  retweet_id  retweet_date  translate  \
count     0.0         0.0      0.0         0.0           0.0        0.0
mean      NaN         NaN      NaN         NaN           NaN        NaN
std       NaN         NaN      NaN         NaN           NaN        NaN
min       NaN         NaN      NaN         NaN           NaN        NaN
25%       NaN         NaN      NaN         NaN           NaN        NaN
50%       NaN         NaN      NaN         NaN           NaN        NaN
75%       NaN         NaN      NaN         NaN           NaN        NaN
max       NaN         NaN      NaN         NaN           NaN        NaN

       trans_src  trans_dest
count        0.0         0.0
mean         NaN         NaN
std          NaN         NaN
min          NaN         NaN
25%          NaN         NaN
50%          NaN         NaN
75%          NaN         NaN
max          NaN         NaN
```

We can see that most of the columns in the dataset are empty, so dropping these columns is better.

2.1 Dealing with Missing values

[5]:
```
nan_cols = data.columns[data.isna().any()]
print("Columns which contains missing values: ")
nan_cols
```

```
Columns which contains missing values:
```

[5]:
```
Index(['place', 'quote_url', 'near', 'geo', 'source', 'user_rt_id', 'user_rt',
       'retweet_id', 'retweet_date', 'translate', 'trans_src', 'trans_dest'],
      dtype='object')
```

[6]:
```
print("Percentage of Missing values in the columns: ")
(data[nan_cols].isna().sum()) / len(data)
```

```
Percentage of Missing values in the columns:
```

[6]:
```
place           0.926494
quote_url       0.895220
near            1.000000
geo             1.000000
source          1.000000
user_rt_id      1.000000
user_rt         1.000000
retweet_id      1.000000
retweet_date    1.000000
translate       1.000000
trans_src       1.000000
trans_dest      1.000000
dtype: float64
```

[7]:
```
# Dropping all the columns which consists of missing values
data.drop(nan_cols, axis=1, inplace=True)
```

[8]:
```
print(data.columns)
```

```
Index(['id', 'conversation_id', 'created_at', 'date', 'time', 'timezone',
       'user_id', 'username', 'name', 'tweet', 'mentions', 'urls', 'photos',
       'replies_count', 'retweets_count', 'likes_count', 'hashtags',
       'cashtags', 'link', 'retweet', 'video', 'reply_to'],
      dtype='object')
```

By careful inspection, we can observe that all the columns except 'tweet' and 'hashtags' are redundant and can be dropped.

[9]:
```
actual_data = data[['tweet', 'hashtags']]
actual_data
```

[9]:
```
                                                    tweet  \
0        https://www.youtube.com/watch?v=-CRbO7ExO1k ……
1        Lockdown 4.0 ka.naam hi lockdown hai\nHai sab …
2        CORONA VIRUS THREAT-\nHOW TO OVERCOME STRESS A…
3        Could you please\n\n#lockdownindia\n@Bhuvan_Ba…
4        In fight with #COVID19, You are the best Docto…
…                                                    …
124379   I pledge to follow the appeal given by Hon'ble…
124380   Four new cases of Coronavirus detected in Luck…
124381   Do you sometimes feel a tingling #sensation or…
124382   Some Time we have to Stay Back … Just to Sav…
124383   Sir, National Medical Emergency should declare…

                                                 hashtags
0        ['#lockdownindia', '#lockdown', '#indiafightsc…
1                          ['#lockdownindia', '#locldown4']
2        ['#covid_19', '#covid_19sa', '#covid_19india',…
3        ['#lockdownindia', '#roastchallenge', '#journa…
4        ['#covid19', '#coronavirus', '#patiencechallen…
…                                                    …
124379   ['#staysafestayhome', '#janta_curfew', '#janta…
124380   ['#coronaindia', '#coronavirusoutbreakindia', …
124381   ['#sensation', '#hands', '#thevoiceofwoman', '…
124382   ['#stayback', '#gobackcorona', '#coronafighter…
124383   ['#coronaindia', '#coronavirusupdate', '#wewil…

[124384 rows x 2 columns]
```

2.2 Preprocessing data

In this, preprocessing step I have removed all the unneccessary information using the regular expression(re) modules like:

```
re.sub(r'(#\w+)', ' ', x) # to remove all the hashtags in the text.

re.sub(r'(\n)', ' ', x) # to remove all the newline characters in the text.

re.sub(r'(@\w+)', ' ', x) # to remove all the handles in the text.

re.sub(r'(…)', ' ', x) # to remove '…' from the text.

re.sub(r'(pic.twitter.com/\w+)', ' ', x) # to remove all pic URLS in the text.

re.sub(r'([^a-zA-Z0-9])\1+', r'\1', x) # to remove all repeating characters except alphabets a

re.sub(r'(http|https|ftp)://[a-zA-Z0-9\\.\?\-\=/]+\xa0', ' ', x) # to remove all URLS in the t
```

[10]:
```
# Removing all the unwanted information from of data
```

```
actual_data['tweet'] = actual_data['tweet'].apply(lambda x: re.sub(r'#\w+', '␣
↪', x))
actual_data['tweet'] = actual_data['tweet'].apply(lambda x: re.sub(r'\n', ' ',␣
↪x))
actual_data['tweet'] = actual_data['tweet'].apply(lambda x: re.sub(r'pic.
↪twitter.com/\w+', ' ', x))
actual_data['tweet'] = actual_data['tweet'].apply(lambda x: re.sub(r'@\w+', '␣
↪', x))
actual_data['tweet'] = actual_data['tweet'].apply(lambda x: re.
↪sub(r'(http|https|ftp)://[a-zA-Z0-9\\.\?\-\=/]+\xa0', ' ', x))
actual_data['tweet'] = actual_data['tweet'].apply(lambda x: re.sub(r'…', ' ',␣
↪x))
actual_data['tweet'] = actual_data['tweet'].apply(lambda x: re.
↪sub(r'([^a-zA-Z0-9])\1+', r'\1', x))
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing imports
until
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:4:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  after removing the cwd from sys.path.
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:5:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:6:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  import sys
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:8:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:9:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  if __name__ == '__main__':
```

data1, data2, data3 all are the copies of the actual_data dataframe's

- data1 is used in sentiment prediction using textblob.
- data2 is used in sentiment prediction using vaderSentiment.
- data3 is used in sentiment prediction using machine learning method.

[11]:
```python
# Making a copy of original dataset to be used for different strategy

