YOUTUBE SENTIMENT ANALYSIS

A PROJECT REPORT FOR J COMPONENT

SOCIAL AND INFORMATION NETWORKS

(CSE 3021)

PROJECT SUPERVISOR

MANJULA R

Submitted by

VEHAAN SINGH KUNDRA: 17BCE2050

SIDDHARTH JAIN: 17BCE0604

School of Computer Science and Engineering

Nov. 2019

CERTIFICATE

This is to certify that the project entitled, “YOUTUBE SENTIMENT ANALYSIS" submitted by Vehaan Singh Kundra(17BCE2050) and Siddharth Jain(17BCE0604) for the course "Social and information networks (CSE-3021)" at “VIT university", is an authentic work carried out by the members of our team. To the best of my knowledge, the matter embodied in the project has not been submitted to any other University / Institute for any purpose and is unique.

Date: 04 Nov 2019

Place: Vellore

ACKNOWLEDGEMENT

We would like to extend our thanks to the management of VIT University, Vellore and our respected chancellor G. Viswanathan for giving us the opportunity to carry out our project in VIT.

We would also like to extend our gratitude to our faculty, Prof. Manjula R for her constant and ceaseless guidance and support through the entire course of this project. We would also want to thank ma'am for guiding us all along the way and encouraging us to think out of the box in an attempt to maximize the project's validity for the society.

VEHAAN SINGH KUNDRA: 17BCE2050

SIDDHARTH JAIN: 17BCE0604

ABSTRACT

Sentiment analysis is the field of study identified with dissect conclusions, sentiments, assessments, demeanours, and feelings of clients which they express via web-based networking media furthermore, other online assets. The unrest of web-based life destinations has likewise pulled in the clients towards video sharing locales, for example, YouTube. The online clients express their conclusions or sentiments on the videos that they watch on such sites. We propose YouTube Sentiment analysis, which aims to find sentiment of comments scraped from a video.

TABLE OF CONTENTS

INTRODUCTION

The project works by scraping YouTube comments and identifying the sentiment of those comments. We used Naive Bayes technique for text classification. We used YouTube API to access comments. The YouTube Application Programming Interface, or the YouTube API, allows developers to access video statistics and YouTube channels' data.

We used Naive Bayes technique for text classification. Text classification aims to assign comments to categories. Naive Bayes classifiers are a collection of classification algorithms based on Bayes’ Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other

LITERATURE SURVEY

In the paper written by ‘Sentiment Analysis on YouTube: A Brief Survey’ the authors state that, the popularity of social media is increasing rapidly because it is easy in use and simple to create and share images, video even from those users who are technically unaware of social media. There are many web platforms that are used to share non-textual content such as videos, images, animations that allow users to add comments for each item. YouTube is probably the most popular of them, with millions of videos uploaded by its users and billions of comments for all of these videos. In social media especially in YouTube, detection of sentiment polarity is a very challenging task due to some limitations in current sentiment dictionaries. In present dictionaries there are no proper sentiments of terms created by community. It is clear from the studies conducted by that the web traffic is 20% and Internet traffic is 10% of the total YouTube traffic. There are many mechanisms of YouTube for the judgment of opinions and views of users on a video. These mechanisms include voting, rating, favourites, sharing. Analysis of user comments is a source through which useful data may be achieved for many applications.

WORK

PROPOSED PROBLEM STATEMENT:

Perform sentiment analysis of comments on YouTube videos.

PROPOSED SOLUTION:

Creating word cloud from frequently used words:
Comment downloader.py
The key variable stores the API key which allows the program to access the YouTube data. The API key is got through google developer account.
YOUTUBE_IN_LINK =
'https://www.googleapis.com/youtube/v3/commentThreads?part=snippet&maxResults=100&order=relevance&pageToken={pageToken}&videoId={videoId}&key={key}'
YOUTUBE_LINK =
'https://www.googleapis.com/youtube/v3/commentThreads?part=snippet&maxResults=100&order=relevance&videoId={videoId}&key={key}'
The 'YOUTUBE_LINK' string is used to create frame the video URL is in the video id and key. The request.get function is used to scrape the page info from the YouTube page of the video given as input. Comments array is created from the page info variable using the following code.

