Twitter Sentiment Analysis

COURSE PROJECT REPORT

18CSE398J -Machine Learning - Core Concepts with Applications

(2018 Regulation)

III Year/ VI Semester

Academic Year: 2022 -2023 (EVEN)

By

NAMAN ANAND – RA2011029010013

SRINIVAS T.B. – RA2011029010015

KIRTI KALAL – RA2011029010031

Under the guidance of

DR.G. VADIVU

Professor

Department of Data Science and Business Systems

DEPARTMENT OF DATA SCIENCE AND BUSINESS SYSTEMS

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

Kattankulathur, Kancheepuram

MAY 2023

TABLE OF CONTENTS

ABSTRACT

Twitter Sentiment Analysis is the process of extracting and categorizing emotions expressed in tweets posted on the social media platform Twitter. This technique has become increasingly popular in recent years due to the growing availability of large amounts of social media data, which can be used to gain insights into public opinions and attitudes towards various topics, products, or events.

Machine learning algorithms have proven to be effective in analyzing large datasets and classifying the sentiment of tweets as positive, negative, or neutral. This project aims to explore the use of machine learning algorithms to analyze Twitter sentiment and determine the effectiveness of different techniques for this task.

The project will involve collecting and preprocessing a large dataset of tweets, selecting appropriate features for sentiment analysis, and training and testing various machine learning models to predict the sentiment of tweets, and the project also involves generating word clouds, finding similar words.

The results of this project can have practical applications in various fields, such as marketing, politics, and social research, by providing insights into public opinion and sentiment towards products, services, events, or political issues.

INTRODUCTION

Sentiment analysis (also known as opinion mining or emotion AI) is the use of natural language processing, text analysis, computational linguistics, and systematically identify, extract, quantify, and study affective states and subjective information. It identifies the emotional tone behind a body of text. Sentiment analysis is widely applied to voice of the customer materials such as reviews and survey responses, online and social media, and healthcare materials for applications that range from marketing to customer service to clinical medicine. A sentiment analysis tool is an automated technique that extracts meaningful information related to a person’s attitudes, emotions, and opinions.

A basic task in sentiment analysis is classifying the polarity of a given text at the document, sentence, or feature/aspect level—whether the expressed opinion in a document, a sentence or an entity feature/aspect is positive, negative, or neutral. A Twitter sentiment analysis is the process of determining the emotional tone behind a series of words, specifically on Twitter.

Twitter sentiment analysis allows to keep track of what's being said about any product or service on social media, and can help detect angry customers or negative mentions before they escalate. Python is widely used for twitter sentiment analysis.

It therefore means, using advanced text mining techniques to analyze the sentiment of the text (here, tweet) in the form of positive, negative and neutral. It is also known as Opinion Mining, is primarily for analyzing conversations, opinions, and sharing of views (all in the form of tweets) for deciding business strategy, political analysis, and also for assessing public actions. Python is widely used for twitter sentiment analysis.

DATASET

The Twitter dataset extracted through a script is a collection of tweets that were scraped using a custom Python script. The dataset contains a vast amount of text data, including the tweet's content, author, location, date, and any associated metadata. The tweets in the dataset can be filtered and sorted by keywords, hashtags, or user accounts.

The dataset is ideal for text analysis and machine learning projects, as it contains a variety of tweet types, including retweets, quotes, and replies. The data can be used to analyze trends in social media conversations, sentiment analysis, and topic modeling.

The Twitter dataset extracted through a script is updated frequently and can be easily integrated into existing data science workflows. The dataset's rich metadata provides additional insights into Twitter users' behavior, such as the number of followers, engagement rates, and tweet frequency.

Overall, this Twitter dataset provides a valuable resource for researchers, marketers, and data scientists looking to gain insights into social media conversations and trends.

The dataset contains 1048576 rows of data. The attributes in the dataset are as follows:

- First column shows the emotion of the tweet.
- Second column shows the unique id of the tweet.
- Third column shows the day, date, and time of the tweet.
- Fourth column shows
- Fifth column shows the username of the user.
- Sixth column shows the tweet done by a particular user.

METHODS

The libraries used in the project are explained below:

NLTK (Natural Language Toolkit)

NLTK is one of the most widely used NLP libraries in Python. It provides a comprehensive suite of libraries and programs for tasks such as tokenization, stemming, lemmatization, and parsing of natural language data. For sentiment analysis, NLTK is often used for text preprocessing, which involves converting raw text data into a format that can be analyzed using machine learning algorithms.

One of the key features of NLTK is its pre-trained classifiers and models. NLTK includes a number of pre-trained classifiers for sentiment analysis, including Naive Bayes, Maximum Entropy, and Support Vector Machines (SVMs). These classifiers can be used to classify tweets into positive, negative, or neutral categories based on their content.

NLTK also includes a number of corpora (collections of texts) that can be used for training and testing sentiment analysis models. For example, the "movie_reviews" corpus contains 2,000 movie reviews that have been classified as either positive or negative. This corpus can be used to train a sentiment analysis model that can then be applied to Twitter data.

Scikit-learn

Scikit-learn is a popular machine learning library in Python that provides a wide range of algorithms for classification, regression, and clustering tasks. Scikit-learn includes several machine learning algorithms that are commonly used in sentiment analysis, including Support Vector Machines (SVM), Naive Bayes, and Random Forest.

Scikit-learn can be used for both supervised and unsupervised learning. In supervised learning, the algorithm is trained on a labeled dataset of tweets (i.e., tweets that have been labeled as positive, negative, or neutral), and then used to classify new tweets based on their content. In unsupervised learning, the algorithm is used to identify patterns or clusters in the data without any prior knowledge of the labels.