data1 = actual_data.copy()
data2 = actual_data.copy()
data3 = actual_data.copy()
```

2.3 Lexicon based sentiment analysis

2.3.1 TextBlob

This sentiment analysis is done using TextBlob function of the module textblob.

The output of the function is a 2-element tuple which has a structure (polarity_value, subjectivity).

[12]:
```python
data1['sentiment'] = data1['tweet'].apply(lambda x: TextBlob(x).sentiment)
```

[13]:
```python
data1['polarity_value'] = data1['sentiment'].apply(lambda x: x[0])
data1['subjectivity'] = data1['sentiment'].apply(lambda x: x[1])
```

[14]:
```python
data1['polarity_type'] = np.where(data1['polarity_value'].values > 0,␣
↪'Positive', np.where(data1['polarity_value'].values < 0, 'Negative',␣
↪'Neutral'))
```

[15]:
```python
# Count of each type of tweets

data1['polarity_type'].value_counts()
```

[15]:
```
Positive    54165
Neutral     47780
Negative    22439
Name: polarity_type, dtype: int64
```

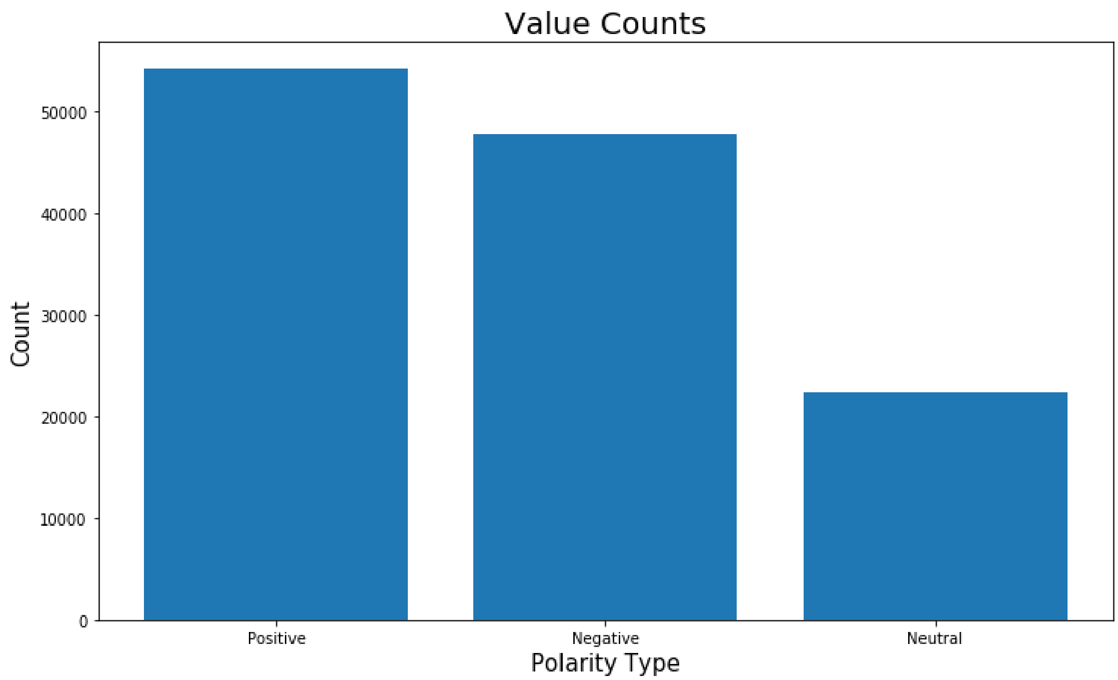
Visualization of data First plot, contains the Number of tweets for each type of tweets.

Second plot, contains the distribution of sentiment of all the tweets on the scale of [-1, 1](-1 being most negative, 1 being most positive).

[16]:
```python
# Polarity count

_, ax = plt.subplots()
ax.set_title("Value Counts", fontsize=20)
ax.set_xlabel("Polarity Type", fontsize=15)
ax.set_ylabel("Count", fontsize=15)
ax.bar(data1['polarity_type'].unique(), height=data1['polarity_type'].
↪value_counts())
```

[16]:
```
<BarContainer object of 3 artists>
```

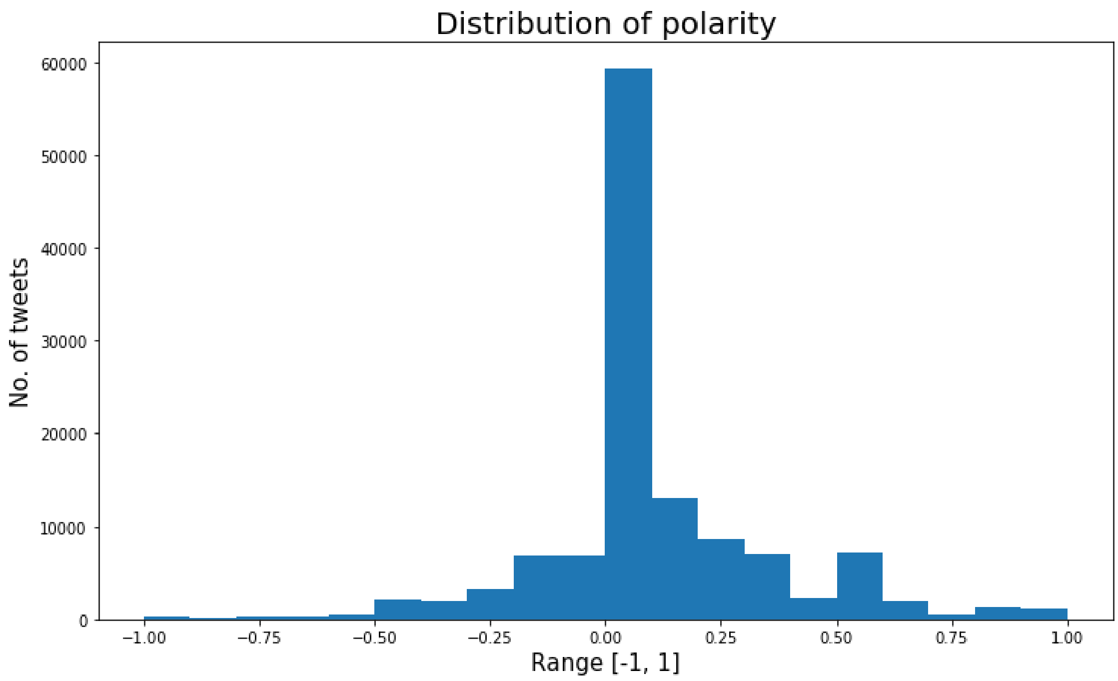


[17]:
```
# Distribution of the polarity value