1. for i in range(len(page_info['items'])):
2. comments.append(page_info['items'][i]['snippet']['topLevelComment']['snippet']['textOriginal'])
3. co += 1
4. if co == count:
5. PB.progress(co, count, cond = True)
6. return comments

Fanysentiment.py
From the comments array we extract the English stop words using the stop words module from nltk.corpus library. Tokenization is a way to split text into tokens. These tokens could be paragraphs, sentences, or individual words. NLTK provides a number of tokenizers in the tokenize module. The words in the comments array are tokenized using the word_tokenize function and saved in the variable word. The stop words are extracted from the words variable above and stored as a string and the temporary string variable temp_filter. The temp_filter is appended to filtered_comments.
"filtered_comments_str = ' '.join(filtered_comments)" is used to join all the filtered comments. The word cloud function is used to create variable sentiment and the generate function is used to generate the word cloud.

training_classifier.py
A collection of bigram association measures. Each association measure is provided as a function with three arguments:

bigram_score_fn(n_ii, (n_ix, n_xi), n_xx)
The arguments constitute the marginals of a contingency table, counting the occurrences of particular events in a corpus.
Bigram counts count the frequency of pairs of characters. We have set the bigram count to 150. A tool for the finding and ranking of bigram collocations or other association measures. “BCF.from_words(words).nbest(scoreF, n)” return the top ngrams than scored by the given function. Itertool.chain function is used to make an iterator that returns elements from the first iterable until it’s exhausted, then proceeds to the next iterable until all of the iterables are exhausted. In the training function, we read the text files named ‘positive’, ‘negative’ and ‘emoji’ and create positive text containing positive text of emoji, negative text containing negative text of emoji. With the Naive Bayes model, we do not take only a small set of positive and negative words into account, but all words the NB Classifier was trained with, i.e. all words presents in the training set. If a word has not appeared in the training set, we have no data available and apply Laplacian smoothing.

Naive Bayes Algorithm:
Naive Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naive) independence assumptions between the features. Naive Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers.
Libraries used:

1. **NLTK:** The Natural Language Toolkit is a suite of libraries and programs for symbolic and statistical natural language processing (NLP) for English written in the Python programming language.
2. **pickle:** Pickling is a way to convert a python object (list, dict, etc.) into a character stream.
3. **itertools:** The python itertools module is a collection of tools for handling iterators simply put iterators are datatypes that are used in for loop. The most common iterator is list.
4. **statistics:** This module provides functions for calculating mathematical statistics of numeric (Real-valued) data.
5. **csv:** The csv module implements classes to read and write tabular data in CSV format. This has been used to read src.csv for the node and edge information.
6. **numpy:** NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
7. **lxml:** lxml is the most feature-rich and easy-to-use library for processing XML and HTML in the Python language.
8. **mathplotlib:** Matplotlib is a python library used to create 2D graphs and plots by using python scripts.
9. **wordcloud:** Word Cloud is a data visualization technique used for representing text data in which the size of each word indicates its frequency or importance.
10. **requests:** Requests will allow you to send HTTP/1.1 requests using Python. With it, you can add content like headers, form data, multipart files, and parameters via simple Python libraries.
11. **time:** This module provides various time-related functions.

12. **cis:** CIS is an open source Python library and command-line tool for the easy collocation, visualization, analysis, and comparison of a diverse set of gridded and ungridded datasets used across earth sciences.