Scikit-learn also includes several preprocessing and feature extraction modules that can be used to preprocess and analyze textual data. These modules include CountVectorizer, which converts a collection of tweets into a matrix of token counts, and TfidfVectorizer, which converts a collection of tweets into a matrix of TF-IDF features.

One of the advantages of Scikit-learn is its speed and scalability. It can be used to train and evaluate machine learning models on large datasets of tweets in a relatively short amount of time.

Gensim

Gensim is a popular library used for topic modeling and text similarity tasks. It includes several modules for analyzing textual data, including pre-processing, similarity measurement, and topic modeling. Gensim can be used to identify the most relevant topics in a tweet dataset, which can help to improve the accuracy of sentiment analysis.

Matplotlib

Matplotlib is a popular library used for data visualization tasks. It includes several modules for creating different types of plots and graphs that can be used to visualize sentiment analysis results. Matplotlib can be used to visualize the sentiment distribution of tweets, the most frequent positive and negative words used, and the sentiment of tweets over time.

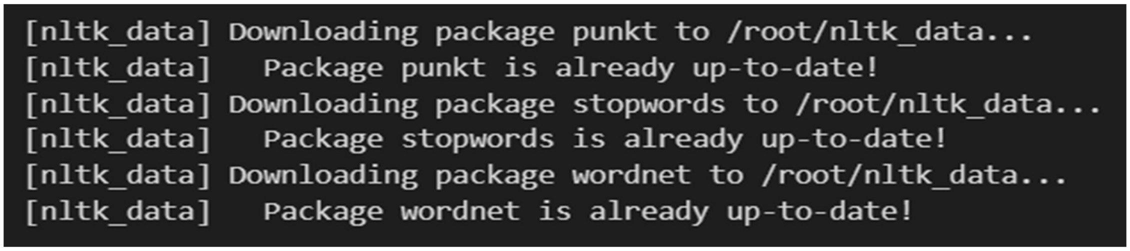
Seaborn

Seaborn is another popular library used for data visualization tasks. It provides several modules for creating statistical visualizations that can be used to analyze and compare sentiment analysis results. Seaborn can be used to create bar plots, heatmaps, and box plots to visualize the sentiment distribution of tweets, the most frequent positive and negative words used, and the sentiment of tweets over time.

EXPERIMENTS AND RESULTS

The codes are given below as text and their corresponding output is given as image below the codes.

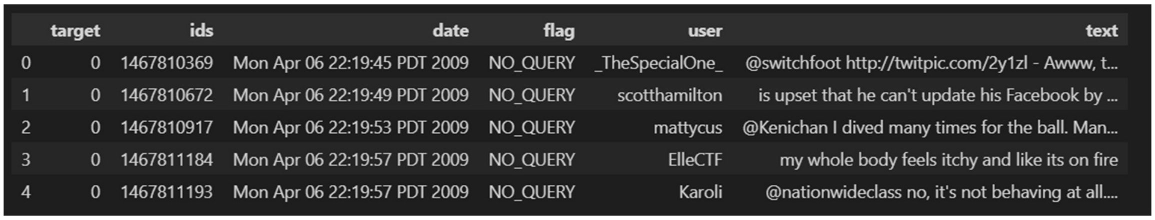
```
import re
import numpy as np
import pandas as pd
# plotting
import seaborn as sns
from wordcloud import WordCloud
import matplotlib.pyplot as plt
# nltk
from nltk.stem import WordNetLemmatizer
# sklearn
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report
from nltk.corpus import stopwords
import re
import string
import nltk
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
from textblob import TextBlob
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem.wordnet import WordNetLemmatizer
```



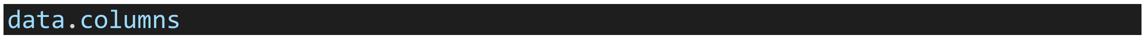
```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

