CSE3024
Web Mining

Topic: Sentiment Analysis of YouTube Videos

Name	Registration No.
Rohitanshu Kar	15-BCE-0550
Nilesh Kumar Raut	15-BCE-0607
Saksham Agarwal	15-BCE-0844
Sharad Jha	15-BCE-2034
Shantanu Pramanik	15-BCE-2071
Slot(s): B1 + TB1	

Faculty: *Prof. Shashank Mouli Satapathy*

School of Computer Science and Engineering

VIT University

Vellore

Table of Content

(1) Abstract

Sentiment analysis or opinion mining is used to automate the detection of subjective information such as opinions, attitudes, emotions, and feelings. Hundreds of thousands care about scientific research and take a long time to select suitable papers for their research. Online reviews on papers are the essential source to help them. The reviews save reading time and save papers cost.

The opinions of others have a significant influence in our daily decision-making process. These decisions range from buying a product such as a smart phone to making investments to choosing a school—all decisions that affect various aspects of our daily life. Before the Internet, people would seek opinions on products and services from sources such as friends, relatives, or consumer reports.

(2) Introduction

World Wide Web (**www**) has become the most popular communication platforms to the public reviews, opinions, comments and sentiments about products, places, scientific books or papers and to daily text reviews. The number of active user bases and the size of their reviews created daily on online websites are massive. There are 2.4 billion active online users, who write and read online and Internet usage around the world. Scientific research domain has a big world in journals and conferences, there are more than 4000 rated conferences and 5000 ranked journals. Each one of them has thousand number of papers such as ACM, Springer and Science direct. Notably, a large fragment of WWW researchers makes their content public, allowing researchers, societies, and universities, corporations to use and analyse data. According to a new survey conducted by Dimensional Research, April 2013: *90% of customer's decisions depends on Online Reviews*. According to 2013 Study: *79% of customer's confidence* *is based on online personal recommendation reviews*. As the result, a large number of studies and research have monitored the trending new research.

Recently, several websites encourage researchers to express and exchange their views, suggestions and opinions related to scientific papers. Sentiment analysis depends on two issues sentiment polarity and sentiment score. Sentiment polarity is a binary value either positive or negative. On the other hand, sentiment score which relies on one of three models. Those models are Bag-of-words model (**BOW**), part of speech (**POS**), and semantic relationships.

Sentiment analysis or *opinion mining* is the process of identifying and detecting subjective information using natural language processing, text analysis, and computational linguistics. In short, the aim of sentiment analysis is to extract information on the attitude of the writer or speaker towards a specific topic or the total polarity of a document. The first papers that used sentiment analysis among their keywords were published about a decade ago, but the field can trace its roots back to the middle of the 19th century. One of the pioneering resources for sentiment analysis is the General Inquirer. Although it was launched already in the 1960s, it is still being maintained. Sentiment identification is a very complex problem, and thus much effort has been put into analysing and trying to understand its different aspects, see for instance. Common sources of opinionated texts have been movie and product reviews, blogs and Twitter posts. As news stories have traditionally been considered neutral and free from sentiments, little focus has been on them. However, the interest in this domain is growing, as automated trading algorithms account for an ever-increasing part of the trade. A fast and simple method for determining the sentiment of a text is using a pre-defined collection of sentiment-bearing words and simply aggregating the sentiments found.

2.1 YouTube

YouTube is an American video-sharing website headquartered in San Bruno, California. Google bought the site in November 2006 for US$1.65 billion; YouTube now operates as one of Google's subsidiaries. YouTube allows users to upload, view, rate, share, add to favourites, report, comment on videos, and subscribe to other users. Available content includes video clips, TV show clips, music videos, short and documentary films, audio recordings, movie trailers, live streams, and other content such as video blogging, short original videos and educational videos.