CODE

Youtube_stats.py

```
__author__ = 'user'
from
googleapiclient.disco
very import build

YOUTUBE_API_SERVI
CE_NAME =
"youtube"
YOUTUBE_API_VERSI
ON = "v3"
DEVELOPER_KEY =
"AIzaSyDhWZjOQ_lC
KHbJJnE14fKx1Vor8d
CBBFI"


def
get_likes_dislikes(vid
eo_id):
   youtube =
build(YOUTUBE_API_
SERVICE_NAME,
YOUTUBE_API_VERSI
ON,
developerKey=DEVEL
OPER_KEY)
   results =
youtube.videos().list(
      part =
"statistics",
      id = video_id,
   ).execute()

   likes =
int(results["items"][0
]["statistics"]["likeCo
unt"])
   dislikes =
int(results["items"][0
]["statistics"]["dislike
Count"])

    return likes,dislikes
```

filter.py

```
 __author__ = 'user'

from nltk.corpus
import stopwords
from nltk.tokenize
import
word_tokenize

def
filter_comments(com
ments):
    for i in
range(0,len(commen
ts)):
        comments[i] =
comments[i].strip('\u
feff')
        comments[i] =
comments[i].lower()


    stop_words =
set(stopwords.words
('english'))
    filtered_comments
= []
    for comment in
comments:
        word_tokens =
word_tokenize(com
ment)
        links = [w for w
in word_tokens if
w.startswith('www.')
or
w.startswith('http')]
        mentions = [w
for w in word_tokens
if w.startswith('@')]
```

```
        filter = [w for w
in word_tokens if w
not in stop_words
and w not in links
and w not in
mentions]
        #filter = [w for w
in word_tokens if w
not in links and w not
in mentions]

filtered_comments.a
ppend(filter)

    return
filtered_comments
```

**youtube_comments.
py**

```
__author__ = 'user'

import sys
import time
import json
import requests
import lxml.html

from lxml.cssselect import CSSSelector

YOUTUBE_COMMENTS_URL = 'https://www.youtube.com/all_comments?v={youtube_id}'
YOUTUBE_COMMENTS_AJAX_URL = 'https://www.youtube.com/comment_ajax'

USER_AGENT = 'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/48.0.2564.116 Safari/537.36'


def find_value(html, key, num_chars=2):
    pos_begin = html.find(key) + len(key) + num_chars
    pos_end = html.find('"', pos_begin)
    return html[pos_begin: pos_end]


def extract_comments(html):
    tree = lxml.html.fromstring(html)
    item_sel = CSSSelector('.comment-item')
    text_sel = CSSSelector('.comment-text-content')
    time_sel = CSSSelector('.time')
    author_sel = CSSSelector('.user-name')

    for item in item_sel(tree):
        id = item.get('data-cid')
        if '.' not in id:
            yield {'cid': item.get('data-cid'),
                   'text': text_sel(item)[0].text_content(),
                   'time': time_sel(item)[0].text_content().strip(),
                   'author': author_sel(item)[0].text_content()}


def extract_reply_cids(html):
    tree = lxml.html.fromstring(html)
    sel = CSSSelector('.comment-replies-header > .load-comments')
    return [i.get('data-cid') for i in sel(tree)]


def ajax_request(session, url, params, data, retries=10, sleep=20):
    for _ in range(retries):
        response = session.post(url, params=params, data=data)
        if response.status_code == 200:
            response_dict = json.loads(response.text)
            return response_dict.get('page_token', None), response_dict['html_content']
        else:
            time.sleep(sleep)


def download_comments(youtube_id, sleep=1):
    session = requests.Session()
    session.headers['User-Agent'] = USER_AGENT

    # Get Youtube page with initial comments
    response = session.get(YOUTUBE_COMMENTS_URL.format(youtube_id=youtube_id))
    html = response.text
    reply_cids = extract_reply_cids(html)

    ret_cids = []
    for comment in extract_comments(html):
        ret_cids.append(comment['cid'])
        yield comment

    page_token = find_value(html, 'data-token')
    session_token = find_value(html, 'XSRF_TOKEN', 4)

    first_iteration = True
```

```
# Get remaining comments (the same as pressing the 'Show more' button)
while page_token:
    data = {'video_id': youtube_id,
            'session_token': session_token}

    params = {'action_load_comments': 1,
              'order_by_time': False,
              'filter': youtube_id}

    if first_iteration:
        params['order_menu'] = True
    else:
        data['page_token'] = page_token

    response = ajax_request(session, YOUTUBE_COMMENTS_AJAX_URL, params, data)
    if not response:
        break

    page_token, html = response

    reply_cids += extract_reply_cids(html)
    for comment in extract_comments(html):
        if comment['cid'] not in ret_cids:
            ret_cids.append(comment['cid'])
            yield comment

    first_iteration = False
    time.sleep(sleep)

# Get replies (the same as pressing the 'View all X replies' link)
for cid in reply_cids:
    data = {'comment_id': cid,
            'video_id': youtube_id,
            'can_reply': 1,
            'session_token': session_token}

    params = {'action_load_replies': 1,
              'order_by_time': False,
              'filter': youtube_id,
              'tab': 'inbox'}

    response = ajax_request(session, YOUTUBE_COMMENTS_AJAX_URL, params, data)
    if not response:
        break

    _, html = response
```

```
        for comment in extract_comments(html):
            if comment['cid'] not in ret_cids:
                ret_cids.append(comment['cid'])
                yield comment
        time.sleep(sleep)


def get_youtube_comments(video_id):


    try:

        print('Downloading Youtube comments for video:', video_id)
        count = 0
        youtube_comments = []

        for comment in download_comments(video_id):
            youtube_comments.append(comment['text'])
            #print(json.dumps(comment['text']), file=fp)
            count += 1
            if count >= 200:
                break
            sys.stdout.write('Downloaded %d comment(s)\r' % count)
            sys.stdout.flush()
        #print("\nDone")

        return youtube_comments

    except Exception as e:
        print('Error:', str(e))
        sys.exit(1)
```