_, ax = plt.subplots()
ax.set_title("Distribution of polarity", fontsize=20)
ax.set_xlabel("Range [-1, 1]", fontsize=15)
ax.set_ylabel("No. of tweets", fontsize=15)

ax.hist(data1['polarity_value'], bins=20)
```

[17]:
```
(array([3.2400e+02, 5.0000e+01, 2.7300e+02, 3.4700e+02, 4.8100e+02,
        2.1510e+03, 1.8610e+03, 3.3110e+03, 6.7980e+03, 6.8430e+03,
        5.9218e+04, 1.3030e+04, 8.6690e+03, 6.9420e+03, 2.3010e+03,
        7.0940e+03, 1.8880e+03, 4.6600e+02, 1.2830e+03, 1.0540e+03]),
 array([-1. , -0.9, -0.8, -0.7, -0.6, -0.5, -0.4, -0.3, -0.2, -0.1,  0. ,
         0.1,  0.2,  0.3,  0.4,  0.5,  0.6,  0.7,  0.8,  0.9,  1. ]),
 <a list of 20 Patch objects>)
```



2.3.2 VaderSentiment

This sentiment analysis is done using VaderSentiment module.

The output of the function is a dictionary which contains four keys 'positive', 'negative', 'neutral', 'compound'

compound gives the overall sentiment of the tweet.

positive says how much positive is the given tweet.

negative says how much negative is the given tweet.

neutral says how much neutral is the given tweet.

[18]:
```
analyzer = SentimentIntensityAnalyzer()

data2['sentiment'] = data2['tweet'].apply(lambda x: analyzer.polarity_scores(x))
```

[19]:
```
data2['positive'] = data2['sentiment'].apply(lambda x: x['pos'])
data2['negative'] = data2['sentiment'].apply(lambda x: x['neg'])
data2['neutral'] = data2['sentiment'].apply(lambda x: x['neu'])

data2['polarity_value'] = data2['sentiment'].apply(lambda x: x['compound'])
```

[20]:
```
data2['polarity_type'] = np.where(data2['polarity_value'].values > 0.05,␣
↪'Positive', np.where(data2['polarity_value'].values < -0.05, 'Negative',␣
↪'Neutral'))
```

[21]:
```
# Count of each type of tweets

data2['polarity_type'].value_counts()
```

[21]:
```
Positive    59641
Neutral     34188
Negative    30555
Name: polarity_type, dtype: int64
```

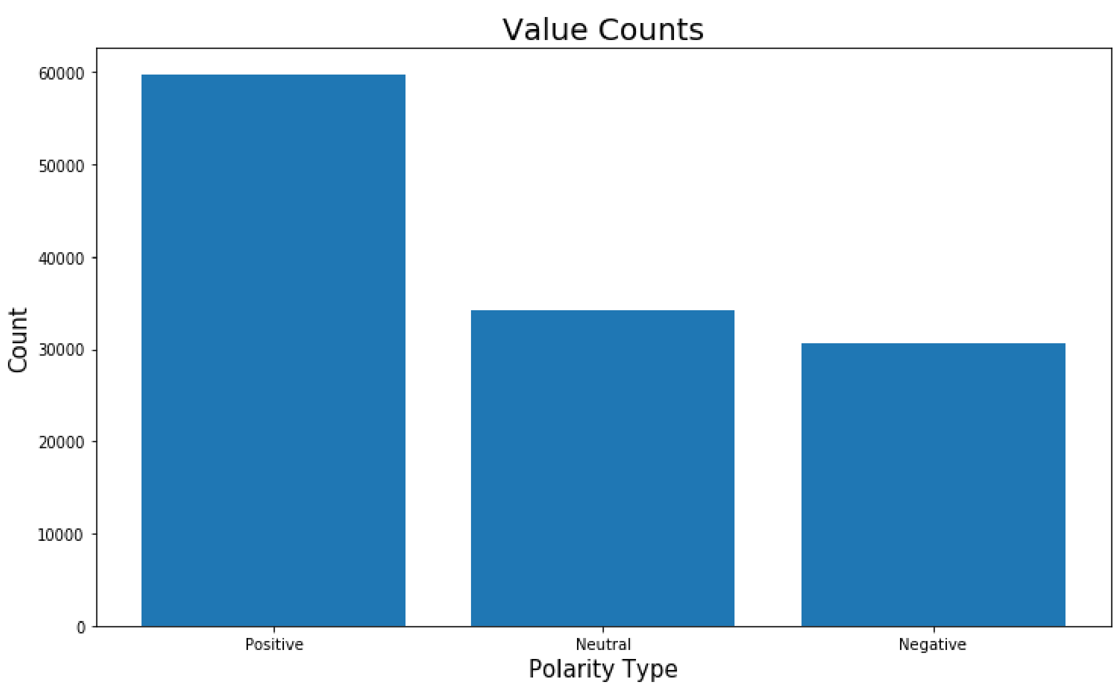
Visulization of data First plot, contains the Number of tweets for each type of tweets.

Second plot, contains the degree of degree of polarity of each polarity type.

[22]:
```
# Polarity count