```
DATASET_COLUMNS=['target','ids','date','flag','user','text']
DATASET_ENCODING = "ISO-8859-1"
data = pd.read_csv('twitter.csv', encoding=DATASET_ENCODING,
names=DATASET_COLUMNS)
data.sample(5)
X = data.iloc[:,[5]]
Y = data.iloc[:,0]
Y[Y == 4] = 1
```

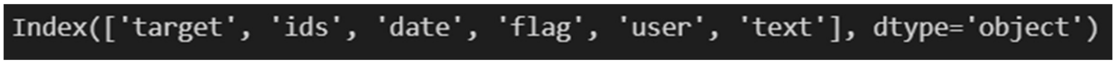
```
data.head()
```



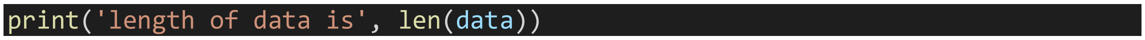
	target	ids	date	flag	user	text
0	0	1467810369	Mon Apr 06 22:19:45 PDT 2009	NO_QUERY	_TheSpecialOne_	@switchfoot http://twitpic.com/2y1zl - Awww, t...
1	0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by ...
2	0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Man...
3	0	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire
4	0	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all....



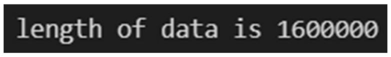
```
data.columns
```



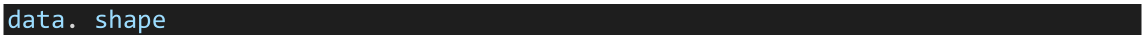
```
Index(['target', 'ids', 'date', 'flag', 'user', 'text'], dtype='object')
```



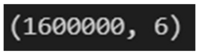
```
print('length of data is', len(data))
```



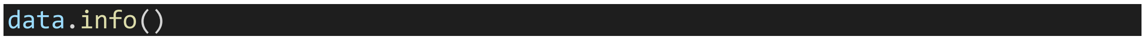
```
length of data is 1600000
```



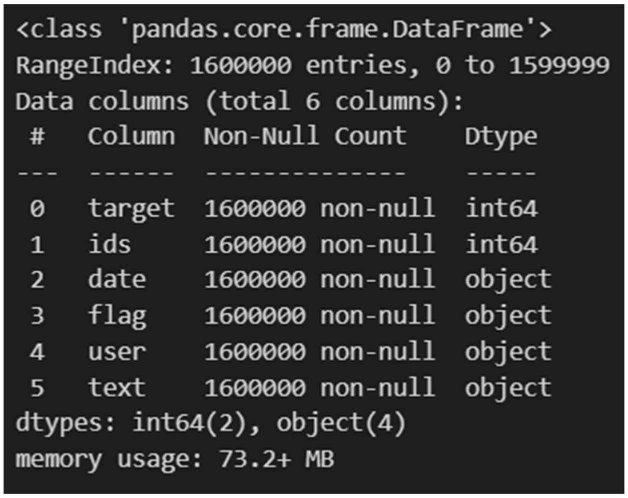
```
data. shape
```



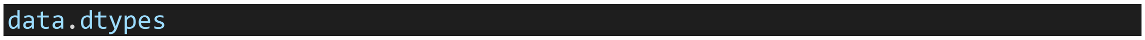
```
(1600000, 6)
```



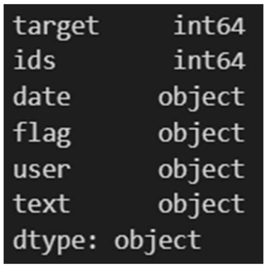
```
data.info()
```



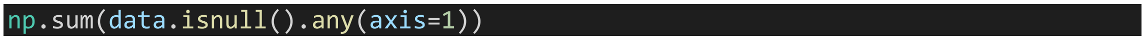
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1600000 entries, 0 to 1599999
Data columns (total 6 columns):
 #   Column  Non-Null Count    Dtype 
---  ------  --------------    ----- 
 0   target  1600000 non-null  int64 
 1   ids     1600000 non-null  int64 
 2   date    1600000 non-null  object
 3   flag    1600000 non-null  object
 4   user    1600000 non-null  object
 5   text    1600000 non-null  object
dtypes: int64(2), object(4)
memory usage: 73.2+ MB
```



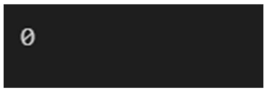
```
data.dtypes
```



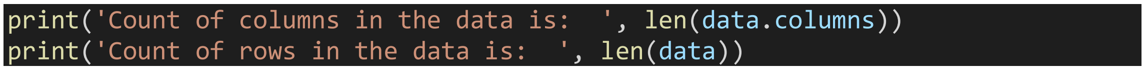
```
target     int64
ids        int64
date      object
flag      object
user      object
text      object
dtype: object
```



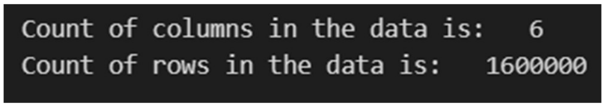
```
np.sum(data.isnull().any(axis=1))
```



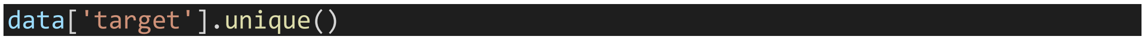
```
0
```



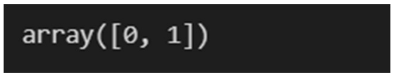
```
print('Count of columns in the data is:  ', len(data.columns))
print('Count of rows in the data is:  ', len(data))
```



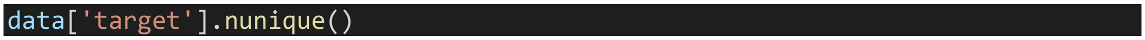
```
Count of columns in the data is:   6
Count of rows in the data is:   1600000
```



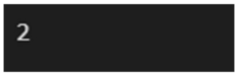
```
data['target'].unique()
```



```
array([0, 1])
```