training.py

```
__author__ = 'user'

import nltk
import random
import pickle
import os.path
from nltk.corpus import movie_reviews


word_features = []
```

```
def find_features(document):
    words = set(document)
    features = {}
    for w in word_features:
        features[w] = (w in words)

    return features



def train_classifier():
    if(os.path.isfile("classifier.pickle")):
        return

    documents = []
    for category in movie_reviews.categories():
        for fileid in movie_reviews.fileids(category):
            documents.append((list(movie_reviews.words(fileid)),category))

    #random.shuffle(documents)

    all_words = []
    for w in movie_reviews.words():
        all_words.append(w.lower())

    all_words = nltk.FreqDist(all_words)

    for w in all_words.most_common(9000):
        if(len(w[0]) >= 3):
            word_features.append(w[0])

    feature_sets = [(find_features(rev), category) for (rev, category) in documents]

    random.shuffle(feature_sets)

    #classifier = nltk.NaiveBayesClassifier.train(feature_sets[:2000])
    classifier = nltk.NaiveBayesClassifier.train(feature_sets[:2000])

    #print(nltk.classify.accuracy(classifier,feature_sets[1500:])*100)

    save_classifier = open("classifier.pickle","wb")
    pickle.dump(classifier,save_classifier)
    save_classifier.close()
```