_, ax = plt.subplots()
ax.set_title("Value Counts", fontsize=20)
ax.set_xlabel("Polarity Type", fontsize=15)
ax.set_ylabel("Count", fontsize=15)
ax.bar(data2['polarity_type'].unique(), height=data2['polarity_type'].
↪value_counts())
```

[22]: <BarContainer object of 3 artists>

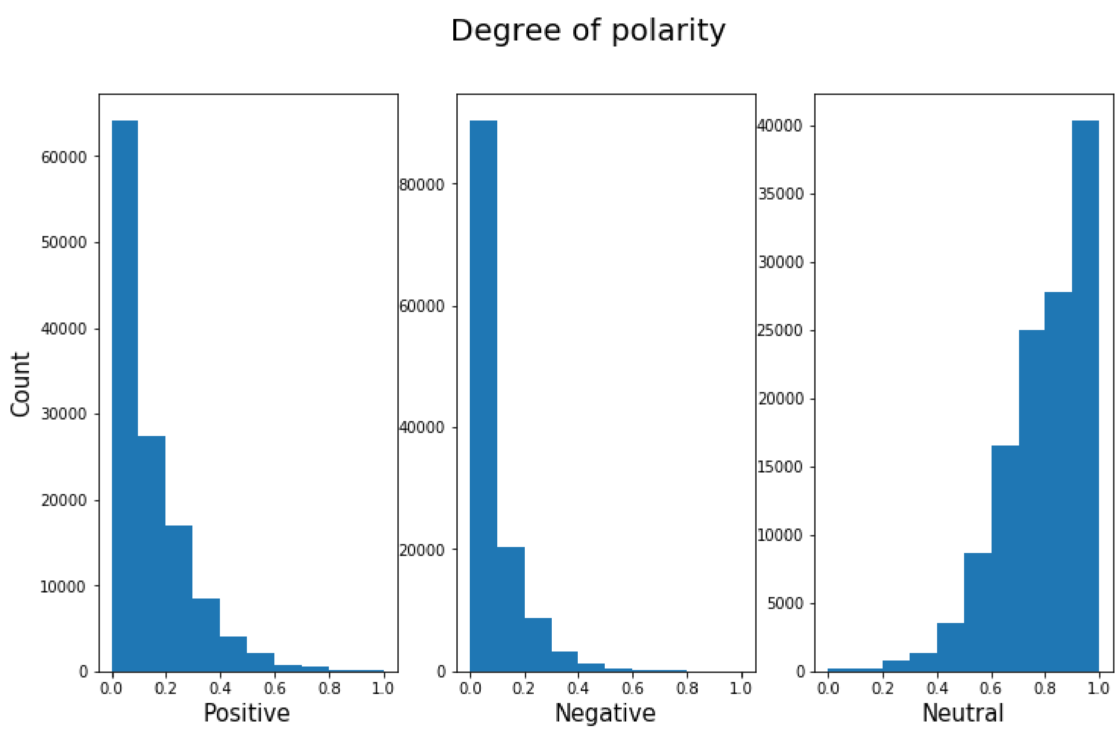


[23]:
```python
# Degree of polarity

fig, ax = plt.subplots(nrows=1, ncols=3)
fig.suptitle("Degree of polarity", fontsize=20)
ax[0].set_xlabel("Positive", fontsize=15)
ax[1].set_xlabel("Negative", fontsize=15)
ax[2].set_xlabel("Neutral", fontsize=15)
ax[0].set_ylabel("Count", fontsize=15)

ax[0].hist(data2['positive'])
ax[1].hist(data2['negative'])
ax[2].hist(data2['neutral'])
```

[23]:
```
(array([  228.,   184.,   712.,  1327.,  3573.,  8682., 16542., 25040.,
        27808., 40288.]),
 array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ]),
 <a list of 10 Patch objects>)
```



2.4 Machine Learning based Sentiment Analysis

Using the imdb dataset to train and build the neural network, after that apply the neural network on the tweets data given.

[24]:
```
# Loading the data

imdb_data = pd.read_csv('./data/imdb_data.zip', compression='zip')
imdb_data.head()
```

[24]:
```
                                              review sentiment
0  One of the other reviewers has mentioned that …  positive
1  A wonderful little production. <br /><br />The…  positive
2  I thought this was a wonderful way to spend ti…  positive
3  Basically there's a family where a little boy …  negative
4  Petter Mattei's "Love in the Time of Money" is…  positive
```

[ ]:
```
# Inspecting data

imdb_data.info()
imdb_data.describe()
```

2.4.1 Preprocessing

Removing all the html tags in the text.

[26]:
```
# Some preprocssing
# There are some html break tags(replacing them with ' ')

imdb_data['review'] = [re.sub(r'<.*?>', ' ', review) for review in␣
↪imdb_data['review']]
```

[27]:
```
# Adding few columns

imdb_data['review_len'] = imdb_data['review'].apply(lambda x: len(x.split()))
```

[28]:
```
# Converting 'sentiment' into numerical value

imdb_data['sentiment_numerical'] = np.where(imdb_data['sentiment'] ==␣
↪'positive', 1, 0)
```

[29]:
```
# Separating out features and labels

X = imdb_data['review']
y = imdb_data['sentiment_numerical']
```

[30]:
```
# Train-Test Split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,␣
↪random_state=88)
```

[31]:
```
max_words = 10000
oov_token = '<OOV>'

tokenizer = tf.keras.preprocessing.text.Tokenizer(num_words=max_words,␣
↪oov_token=oov_token)
```

[32]:
```
tokenizer.fit_on_texts(X)
tokenizer.word_index
```

[32]:
```
{'<OOV>': 1,
 'the': 2,
 'and': 3,
 'a': 4,
 'of': 5,
 'to': 6,
 'is': 7,
 'in': 8,
 'it': 9,
 'i': 10,
 'this': 11,
 'that': 12,
 'was': 13,
 'as': 14,
 'for': 15,
 'with': 16,
 'movie': 17,
 'but': 18,
 'film': 19,
 'on': 20,
 'not': 21,
 'you': 22,
 'are': 23,
 'his': 24,
 'have': 25,
 'be': 26,
 'one': 27,
 'he': 28,
 'all': 29,
 'at': 30,
 'by': 31,
 'an': 32,
 'they': 33,
```

```
'so': 34,
'who': 35,
'from': 36,
'like': 37,
'or': 38,
'just': 39,
'her': 40,
'out': 41,
'about': 42,
'if': 43,
"it's": 44,
'has': 45,
'there': 46,
'some': 47,
'what': 48,
'good': 49,
'when': 50,
'more': 51,
'very': 52,
'up': 53,
'no': 54,
'time': 55,
'my': 56,
'even': 57,
'would': 58,
'she': 59,
'which': 60,
'only': 61,
'really': 62,
'see': 63,
'story': 64,
'their': 65,
'had': 66,
'can': 67,
'me': 68,
'well': 69,
'were': 70,
'than': 71,
'much': 72,
'we': 73,
'bad': 74,
'been': 75,
'get': 76,
'do': 77,
'great': 78,
'other': 79,
'will': 80,
```

```
'also': 81,
'into': 82,
'people': 83,
'because': 84,
'how': 85,
'first': 86,
'him': 87,
'most': 88,
"don't": 89,
'made': 90,
'then': 91,
'them': 92,
'its': 93,
'make': 94,
'way': 95,
'too': 96,
'movies': 97,
'could': 98,
'any': 99,
'after': 100,
'think': 101,
'characters': 102,
'watch': 103,
'films': 104,
'two': 105,
'many': 106,
'seen': 107,
'character': 108,
'being': 109,
'never': 110,
'plot': 111,
'love': 112,
'acting': 113,
'life': 114,
'did': 115,
'best': 116,
'where': 117,
'know': 118,
'show': 119,
'little': 120,
'over': 121,
'off': 122,
'ever': 123,
'does': 124,
'your': 125,
'better': 126,
'end': 127,
```