Founders	Chad Hurley, Steve Chen & Jawed Karim
Founded	February 14, 2005
CEO	Susan Diane Wojcicki
Headquarters	San Bruno, California, USA
Acquisition Date	November 13, 2006
Parent Company	Google

2.2 Related Works

Out of all the research papers and projects in this area, our project strives to conduct sentiment analysis of videos and documentaries uploaded to the online video-sharing website, **YouTube**, which means the project helps the user to find the sentiment polarity (*positive, negative, or neutral*) of a text reviews data and evaluate the sentiment score of the text review. Generally a text review is divided into single sentences (*sentence-based*) and words (*word-based*) or very short texts from a single source (the source here being **YouTube**).

A. Sentiment Analysis: An Overview

The methodologies which have been reviewed presented a tool which judges the quality of text based on annotations on scientific papers. The tested methodologies declares in collective's sentiment of annotations in two approaches. The methodologies counts all the annotation produces the documents and calculates total sentiment scores. The problem of this methodology appears in a relationship between annotations that is complex. The technique needs to have a big query knowledge base containing metadata. The notion declares in that the values are not accurate enough such as the value of *Good=0.875* has greater value than the value of *Best=0.75* although the result is wrong in logical meaning. Nevertheless, believing that collecting metadata and evaluating them could be useful to achieve higher analysis quality.

B. Sentiment Analysis Techniques

This section provides a brief description of the two sentiment analysis techniques investigated in this project. These techniques are the most popular in the literature and they cover diverse techniques such as the use of Natural Language Processing (**NLP**) in assigning polarity and sentiment score.

1. Natural Language Toolkit: This method aims at an evaluation sentiment scores and polarity. They produce the *Natural Language Toolkit (**NLTK**)*. NLTK is a text analysis technique that evaluates cognitive and constitutional components of a given text reviews based on using lexicon including words. They use hierarchal sentiment classification level with two levels (*Neutral, Positive, and Negative*). The drawback of this technique illustrated in low accuracy and some logical errors.

2. NLP Stanford sentiment (NLPS): The researchers introduce recursive neural models have in common: word vector representations and classification. The authors released a tool named *NLP Stanford* NLPS, which develops an integration of learning techniques that produces better results and higher accuracy training model empirically. Their goal is based on Semantic word spaces have been very beneficial but NLPS cannot express the meaning of longer phrases in a primary way.

Finding sentiments for aspects can be divided into four sub-tasks, which have been described below:

1. Text extraction: extract sentences and phrases from the reviews

2. Sentiment classification: run a sentiment classifier on each extracted sentence and phrase to determine if it is positive, negative, or neutral

3. Aspect extractor: perform aspect term extraction for the ones that express a sentiment and determine the polarity for each of the aspects identified (**e.g.** food, service)

4. Aspect polarity aggregation: group the sentiments for aspects together and produce a final summary (**e.g.** food: positive, service: negative)

(3) Challenges in Sentiment Analysis

However, there are still some challenges to overcome before sentiment analysis can become a more perfect tool. For example, human judgment is still far more accurate as a gauge in sentiment analysis. Automated systems cannot differentiate sarcasm from sincere text, nor can they always correctly analyse the specific contextual meaning of a word. Use of acronyms like *lol* or word abbreviations also pose interpretation challenges. Furthermore, mixed opinions such as *I like the printer quality, but the size is too big*, can be difficult to classify, as is identifying genuine feedback in a general comment like, *I surprised my wife on her birthday with this new Samsung phone, and she just freaked out!* It's also unlikely that an automated system could identify biased or fake reviews on a product or service.