sentiments.py

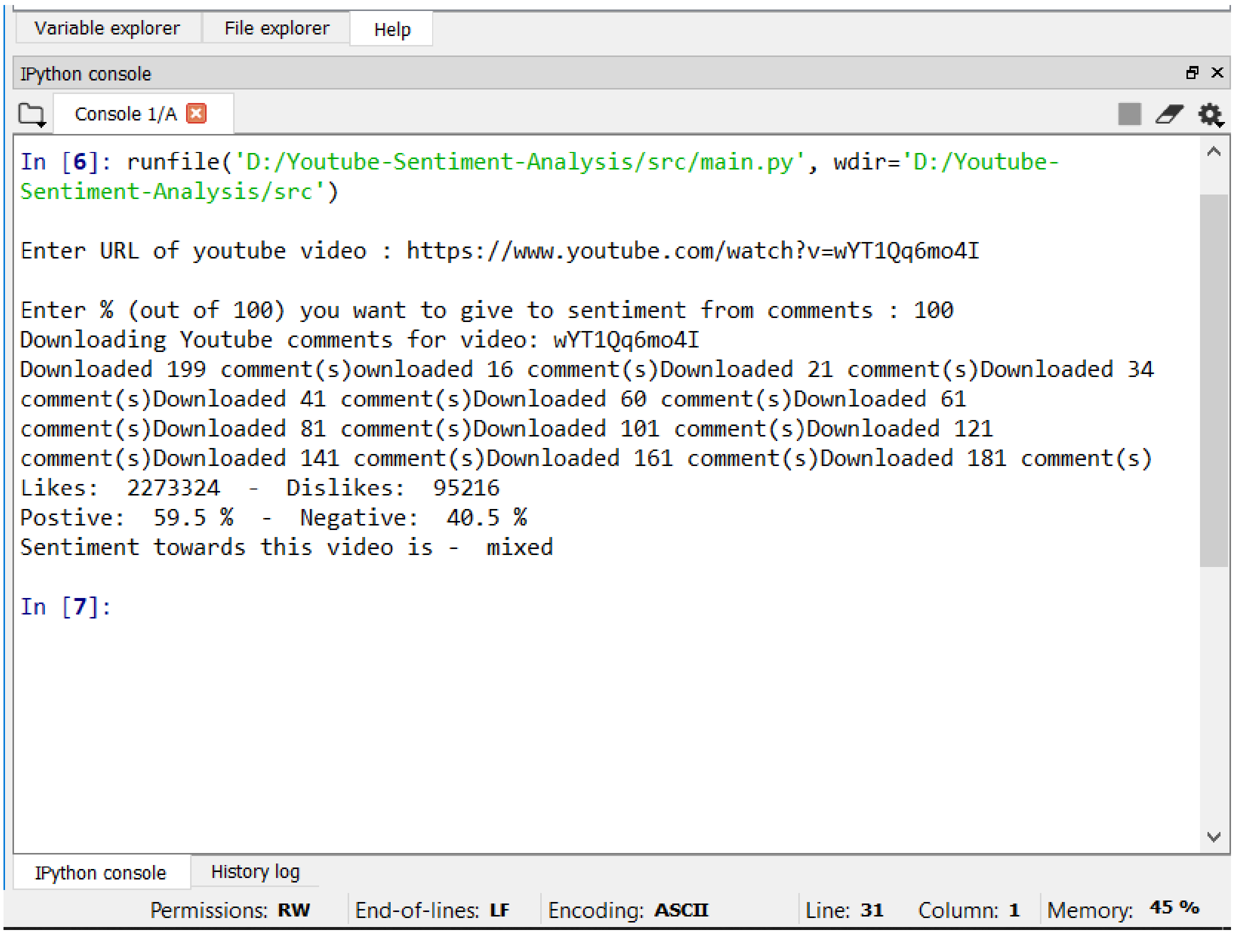
```
__author__ = 'user'

from src import youtube_comments
from src import filter
from src import training
from src import youtube_stats

import pickle


def calculate_score(pos,neg,likes,dislikes,comments_ratio):
    pos_percent = None
    neg_percent = None

    if pos == 0 and neg == 0:
        pos_percent = 0
        neg_percent = 0
    elif pos == 0:
        pos_percent = 0
        neg_percent = 100
    elif neg == 0:
        pos_percent = 100
        neg_percent = 0
    else:
        pos_percent = (pos/(pos+neg))*100
        neg_percent = 100-pos_percent

    likes_percent = (likes/(likes+dislikes))*100
    dislikes_percent = 100 - likes_percent

    pos_percent = pos_percent * (comments_ratio/100)
    neg_percent = neg_percent * (comments_ratio/100)

    likes_dislikes_ratio = 1-(comments_ratio/100)

    likes_percent *= likes_dislikes_ratio
    dislikes_percent *= likes_dislikes_ratio

    pos_sentiment = pos_percent + likes_percent
    neg_sentiment = neg_percent + dislikes_percent

    print("Postive: ", pos_sentiment,"%"," - ","Negative: ", neg_sentiment, "%")

    if pos_sentiment > 45 and pos_sentiment < 60:
        return "mixed"
    elif pos_sentiment >= 60:
        return "positive"
    else:
        return "negative"


def bag_of_words(words):
    return dict([word, True] for word in words)

def get_sentiment(video_id,comments_ratio):
    comments = youtube_comments.get_youtube_comments(video_id)

    pos = 0
    neg = 0


    filtered_comments = filter.filter_comments(comments)

    training.train_classifier()

    classifier_f = open("classifier.pickle", "rb")
    classifier = pickle.load(classifier_f)
    classifier_f.close()

    #print(classifier.show_most_informative_features(20))

    for comment in filtered_comments:
        result = classifier.classify(bag_of_words(comment))
        #print(comment," -> ",result)
        if result == "pos":
            pos += 1
        else:
            neg += 1

    #print("pos:", pos, " - ", "neg: ", neg)


    no_of_likes , no_of_dislikes = youtube_stats.get_likes_dislikes(video_id)

    print()
    print("Likes: ", no_of_likes," - ","Dislikes: ", no_of_dislikes)

    result = calculate_score(pos,neg,no_of_likes,no_of_dislikes,comments_ratio)

    return result
```