```
'man': 128,
'scene': 129,
'still': 130,
'say': 131,
'these': 132,
'here': 133,
'scenes': 134,
'why': 135,
'while': 136,
'something': 137,
'such': 138,
'go': 139,
'through': 140,
'back': 141,
'should': 142,
'those': 143,
'real': 144,
"i'm": 145,
'now': 146,
'watching': 147,
'thing': 148,
"doesn't": 149,
'actors': 150,
'though': 151,
'funny': 152,
'years': 153,
"didn't": 154,
'old': 155,
'10': 156,
'another': 157,
'work': 158,
'before': 159,
'actually': 160,
'nothing': 161,
'makes': 162,
'look': 163,
'director': 164,
'find': 165,
'going': 166,
'same': 167,
'new': 168,
'lot': 169,
'every': 170,
'few': 171,
'again': 172,
'part': 173,
'cast': 174,
```

```
'down': 175,
'us': 176,
'things': 177,
'want': 178,
'quite': 179,
'pretty': 180,
'world': 181,
'horror': 182,
'around': 183,
'seems': 184,
"can't": 185,
'young': 186,
'take': 187,
'however': 188,
'got': 189,
'thought': 190,
'big': 191,
'fact': 192,
'enough': 193,
'long': 194,
'both': 195,
"that's": 196,
'give': 197,
"i've": 198,
'own': 199,
'may': 200,
'between': 201,
'comedy': 202,
'right': 203,
'series': 204,
'action': 205,
'must': 206,
'music': 207,
'without': 208,
'times': 209,
'saw': 210,
'always': 211,
'original': 212,
"isn't": 213,
'role': 214,
'come': 215,
'almost': 216,
'gets': 217,
'interesting': 218,
'guy': 219,
'point': 220,
'done': 221,
```

"there's": 222,
'whole': 223,
'least': 224,
'far': 225,
'bit': 226,
'script': 227,
'minutes': 228,
'feel': 229,
'2': 230,
'anything': 231,
'making': 232,
'might': 233,
'since': 234,
'am': 235,
'family': 236,
"he's": 237,
'last': 238,
'probably': 239,
'tv': 240,
'performance': 241,
'kind': 242,
'away': 243,
'yet': 244,
'fun': 245,
'worst': 246,
'sure': 247,
'rather': 248,
'hard': 249,
'anyone': 250,
'girl': 251,
'each': 252,
'played': 253,
'day': 254,
'found': 255,
'looking': 256,
'woman': 257,
'screen': 258,
'although': 259,
'our': 260,
'especially': 261,
'believe': 262,
'having': 263,
'trying': 264,
'course': 265,
'dvd': 266,
'everything': 267,
'set': 268,

'goes': 269,
'comes': 270,
'put': 271,
'ending': 272,
'maybe': 273,
'place': 274,
'book': 275,
'shows': 276,
'three': 277,
'worth': 278,
'different': 279,
'main': 280,
'once': 281,
'sense': 282,
'american': 283,
'reason': 284,
'looks': 285,
'effects': 286,
'watched': 287,
'play': 288,
'true': 289,
'money': 290,
'actor': 291,
"wasn't": 292,
'job': 293,
'together': 294,
'war': 295,
'someone': 296,
'plays': 297,
'instead': 298,
'high': 299,
'during': 300,
'year': 301,
'said': 302,
'half': 303,
'everyone': 304,
'later': 305,
'takes': 306,
'1': 307,
'seem': 308,
'audience': 309,
'special': 310,
'beautiful': 311,
'left': 312,
'himself': 313,
'seeing': 314,
'john': 315,

```
'night': 316,
'black': 317,
'version': 318,
'shot': 319,
'excellent': 320,
'idea': 321,
'house': 322,
'mind': 323,
'star': 324,
'wife': 325,
'fan': 326,
'death': 327,
'used': 328,
'else': 329,
'simply': 330,
'nice': 331,
'budget': 332,
'poor': 333,
'short': 334,
'completely': 335,
'second': 336,
"you're": 337,
'3': 338,
'read': 339,
'less': 340,
'along': 341,
'top': 342,
'help': 343,
'home': 344,
'men': 345,
'either': 346,
'line': 347,
'boring': 348,
'dead': 349,
'friends': 350,
'kids': 351,
'try': 352,
'production': 353,
'enjoy': 354,
'camera': 355,
'use': 356,
'wrong': 357,
'given': 358,
'low': 359,
'classic': 360,
'father': 361,
'need': 362,
```

```
'full': 363,
'stupid': 364,
'until': 365,
'next': 366,
'performances': 367,
'school': 368,
'hollywood': 369,
'rest': 370,
'truly': 371,
'awful': 372,
'video': 373,
'couple': 374,
'start': 375,
'sex': 376,
'recommend': 377,
'women': 378,
'let': 379,
'tell': 380,
'terrible': 381,
'remember': 382,
'mean': 383,
'came': 384,
'getting': 385,
'understand': 386,
'perhaps': 387,
'moments': 388,
'name': 389,
'keep': 390,
'face': 391,
'itself': 392,
'wonderful': 393,
'playing': 394,
'human': 395,
'style': 396,
'small': 397,
'episode': 398,
'perfect': 399,
'others': 400,
'person': 401,
'doing': 402,
'often': 403,
'early': 404,
'stars': 405,
'definitely': 406,
'written': 407,
'head': 408,
'lines': 409,
```

```
'dialogue': 410,
'gives': 411,
'piece': 412,
"couldn't": 413,
'went': 414,
'finally': 415,
'mother': 416,
'case': 417,
'title': 418,
'absolutely': 419,
'boy': 420,
'live': 421,
'yes': 422,
'laugh': 423,
'certainly': 424,
'liked': 425,
'become': 426,
'entertaining': 427,
'worse': 428,
'oh': 429,
'sort': 430,
'loved': 431,
'lost': 432,
'called': 433,
'hope': 434,
'picture': 435,
'felt': 436,
'overall': 437,
'entire': 438,
'mr': 439,
'several': 440,
'based': 441,
'supposed': 442,
'cinema': 443,
'friend': 444,
'guys': 445,
'sound': 446,
'5': 447,
'problem': 448,
'drama': 449,
'against': 450,
'waste': 451,
'white': 452,
'beginning': 453,
'4': 454,
'fans': 455,
'totally': 456,
```