(4) Code Snippet

4.1 Sentiment-Analysis Python Code

```
# -*- coding: utf-8 -*-
import os
import google.oauth2.credentials


import google_auth_oauthlib.flow
from googleapiclient.discovery import build
from googleapiclient.errors import HttpError
from google_auth_oauthlib.flow import InstalledAppFlow


# The CLIENT_SECRETS_FILE variable specifies the name of a file that contains
# the OAuth 2.0 information for this application, including its client_id and
# client_secret.
CLIENT_SECRETS_FILE = "client_secret.json"
```

```
# This OAuth 2.0 access scope allows for full read/write access to the
# authenticated user's account and requires requests to use an SSL connection.
SCOPES = ['https://www.googleapis.com/auth/youtube.force-ssl']
API_SERVICE_NAME = 'youtube'
API_VERSION = 'v3'


def get_authenticated_service():
    flow = InstalledAppFlow.from_client_secrets_file(CLIENT_SECRETS_FILE, SCOPES)
    credentials = flow.run_console()
    return build(API_SERVICE_NAME, API_VERSION, credentials=credentials)
def print_response(response):
    response = str(response)
    response = response.replace("\"", "'");
    response = response.replace("True", "\"True\"");
```

```
    response = response.replace("''", "\" ");
    response = response.replace("{'", "{\"");
    response = response.replace(" '", " \"");
    response = response.replace("':", "\":");
    response = response.replace(":'", ":\"");
    response = response.replace("',", "\",");
    response = response.replace("'}", "\"}");
    print(response)
    try :
        text_file  =  open("commentsList.json", 
"w")
        text_file.write(response)
        text_file.close()
    except FileNotFoundError as error:
        print("There was an error writing to the 
file" + error)
    else:
        print("The   list   of   comments   was 
successffully    written    to    the    file 
'commentsList.txt'")
```

```
# Build a resource based on a list of properties given as key-value pairs.
# Leave properties with empty values out of the inserted resource.
def build_resource(properties):
    resource = {}
    for p in properties:
        # Given a key like "snippet.title", split into "snippet" and "title", where
        # "snippet" will be an object and "title" will be a property in that object.
        prop_array = p.split('.')
        ref = resource
        for pa in range(0, len(prop_array)):
            is_array = False
            key = prop_array[pa]

            # For properties that have array values, convert a name like
            # "snippet.tags[]" to snippet.tags, and set a flag to handle
```

```
            # the value as an array.
            if key[-2:] == '[]':
                key = key[0:len(key) - 2:]
                is_array = True

            if pa == (len(prop_array) - 1):
                # Leave properties without
values out of inserted resource.
                if properties[p]:
                    if is_array:
              ref[key]=properties[p].split(',')
                    else:
                    ref[key] = properties[p]
            elif key not in ref:
                # For example, the property is
"snippet.title", but the resource does
                # not yet have a "snippet"
object. Create the snippet object here.
                # Setting "ref = ref[key]" means
that in the next time through the
```

```
                # "for pa in range ..." loop, we will be setting a property in the
                # resource's "snippet" object.
                ref[key] = {}
                ref = ref[key]
            else:
                # For example, the property is "snippet.description", and the resource
                # already has a "snippet" object.
                ref = ref[key]
    return resource

# Remove keyword arguments that are not set
def remove_empty_kwargs(**kwargs):
    good_kwargs = {}
    if kwargs is not None:
        for key, value in kwargs.iteritems():
            if value:
                good_kwargs[key] = value
    return good_kwargs
```

```
def    comment_threads_list_by_video_id(client,
**kwargs):
    # See full sample for function
    # kwargs = remove_empty_kwargs(**kwargs)
    response                                    =
client.commentThreads().list(**kwargs).execute(
)
    return print_response(response)
if __name__ == '__main__':
    # When running locally, disable OAuthlib's
HTTPs verification. When
    # running in production *do not* leave this
option enabled.
    os.environ['OAUTHLIB_INSECURE_TRANSPORT'] =
'1'
    client = get_authenticated_service()


    comment_threads_list_by_video_id(client,
part='snippet,replies', videoId='XS6ysDFTbLU')
```