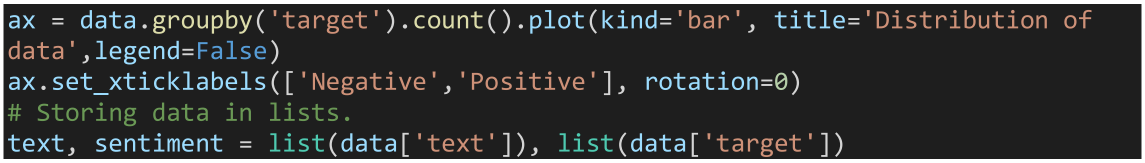


```
data['target'].nunique()
```

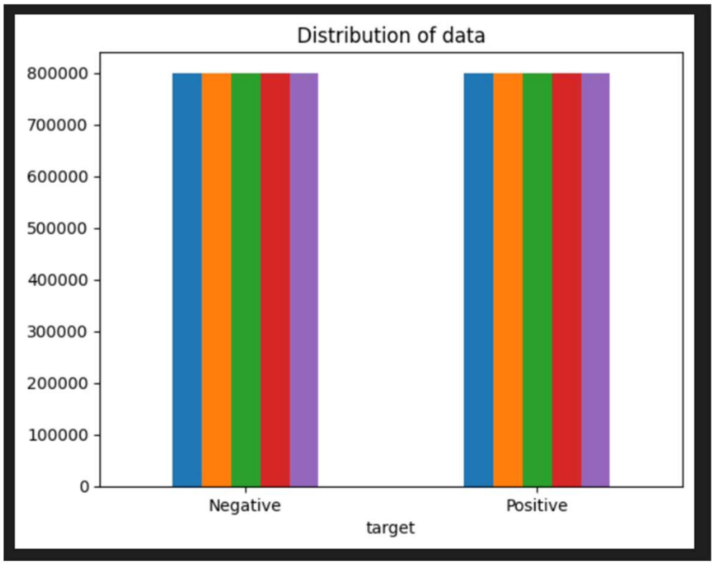


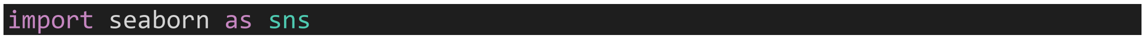
```
2
```

Distribution of Data

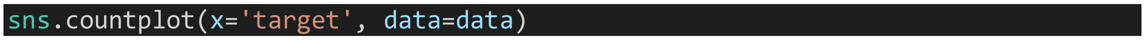


```
ax = data.groupby('target').count().plot(kind='bar', title='Distribution of
data',legend=False)
ax.set_xticklabels(['Negative','Positive'], rotation=0)
# Storing data in lists.
text, sentiment = list(data['text']), list(data['target'])
```

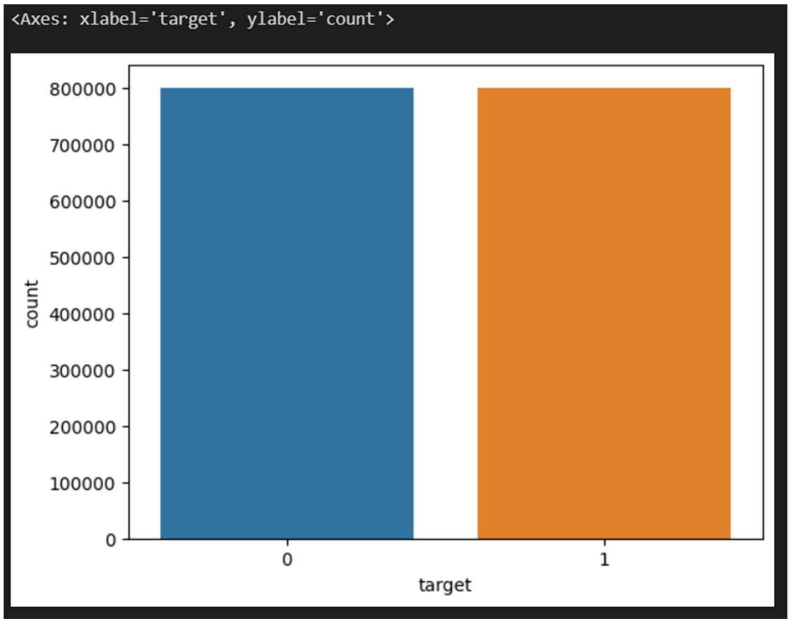




```
import seaborn as sns
```

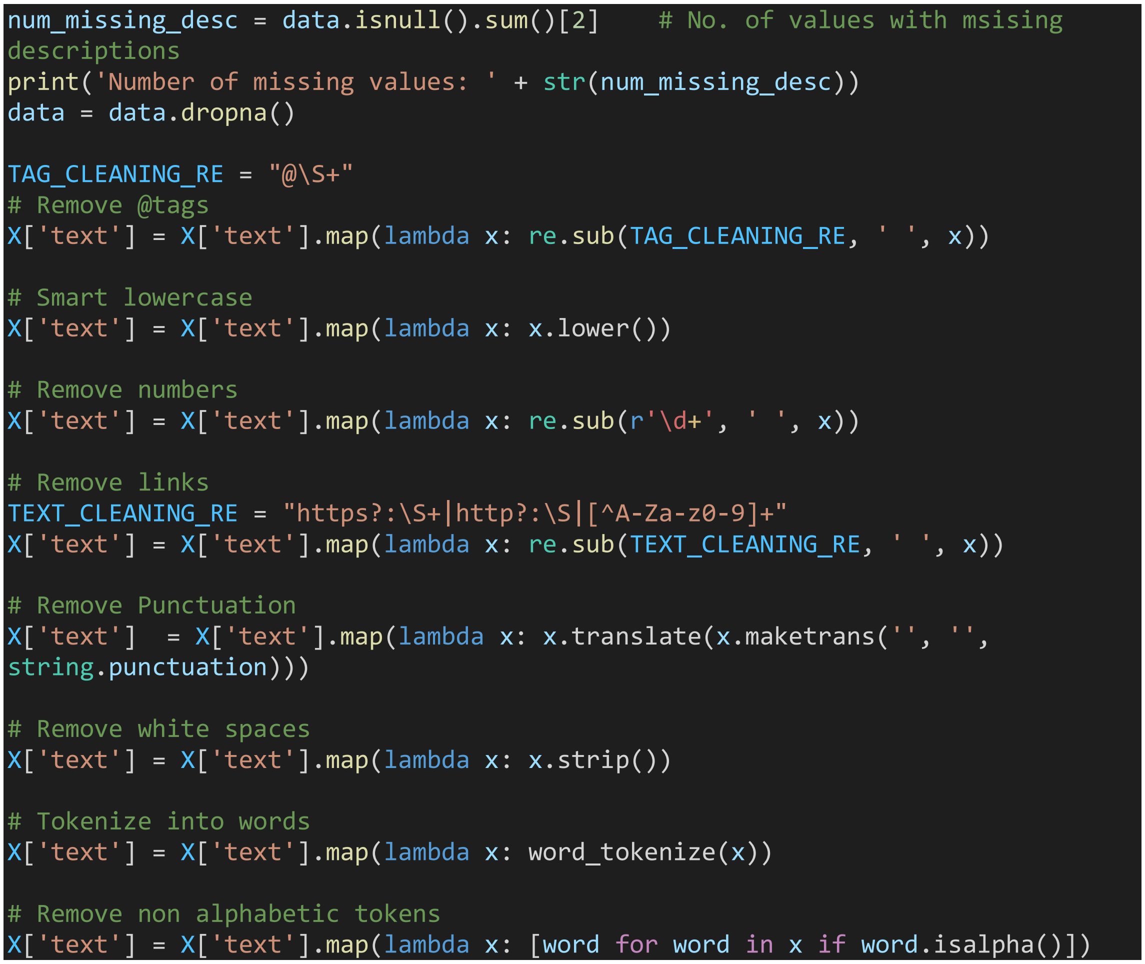


```
sns.countplot(x='target', data=data)
```