```
'dark': 457,
'care': 458,
'direction': 459,
'humor': 460,
'wanted': 461,
"she's": 462,
'seemed': 463,
'under': 464,
'game': 465,
'children': 466,
'despite': 467,
'lives': 468,
'lead': 469,
'guess': 470,
'example': 471,
'already': 472,
'final': 473,
"you'll": 474,
'throughout': 475,
'turn': 476,
'evil': 477,
'becomes': 478,
'unfortunately': 479,
'able': 480,
'quality': 481,
"i'd": 482,
'days': 483,
'history': 484,
'fine': 485,
'side': 486,
'wants': 487,
'horrible': 488,
'heart': 489,
'writing': 490,
'amazing': 491,
'b': 492,
'flick': 493,
'killer': 494,
'run': 495,
'son': 496,
'\x96': 497,
'michael': 498,
'works': 499,
'close': 500,
"they're": 501,
'act': 502,
'art': 503,
```

```
'kill': 504,
'matter': 505,
'etc': 506,
'tries': 507,
"won't": 508,
'past': 509,
'town': 510,
'enjoyed': 511,
'turns': 512,
'brilliant': 513,
'gave': 514,
'behind': 515,
'parts': 516,
'stuff': 517,
'genre': 518,
'eyes': 519,
'favorite': 520,
'car': 521,
'directed': 522,
'late': 523,
'hand': 524,
'expect': 525,
'soon': 526,
'hour': 527,
'obviously': 528,
'themselves': 529,
'sometimes': 530,
'killed': 531,
'thinking': 532,
'actress': 533,
'child': 534,
'girls': 535,
'viewer': 536,
'starts': 537,
'city': 538,
'myself': 539,
'decent': 540,
'highly': 541,
'stop': 542,
'type': 543,
'self': 544,
'god': 545,
'says': 546,
'group': 547,
'anyway': 548,
'voice': 549,
'took': 550,
```

```
'known': 551,
'blood': 552,
'kid': 553,
'heard': 554,
'happens': 555,
'except': 556,
'fight': 557,
'feeling': 558,
'experience': 559,
'coming': 560,
'slow': 561,
'daughter': 562,
'writer': 563,
'stories': 564,
'moment': 565,
'leave': 566,
'told': 567,
'extremely': 568,
'score': 569,
'violence': 570,
'involved': 571,
'police': 572,
'strong': 573,
'lack': 574,
'chance': 575,
'cannot': 576,
'hit': 577,
'roles': 578,
'hilarious': 579,
's': 580,
'wonder': 581,
'happen': 582,
'particularly': 583,
'ok': 584,
'including': 585,
'save': 586,
'living': 587,
'looked': 588,
"wouldn't": 589,
'crap': 590,
'please': 591,
'simple': 592,
'murder': 593,
'cool': 594,
'obvious': 595,
'happened': 596,
'complete': 597,
```

```
'cut': 598,
'serious': 599,
'age': 600,
'gore': 601,
'attempt': 602,
'hell': 603,
'ago': 604,
'song': 605,
'shown': 606,
'taken': 607,
'english': 608,
'james': 609,
'robert': 610,
'david': 611,
'seriously': 612,
'released': 613,
'reality': 614,
'opening': 615,
'jokes': 616,
'interest': 617,
'across': 618,
'none': 619,
'hero': 620,
'exactly': 621,
'today': 622,
'possible': 623,
'alone': 624,
'sad': 625,
'brother': 626,
'number': 627,
'career': 628,
'saying': 629,
"film's": 630,
'usually': 631,
'hours': 632,
'cinematography': 633,
'talent': 634,
'view': 635,
'annoying': 636,
'running': 637,
'yourself': 638,
'relationship': 639,
'documentary': 640,
'wish': 641,
'order': 642,
'huge': 643,
'shots': 644,
```

```
'whose': 645,
'ridiculous': 646,
'taking': 647,
'important': 648,
'light': 649,
'body': 650,
'middle': 651,
'level': 652,
'ends': 653,
'started': 654,
'female': 655,
'call': 656,
'husband': 657,
"i'll": 658,
'four': 659,
'word': 660,
'turned': 661,
'power': 662,
'major': 663,
'opinion': 664,
'change': 665,
'mostly': 666,
'usual': 667,
'scary': 668,
'silly': 669,
'rating': 670,
'beyond': 671,
'somewhat': 672,
'happy': 673,
'ones': 674,
'words': 675,
'room': 676,
'knows': 677,
'knew': 678,
'country': 679,
'disappointed': 680,
'talking': 681,
'novel': 682,
'apparently': 683,
'non': 684,
'strange': 685,
'upon': 686,
'attention': 687,
'basically': 688,
'single': 689,
'finds': 690,
'cheap': 691,
```

'modern': 692,
'due': 693,
'jack': 694,
'television': 695,
'musical': 696,
'problems': 697,
'miss': 698,
'episodes': 699,
'clearly': 700,
'local': 701,
'7': 702,
'british': 703,
'thriller': 704,
'talk': 705,
'events': 706,
'sequence': 707,
'five': 708,
"aren't": 709,
'class': 710,
'french': 711,
'moving': 712,
'ten': 713,
'fast': 714,
'earth': 715,
'tells': 716,
'review': 717,
'predictable': 718,
'songs': 719,
'team': 720,
'comic': 721,
'straight': 722,
'whether': 723,
'8': 724,
'die': 725,
'add': 726,
'dialog': 727,
'entertainment': 728,
'above': 729,
'sets': 730,
'future': 731,
'enjoyable': 732,
'appears': 733,
'near': 734,
'space': 735,
'easily': 736,
'hate': 737,
'soundtrack': 738,