4.2 YouTube-API-CMD Python Code

- *Successful Data Extraction and Analysis*

```
# !/usr/bin/python
# Usage:
# python scraper.py --videoid='<video_id>'


from apiclient.errors import HttpError
from oauth2client.tools import argparser
from apiclient.discovery import build
YOUTUBE_API_SERVICE_NAME = "youtube analytics"
YOUTUBE_API_VERSION = "v3"
DEVELOPER_KEY                                    =
'AIzaSyB3OjFaaHL7yjV8B_SySi9wEYjR_513icQ'
def    get_comment_threads(youtube,    video_id,
comments):
    threads = []
    results = youtube.commentThreads().list(
        part="snippet",
        videoId=video_id,
        textFormat="plainText",
```

```
    ).execute()
    # Get the first set of comments
    for item in results["items"]:
        threads.append(item)
        comment                                                =
item["snippet"]["topLevelComment"]
        text                                                   =
comment["snippet"]["textDisplay"]
        comments.append(text)

        #   Keep   getting   comments   from   the
following pages
    while ("nextPageToken" in results):
        results                                                =
youtube.commentThreads().list(
            part="snippet",
            videoId=video_id,
            pageToken=results["nextPageToken"],
            textFormat="plainText",
        ).execute()
        for item in results["items"]:
```

```
            threads.append(item)
            comment                                   =
item["snippet"]["topLevelComment"]
            text                                      =
comment["snippet"]["textDisplay"]
            comments.append(text)
    print("Total threads: %d" % len(threads))
    return threads
def get_comments(youtube, parent_id, comments):
    results = youtube.comments().list(
        part="snippet",
        parentId=parent_id,
        textFormat="plainText"
    ).execute()

    for item in results["items"]:
        text = item["snippet"]["textDisplay"]
        comments.append(text)

    return results["items"]
```

```
if __name__ == "__main__":
    argparser.add_argument("--videoid",
help="Required; ID for video for which the
comment will be inserted.")
    args = argparser.parse_args()
    youtube = build(YOUTUBE_API_SERVICE_NAME,
YOUTUBE_API_VERSION,
developerKey=DEVELOPER_KEY)


    try:
        output_file = open("output.txt", "w")
        comments = []
        video_comment_threads =
get_comment_threads(youtube, args.videoid,
comments)
        for thread in video_comment_threads:
    get_comments(youtube,thread["id"],comments)
        for comment in comments:
output_file.write(comment.encode("utf-8")+"\n")
        output_file.close()
```

```
        print("Total comments: %d" % len(comments))
    except HttpError as e:
        print("An HTTP error %d occurred:\n%s" % (e.resp.status, e.content))
```