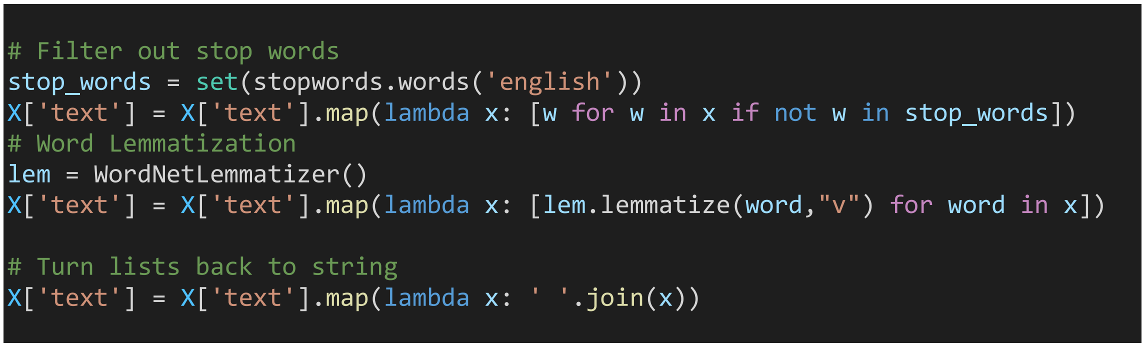


```
<Axes: xlabel='target', ylabel='count'>
```

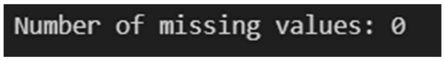
Preprocessing of Text



```
num_missing_desc = data.isnull().sum()[2]    # No. of values with msising
descriptions
print('Number of missing values: ' + str(num_missing_desc))
data = data.dropna()

TAG_CLEANING_RE = "@\S+"
# Remove @tags
X['text'] = X['text'].map(lambda x: re.sub(TAG_CLEANING_RE, ' ', x))

# Smart lowercase
X['text'] = X['text'].map(lambda x: x.lower())

# Remove numbers
X['text'] = X['text'].map(lambda x: re.sub(r'\d+', ' ', x))

# Remove links
TEXT_CLEANING_RE = "https?:\S+|http?:\S|[^A-Za-z0-9]+"
X['text'] = X['text'].map(lambda x: re.sub(TEXT_CLEANING_RE, ' ', x))

# Remove Punctuation
X['text']  = X['text'].map(lambda x: x.translate(x.maketrans('', '',
string.punctuation)))

# Remove white spaces
X['text'] = X['text'].map(lambda x: x.strip())

# Tokenize into words
X['text'] = X['text'].map(lambda x: word_tokenize(x))

# Remove non alphabetic tokens
X['text'] = X['text'].map(lambda x: [word for word in x if word.isalpha()])
```

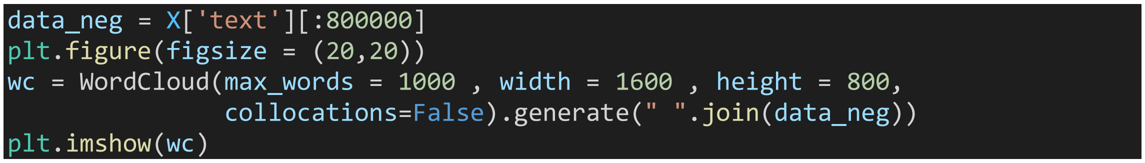


```
# Filter out stop words
stop_words = set(stopwords.words('english'))
X['text'] = X['text'].map(lambda x: [w for w in x if not w in stop_words])
# Word Lemmatization
lem = WordNetLemmatizer()
X['text'] = X['text'].map(lambda x: [lem.lemmatize(word,"v") for word in x])

# Turn lists back to string
X['text'] = X['text'].map(lambda x: ' '.join(x))
```