```
'bring': 739,
'giving': 740,
'lots': 741,
'similar': 742,
'romantic': 743,
'george': 744,
'supporting': 745,
'release': 746,
'mention': 747,
'within': 748,
'filmed': 749,
'message': 750,
'sequel': 751,
'clear': 752,
'falls': 753,
"haven't": 754,
'needs': 755,
'dull': 756,
'suspense': 757,
'bunch': 758,
'eye': 759,
'surprised': 760,
'showing': 761,
'tried': 762,
'sorry': 763,
'certain': 764,
'working': 765,
'easy': 766,
'ways': 767,
'theme': 768,
'theater': 769,
'among': 770,
'named': 771,
"what's": 772,
'storyline': 773,
'monster': 774,
'king': 775,
'stay': 776,
'effort': 777,
'minute': 778,
'fall': 779,
'stand': 780,
'gone': 781,
'rock': 782,
'using': 783,
'9': 784,
'feature': 785,
```

'comments': 786,
'buy': 787,
"'": 788,
'typical': 789,
't': 790,
'editing': 791,
'sister': 792,
'tale': 793,
'avoid': 794,
'dr': 795,
'mystery': 796,
'deal': 797,
'doubt': 798,
'fantastic': 799,
'kept': 800,
'nearly': 801,
'feels': 802,
'subject': 803,
'okay': 804,
'viewing': 805,
'elements': 806,
'oscar': 807,
'check': 808,
'points': 809,
'realistic': 810,
'means': 811,
'greatest': 812,
'herself': 813,
'parents': 814,
'famous': 815,
'imagine': 816,
'rent': 817,
'viewers': 818,
'richard': 819,
'crime': 820,
'form': 821,
'peter': 822,
'actual': 823,
'lady': 824,
'general': 825,
'dog': 826,
'follow': 827,
'believable': 828,
'period': 829,
'red': 830,
'brought': 831,
'move': 832,

'material': 833,
'forget': 834,
'somehow': 835,
'begins': 836,
're': 837,
'reviews': 838,
'animation': 839,
'paul': 840,
"you've": 841,
'leads': 842,
'weak': 843,
'figure': 844,
'surprise': 845,
'hear': 846,
'sit': 847,
'average': 848,
'open': 849,
'sequences': 850,
'killing': 851,
'atmosphere': 852,
'eventually': 853,
'tom': 854,
'learn': 855,
'premise': 856,
'wait': 857,
'20': 858,
'sci': 859,
'deep': 860,
'fi': 861,
'expected': 862,
'whatever': 863,
'indeed': 864,
'lame': 865,
'poorly': 866,
'particular': 867,
'note': 868,
'dance': 869,
'imdb': 870,
'shame': 871,
'situation': 872,
'third': 873,
'york': 874,
'box': 875,
'truth': 876,
'decided': 877,
'free': 878,
'hot': 879,

```
"who's": 880,
'difficult': 881,
'needed': 882,
'season': 883,
'acted': 884,
'leaves': 885,
'unless': 886,
'romance': 887,
'emotional': 888,
'possibly': 889,
'gay': 890,
'sexual': 891,
'boys': 892,
'footage': 893,
'write': 894,
'western': 895,
'credits': 896,
'forced': 897,
'memorable': 898,
'doctor': 899,
'reading': 900,
'became': 901,
'otherwise': 902,
'air': 903,
'begin': 904,
'de': 905,
'crew': 906,
'question': 907,
'meet': 908,
'society': 909,
'male': 910,
"let's": 911,
'meets': 912,
'plus': 913,
'cheesy': 914,
'hands': 915,
'superb': 916,
'screenplay': 917,
'beauty': 918,
'interested': 919,
'street': 920,
'features': 921,
'perfectly': 922,
'masterpiece': 923,
'whom': 924,
'laughs': 925,
'nature': 926,
```

```
'stage': 927,
'effect': 928,
'forward': 929,
'comment': 930,
'nor': 931,
'previous': 932,
'e': 933,
'badly': 934,
'sounds': 935,
'japanese': 936,
'weird': 937,
'island': 938,
'inside': 939,
'personal': 940,
'quickly': 941,
'total': 942,
'keeps': 943,
'towards': 944,
'america': 945,
'result': 946,
'crazy': 947,
'battle': 948,
'worked': 949,
'incredibly': 950,
'setting': 951,
'earlier': 952,
'background': 953,
'mess': 954,
'cop': 955,
'writers': 956,
'fire': 957,
'copy': 958,
'realize': 959,
'dumb': 960,
'unique': 961,
'powerful': 962,
'mark': 963,
'lee': 964,
'business': 965,
'rate': 966,
'older': 967,
'dramatic': 968,
'pay': 969,
'following': 970,
'girlfriend': 971,
'directors': 972,
'joke': 973,
```

```
 'plenty': 974,
 'directing': 975,
 'various': 976,
 'baby': 977,
 'creepy': 978,
 'development': 979,
 'appear': 980,
 'brings': 981,
 'front': 982,
 'dream': 983,
 'ask': 984,
 'water': 985,
 'rich': 986,
 'bill': 987,
 'admit': 988,
 'apart': 989,
 'joe': 990,
 'political': 991,
 'fairly': 992,
 'leading': 993,
 'reasons': 994,
 'spent': 995,
 'portrayed': 996,
 'telling': 997,
 'outside': 998,
 'cover': 999,
 'fighting': 1000,
 …}
```

[33]:
```python
# Converting tokens into Sequences

X_train_tokens = tokenizer.texts_to_sequences(X_train)
X_test_tokens = tokenizer.texts_to_sequences(X_test)
```

[34]:
```python
max_train_word = int(np.mean(imdb_data['review_len']) + 2 * np.
 ↪std(imdb_data['review_len']))
```

[35]:
```python
# Padding the input to make it uniform length

pad_type = 'pre'

X_train_padded = tf.keras.preprocessing.sequence.pad_sequences(X_train_tokens,␣
 ↪maxlen=max_train_word, padding=pad_type, truncating=pad_type)
X_test_padded = tf.keras.preprocessing.sequence.pad_sequences(X_test_tokens,␣
 ↪maxlen=max_train_word, padding=pad_type, truncating=pad_type)
```

Model Architecture

My model consists of 5 layers, 1. Embedding layer (with a dimension of 8)

which finds the word embedding's of all the unique words in the text which are tokenized by the 'tokenizer' function.

2. LSTM (with an input dimension of 16)
3. LSTM (with an input dimension of 8)
4. LSTM (with an input dimension of 4)
5. Dense (this output, the sentiment)

For more information about the model you can see cell number 39.

Hyperparameters

The Hyperparameters for this model are * Input dimension * Ouput dimension * Length of the input sequence * Number of layers and there dimension's * Number of epochs * Batch Size

[36]:
```
# Defining the model

embedding_dim = 8

model = tf.keras.models.Sequential([
    tf.keras.layers.Embedding(input_dim=max_words,
                              output_dim=embedding_dim,
                              input_length=max_train_word,
                              name='embedding_layer'),
    tf.keras.layers.LSTM(units=16, return_sequences=True),
    tf.keras.layers.LSTM(units=8, return_sequences=True),
    tf.keras.layers.LSTM(units=4),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

[37]:
```
# Defining the optimizer

optimizer = tf.keras.optimizers.Adam(lr=0.1)
```