4.3 Kaggle Word-to-Vector Python Code

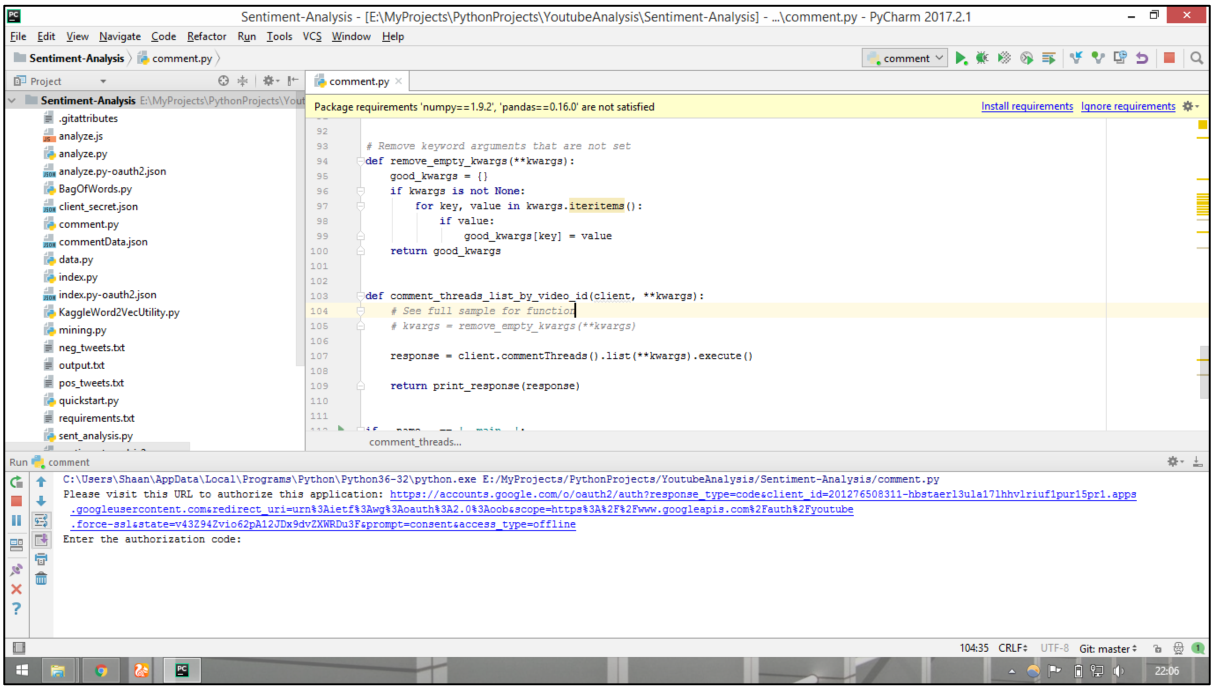
```
#!/usr/bin/env python
import re
import nltk
import pandas as pd
import numpy as np


from bs4 import BeautifulSoup
from nltk.corpus import stopwords
class KaggleWord2VecUtility(object):
    """KaggleWord2VecUtility is a utility class for processing raw HTML text into segments for further learning"""
    @staticmethod
```

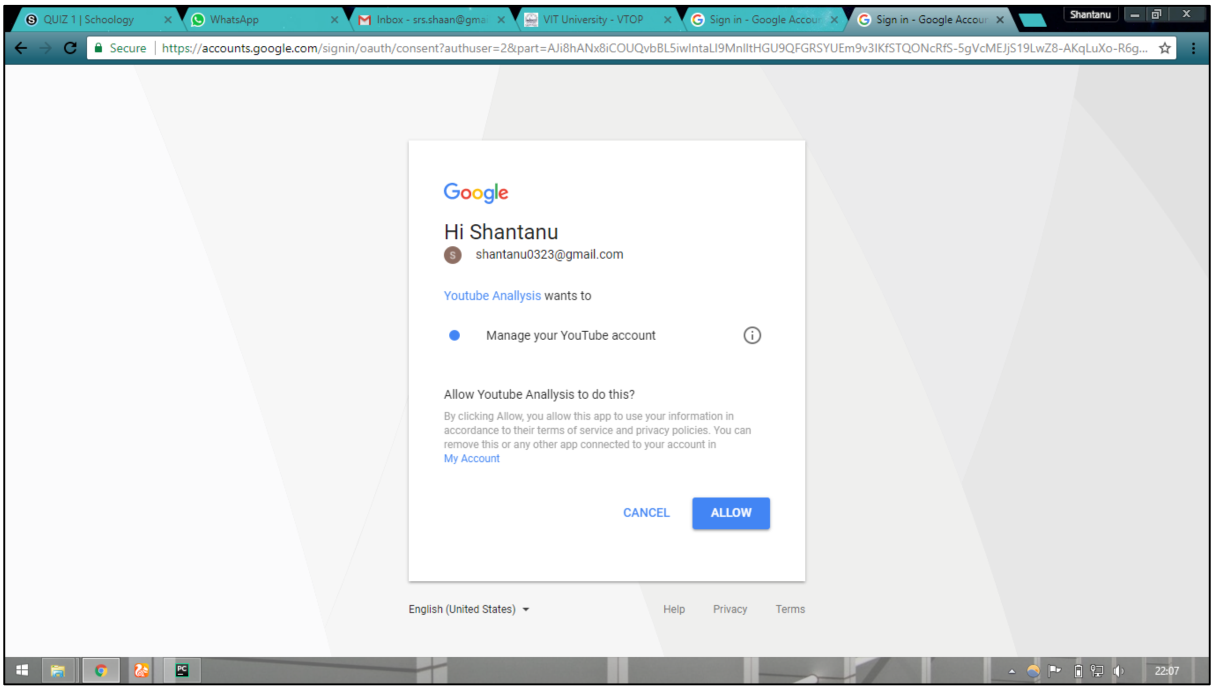
```
def review_to_wordlist( review, remove_stopwords=False ):
    # Function to convert a document to a sequence of words,
    # optionally removing stop words. Returns a list of words.
    #
    # 1. Remove HTML
    review_text = BeautifulSoup(review).get_text()
    #
    # 2. Remove non-letters
    review_text = re.sub("[^a-zA-Z]"," ", review_text)
    #
    # 3. Convert words to lower case and split them
    words = review_text.lower().split()
    #
    # 4. Optionally remove stop words (false by default)
```