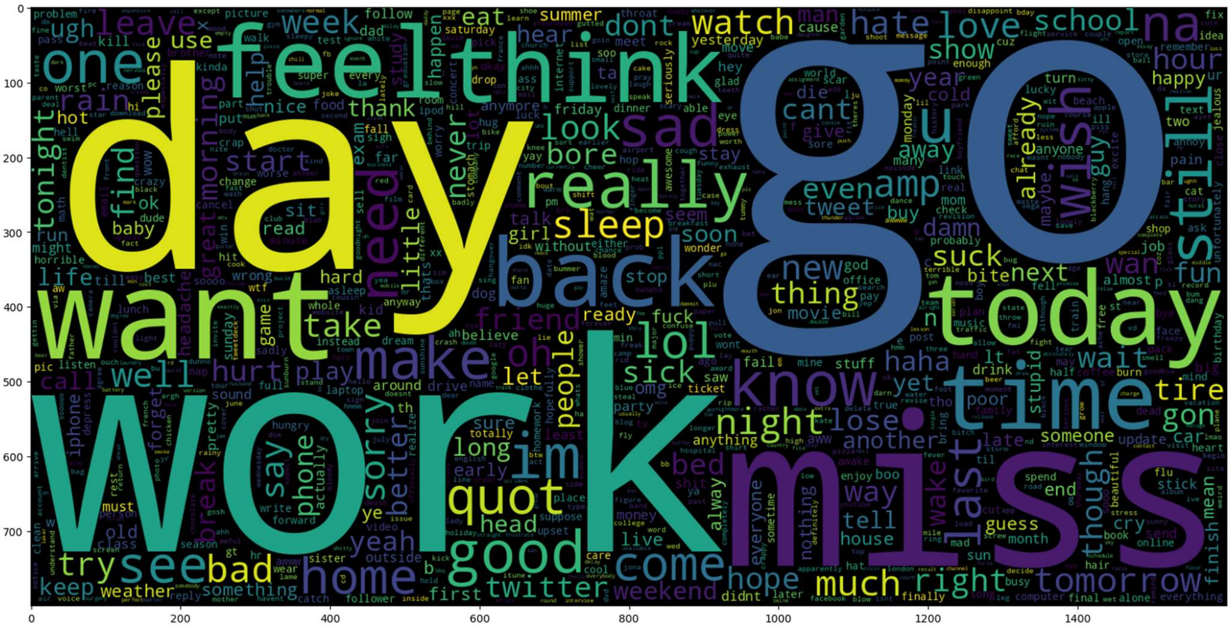


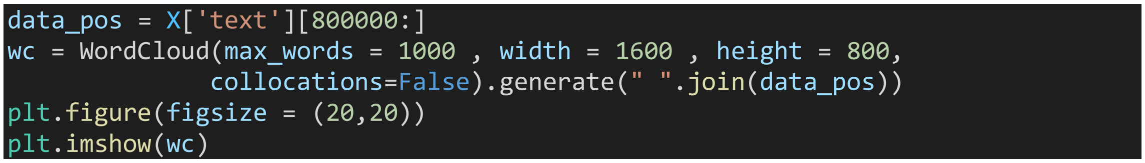
```
Number of missing values: 0
```

Generating Word Cloud

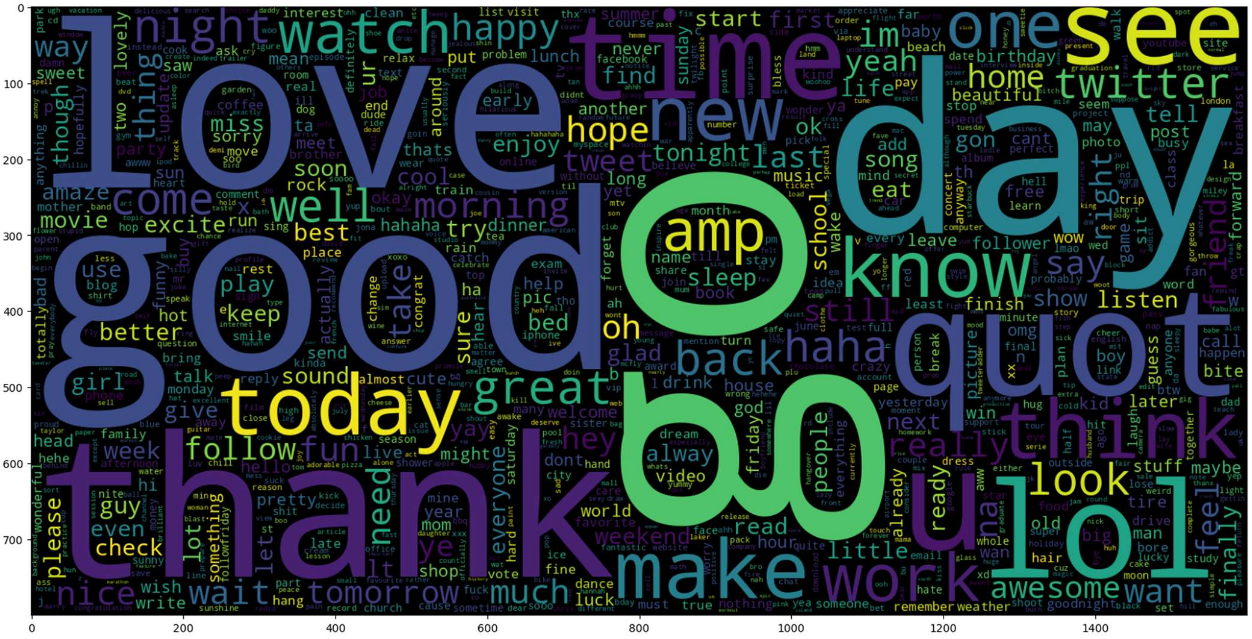


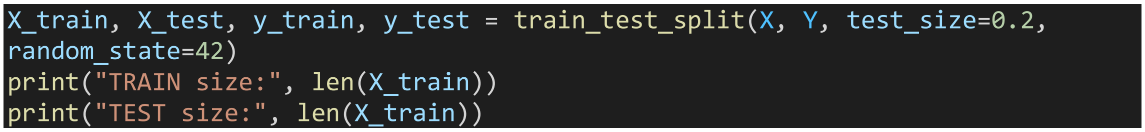
```
data_neg = X['text'][:800000]
plt.figure(figsize = (20,20))
wc = WordCloud(max_words = 1000 , width = 1600 , height = 800,
               collocations=False).generate(" ".join(data_neg))
plt.imshow(wc)
```