main.py

```
__author__ = 'user'

from src import sentiment
import re

def extract_video_id(url):
    pattern1  = "^https://www\.youtube\.com/watch\?v=.*$"
    pattern2  = "www\.youtube\.com/watch\?v=.*$"
    if re.match(pattern1,url) or re.match(pattern2,url):
        index = url.find('=')
        video_id = url[index+1:]
        return video_id
    else:
        return None


if __name__ == "__main__":
    youtube_url = input("Enter URL of youtube video : ")

    comments_ratio = float(input("Enter % (out of 100) you want to give to sentiment from
comments : "))

    if comments_ratio > 100 or comments_ratio < 0:
        print("Invalid numeric")
        exit(-1)

    video_id = extract_video_id(youtube_url)

    if video_id is None:
        print("Invalid URL")

    else:
        print("Sentiment towards this video is -
",sentiment.get_sentiment(video_id,comments_ratio))
```

OUTPUT:

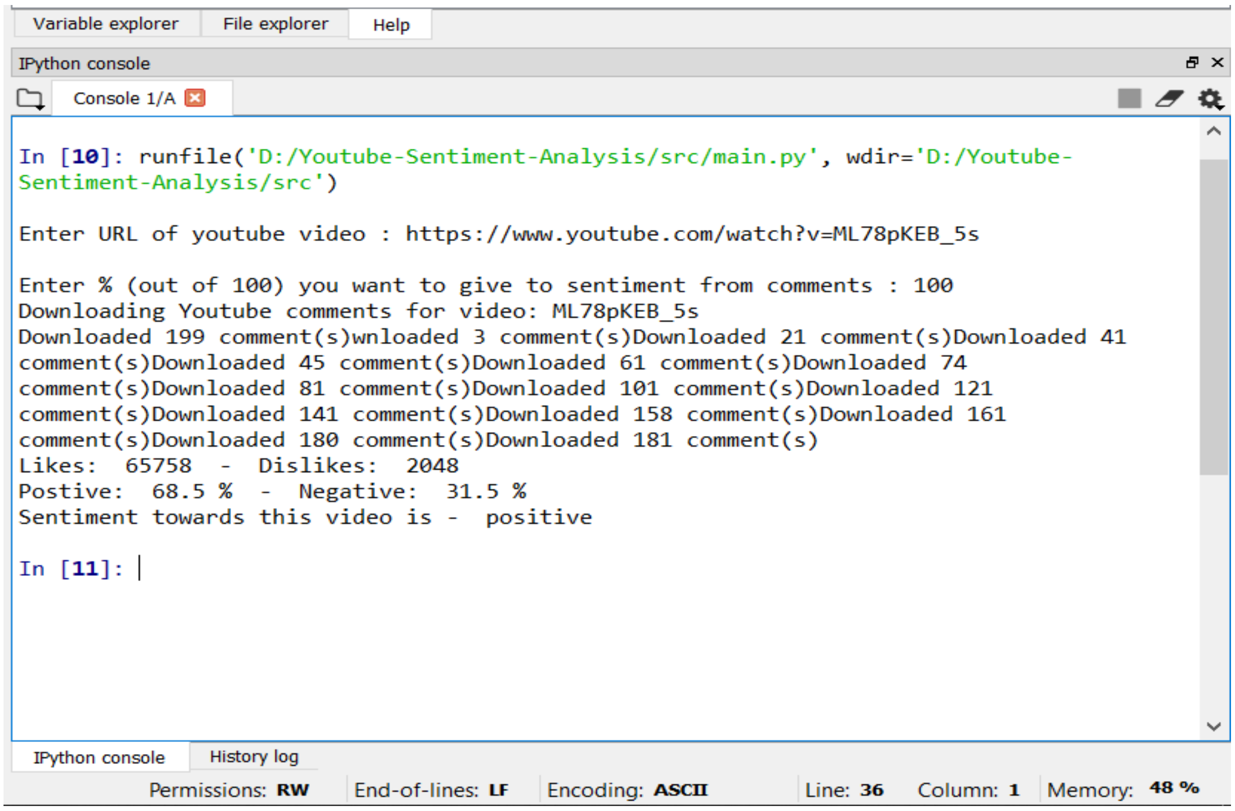
YOUTUBE VIDEO LINK: https://www.youtube.com/watch?v=wYT1Qq6mo4I



```
In [6]: runfile('D:/Youtube-Sentiment-Analysis/src/main.py', wdir='D:/Youtube-
Sentiment-Analysis/src')

Enter URL of youtube video : https://www.youtube.com/watch?v=wYT1Qq6mo4I

Enter % (out of 100) you want to give to sentiment from comments : 100
Downloading Youtube comments for video: wYT1Qq6mo4I
Downloaded 199 comment(s)ownloaded 16 comment(s)Downloaded 21 comment(s)Downloaded 34
comment(s)Downloaded 41 comment(s)Downloaded 60 comment(s)Downloaded 61
comment(s)Downloaded 81 comment(s)Downloaded 101 comment(s)Downloaded 121
comment(s)Downloaded 141 comment(s)Downloaded 161 comment(s)Downloaded 181 comment(s)
Likes:  2273324  -  Dislikes:  95216
Postive:  59.5 %  -  Negative:  40.5 %
Sentiment towards this video is -  mixed

In [7]:
```