[38]:
```
model.compile(loss='binary_crossentropy',
              optimizer=optimizer,
              metrics=['acc'])
```

[39]:
```
model.summary()
```

```
Model: "sequential"

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_layer (Embedding)  (None, 568, 8)            80000
_________________________________________________________________
unified_lstm (UnifiedLSTM)   (None, 568, 16)           1600
_________________________________________________________________
```

```
unified_lstm_1 (UnifiedLSTM) (None, 568, 8)            800
_________________________________________________________________
unified_lstm_2 (UnifiedLSTM) (None, 4)                 208
_________________________________________________________________
dense (Dense)                (None, 1)                 5
=================================================================
Total params: 82,613
Trainable params: 82,613
Non-trainable params: 0
_________________________________________________________________
```

[40]:
```
# Model fitting on the training data

epochs=3
batch_size = 64

history = model.fit(X_train_padded, y_train, validation_split=0.05,␣
↪epochs=epochs, batch_size=batch_size)
```

```
Train on 35625 samples, validate on 1875 samples
Epoch 1/3
35625/35625 [==============================] - 475s 13ms/sample - loss: 0.5990 -
acc: 0.6638 - val_loss: 0.4796 - val_acc: 0.7739
Epoch 2/3
35625/35625 [==============================] - 495s 14ms/sample - loss: 0.4278 -
acc: 0.8044 - val_loss: 0.4274 - val_acc: 0.8048
Epoch 3/3
35625/35625 [==============================] - 531s 15ms/sample - loss: 0.3969 -
acc: 0.8219 - val_loss: 0.4303 - val_acc: 0.8096
```

[41]:
```
# Measuring Training Loss

train_loss = model.evaluate(X_train_padded, y_train)
print("Training accuracy: {:6f}".format(train_loss[1]))
```

```
37500/37500 [==============================] - 108s 3ms/sample - loss: 0.3926 -
acc: 0.8258
Training accuracy: 0.825787
```

[42]:
```
# Measuring Testing Loss

test_loss = model.evaluate(X_test_padded, y_test)
print("Testing accuracy: {:6f}".format(test_loss[1]))
```

```
12500/12500 [==============================] - 37s 3ms/sample - loss: 0.4067 -
acc: 0.8182
Testing accuracy: 0.818240
```

2.5 Now introducing the original data

2.5.1 Some preprocessing

Removing all the emoji's from the text and converting the text into lower case.

[43]:
```
# Removing unnecessary spaces and emoji's

data3['tweet'] = data3['tweet'].apply(lambda x: " ".join(map(str.lower, x.
↪split())))
data3['tweet'] = data3['tweet'].apply(lambda x: "".join([char for char in x if␣
↪char not in emoji.UNICODE_EMOJI]))
```

2.5.2 Prediction using pretrained model

[44]:
```
# Applying trained model on tweets data

X_data_tokens = tokenizer.texts_to_sequences(data3['tweet'])
X_data_padded = tf.keras.preprocessing.sequence.pad_sequences(X_data_tokens,␣
↪maxlen=max_train_word, padding=pad_type, truncating=pad_type)
```

[50]:
```
predicted = model.predict(X_data_padded)
sentiment = np.where(predicted > 0.5, 'Positive', 'Negative')
```

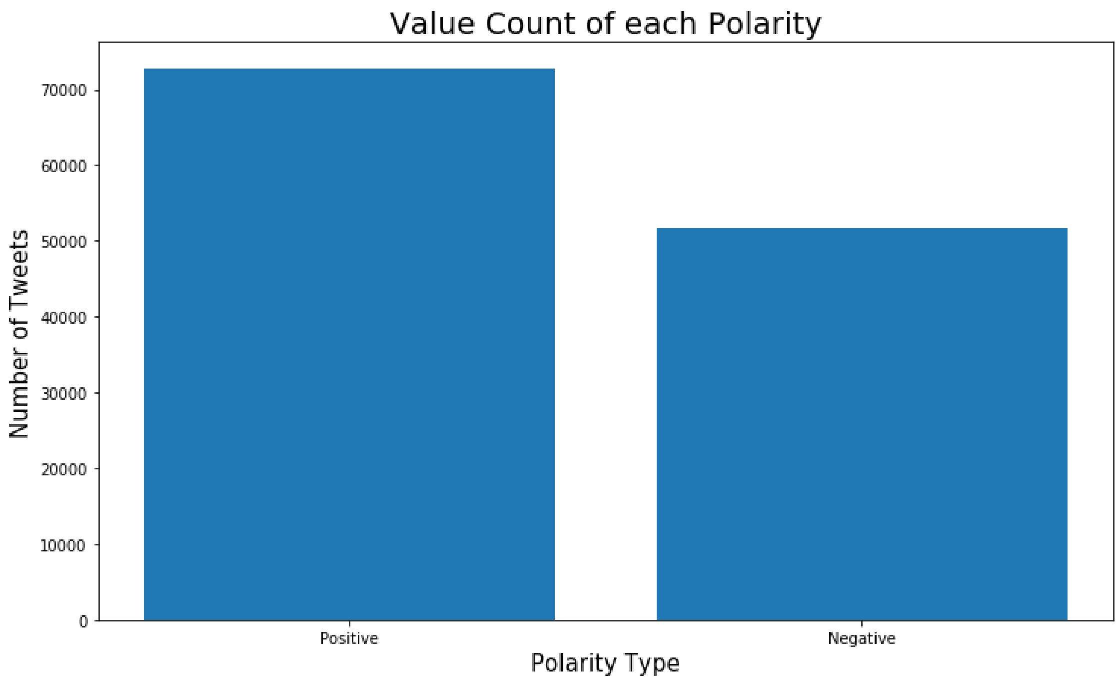
[52]:
```
final_data = pd.DataFrame(data=np.append(predicted, sentiment, axis=1),␣
↪columns=['Polarity_value', 'Polarity_type'])
```

2.5.3 Visualization

[53]:
```
# Some plots

_, ax = plt.subplots()
ax.set_xlabel("Polarity Type", fontsize=15)
ax.set_ylabel("Number of Tweets", fontsize=15)
ax.set_title("Value Count of each Polarity", fontsize=20)
ax.bar(final_data['Polarity_type'].unique(), final_data['Polarity_type'].
↪value_counts())
```

[53]:
```
<BarContainer object of 2 artists>
```


Value Count of each Polarity
Number of Tweets
70000
60000
50000
40000
30000
20000
10000
0
Positive
Negative
Polarity Type