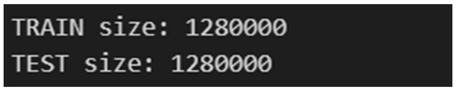


```
data_pos = X['text'][800000:]
wc = WordCloud(max_words = 1000 , width = 1600 , height = 800,
              collocations=False).generate(" ".join(data_pos))
plt.figure(figsize = (20,20))
plt.imshow(wc)
```

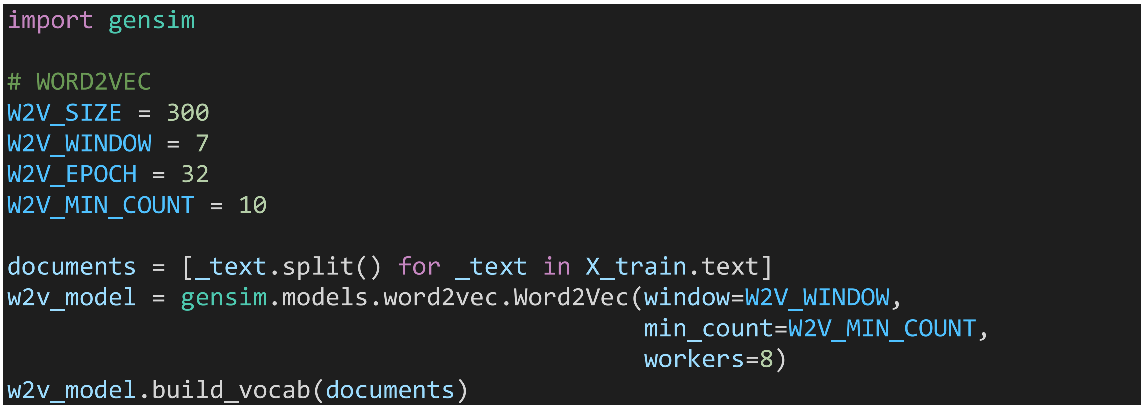




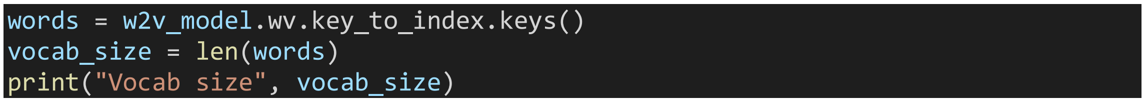
```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2,
random_state=42)
print("TRAIN size:", len(X_train))
print("TEST size:", len(X_train))
```



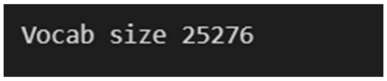
```
TRAIN size: 1280000
TEST size: 1280000
```



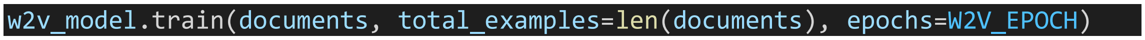
```
import gensim

# WORD2VEC
W2V_SIZE = 300
W2V_WINDOW = 7
W2V_EPOCH = 32
W2V_MIN_COUNT = 10

documents = [_text.split() for _text in X_train.text]
w2v_model = gensim.models.word2vec.Word2Vec(window=W2V_WINDOW,
                                            min_count=W2V_MIN_COUNT,
                                            workers=8)
w2v_model.build_vocab(documents)
```



```
words = w2v_model.wv.key_to_index.keys()
vocab_size = len(words)
print("Vocab size", vocab_size)
```



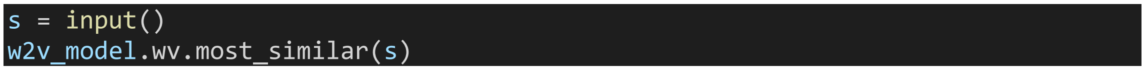
```
Vocab size 25276
```



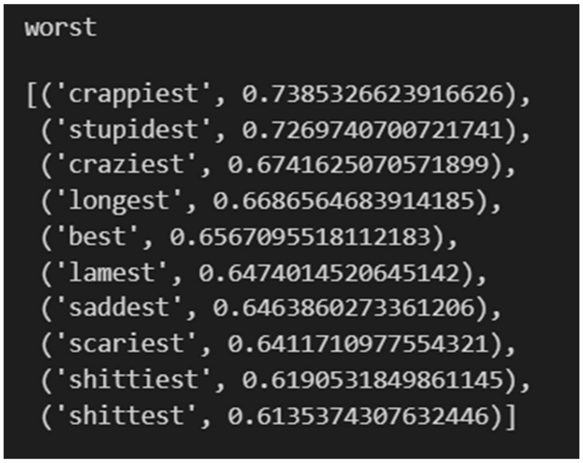
```
w2v_model.train(documents, total_examples=len(documents), epochs=W2V_EPOCH)
```

```
(251367682, 289225504)
```