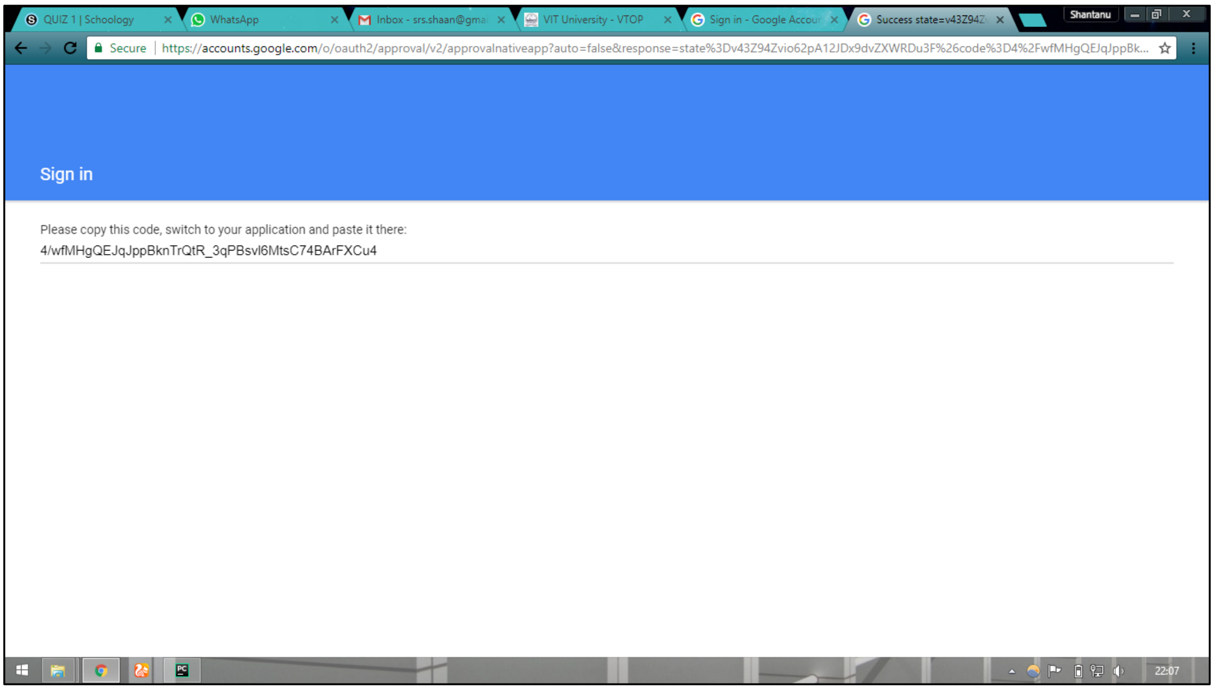
```
        if remove_stopwords:
            stops = set(stopwords.words("english"))
            words = [w for w in words if not w in stops]
        #
        # 5. Return a list of words
        return(words)
    # Define a function to split a review into parsed sentences
    @staticmethod
    def review_to_sentences( review, tokenizer, remove_stopwords=False ):
        # Function to split a review into parsed sentences. Returns a
        # list of sentences, where each sentence is a list of words
        #
        # 1. Use the NLTK tokenizer to split the paragraph into sentences
```