Permissions: RW End-of-lines: LF Encoding: ASCII Line: 31 Column: 1 Memory: 45 %

YOUTUBE VIDEO LINK: https://www.youtube.com/watch?v=ML78pKEB_5s



```
In [10]: runfile('D:/Youtube-Sentiment-Analysis/src/main.py', wdir='D:/Youtube-
Sentiment-Analysis/src')

Enter URL of youtube video : https://www.youtube.com/watch?v=ML78pKEB_5s

Enter % (out of 100) you want to give to sentiment from comments : 100
Downloading Youtube comments for video: ML78pKEB_5s
Downloaded 199 comment(s)wnloaded 3 comment(s)Downloaded 21 comment(s)Downloaded 41
comment(s)Downloaded 45 comment(s)Downloaded 61 comment(s)Downloaded 74
comment(s)Downloaded 81 comment(s)Downloaded 101 comment(s)Downloaded 121
comment(s)Downloaded 141 comment(s)Downloaded 158 comment(s)Downloaded 161
comment(s)Downloaded 180 comment(s)Downloaded 181 comment(s)
Likes:  65758  -  Dislikes:  2048
Postive:  68.5 %  -  Negative:  31.5 %
Sentiment towards this video is -  positive

In [11]: |
```

Permissions: RW End-of-lines: LF Encoding: ASCII Line: 36 Column: 1 Memory: 48 %

CONCLUSION:

We scraped comments from YouTube using video id of videos. We performed positive and negative sentiment analysis on comments. We calculated accuracy, positive sentiment and negative sentiment.

REFERENCES:

1. Sentiment Analysis on YouTube: A Brief Survey
2. Retrieving YouTube video by sentiment analysis on user comment
3. Sentiment analysis for YouTube channels – with NLTK
4. Retrieving YouTube video by sentiment analysis on user comment
5. Naive Bayes Tutorial: Naive Bayes Classifier in Python
6. How To Implement Naive Bayes From Scratch in Python
7. Removing stop words with NLTK in Python