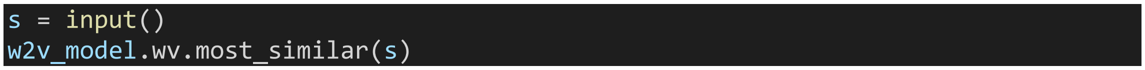
Finding similar words



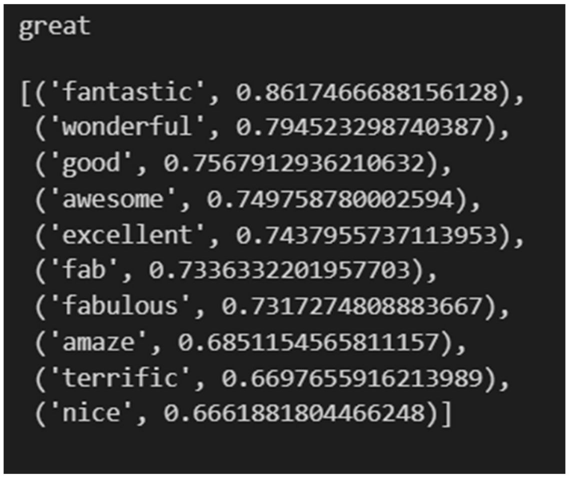
```
s = input()
w2v_model.wv.most_similar(s)
```



```
worst

[('crappiest', 0.7385326623916626),
 ('stupidest', 0.7269740700721741),
 ('craziest', 0.6741625070571899),
 ('longest', 0.6686564683914185),
 ('best', 0.6567095518112183),
 ('lamest', 0.6474014520645142),
 ('saddest', 0.6463860273361206),
 ('scariest', 0.6411710977554321),
 ('shittiest', 0.6190531849861145),
 ('shittest', 0.6135374307632446)]
```



```
s = input()
w2v_model.wv.most_similar(s)
```



```
great

[('fantastic', 0.8617466688156128),
 ('wonderful', 0.794523298740387),
 ('good', 0.7567912936210632),
 ('awesome', 0.749758780002594),
 ('excellent', 0.7437955737113953),
 ('fab', 0.7336332201957703),
 ('fabulous', 0.7317274808883667),
 ('amaze', 0.6851154565811157),
 ('terrific', 0.6697655916213989),
 ('nice', 0.6661881804466248)]
```

Result

Twitter Sentiment Analysis is done, wordclouds are generated, similar words are found.

Github Links

Name – Naman Anand
Reg.No. – RA2011029010013
Section – N2
Github Link:

https://github.com/Naman-anand88/ML_Project_RA2011029010013

Name – Srinivas T.B.
Reg.No. – RA2011029010015
Section – N2
Github Link:

https://github.com/notahuman-1-0/ML_Project-RA2011029010015

Name – Kirti Kalal
Reg.No. – RA2011029010031
Section – N2
Github Link:

https://github.com/kirtikalal/MLProject_RA2011029010031

CONCLUSION AND FUTURE WORKS

Twitter sentiment analysis has become a popular application of Natural Language Processing (NLP) techniques. It involves using algorithms and machine learning models to analyze the sentiment of tweets, which can be useful for various applications, such as brand monitoring, product analysis, and customer service.

Sentiment analysis on Twitter using word clouds and similar word analysis can provide valuable insights into public opinion and sentiment related to a particular topic or product. This type of analysis can help businesses and organizations understand their customers or the general public and identify potential areas for improvement or opportunities.

A word cloud is a visual representation of the most frequently occurring words in a dataset, with the size of each word indicating its frequency. The word cloud can help to identify the most common themes and topics discussed on Twitter related to the sentiment of interest. By analyzing the word cloud, we can gain an understanding of the most common themes and topics associated with the sentiment of interest.

Also, finding similar words helps to find which words are more used in place of a word to represent an emotion.

Overall, sentiment analysis on Twitter using word clouds and similar word analysis can provide valuable insights into public opinion and sentiment related to a particular topic or product. This type of analysis can help businesses and organizations understand their customers or the general public and identify potential areas for improvement or opportunities.

Future Works

The future of Twitter sentiment analysis is promising, as the use of social media platforms continues to grow and evolve. Here are some potential developments that may shape the future of Twitter sentiment analysis:

1. Advances in Natural Language Processing (NLP) techniques: As NLP techniques continue to improve, sentiment analysis algorithms may become more accurate and reliable. This could lead to better insights and decision-making for businesses, organizations, and governments.

2. Integration with other data sources: Twitter sentiment analysis may be integrated with other data sources, such as customer feedback surveys or sales data, to provide a more comprehensive understanding of customer sentiment and behavior.

3. Real-time sentiment analysis: Real-time sentiment analysis can provide immediate insights into customer sentiment, allowing businesses and organizations to respond quickly to feedback and concerns.

4. Use of machine learning: Machine learning algorithms can learn from past data to improve sentiment analysis accuracy and adapt to changing language usage and trends.

5. Analysis of visual content: Twitter sentiment analysis may expand to include analysis of visual content, such as images and videos, to provide a more complete understanding of customer sentiment.

6. Sentiment analysis of multiple languages: As Twitter continues to be used globally, sentiment analysis may expand to include multiple languages, allowing businesses and organizations to analyze sentiment across multiple markets.

Overall, the future of Twitter sentiment analysis is likely to be shaped by advances in technology, the growth of social media platforms, and the increasing need for businesses, organizations, and governments to understand public sentiment and behavior. As sentiment analysis techniques continue to evolve and improve, businesses and organizations will be better equipped to make informed decisions and improve their products, services, and policies based on customer feedback and sentiment.

REFERENCES

1. Sentiment analysis (https://en.wikipedia.org/wiki/Sentiment_analysis)
2. Pandas Documentation (https://pandas.pydata.org/docs/)
3. Seaborn Documentation (https://seaborn.pydata.org/)
4. Wordcloud Documentation (https://pypi.org/project/wordcloud/)
5. Matplotlib.pyplot Documentation (https://matplotlib.org/3.5.3/api/_as_gen/matplotlib.pyplot.html)
6. NLTK Documentation (https://www.nltk.org/)
7. Scikit-learn Documentation (https://scikit-learn.org/stable/)
8. Re Documentation (https://docs.python.org/3/library/re.html)
9. Stopwords Documentation (https://pypi.org/project/stop-words/)
10. WordNetLemmatizer Documentation (https://www.nltk.org/_modules/nltk/stem/wordnet.html)