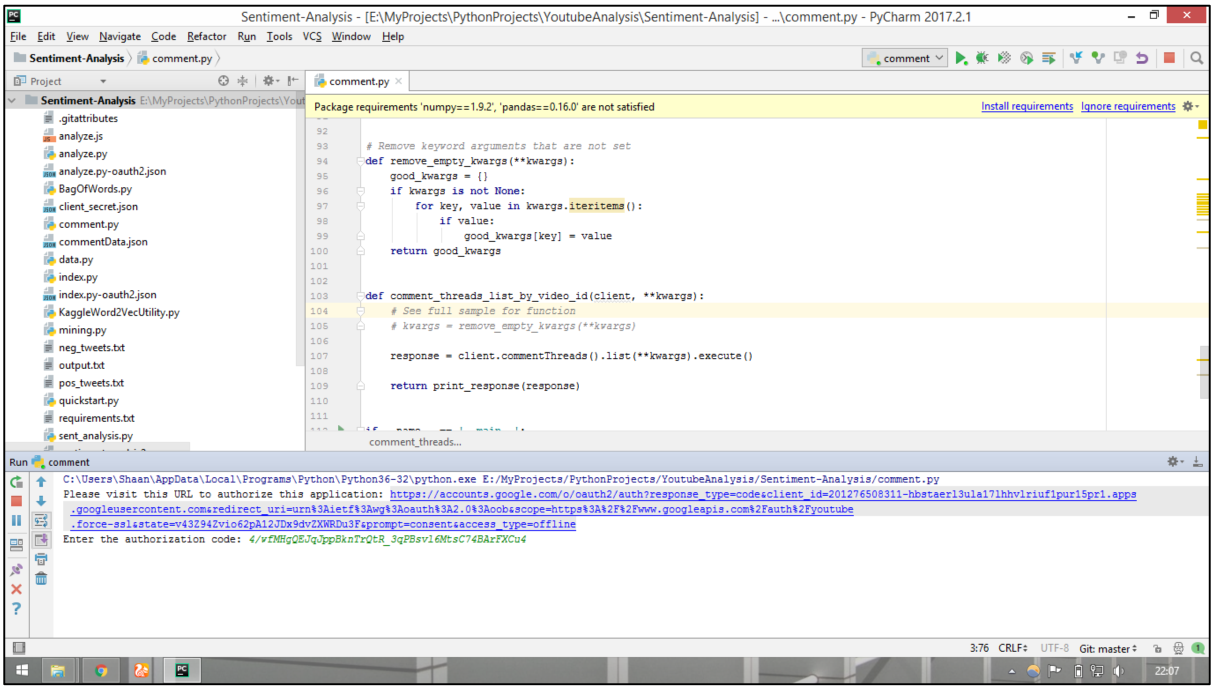
```
        raw_sentences                          =
tokenizer.tokenize(review.decode('utf8').strip(
))
        #
        # 2. Loop over each sentence
        sentences = []
        for raw_sentence in raw_sentences:
            # If a sentence is empty, skip it
            if len(raw_sentence) > 0:
                #        Otherwise,        call
review_to_wordlist to get a list of words
                sentences.append(
KaggleWord2VecUtility.review_to_wordlist(
raw_sentence, \
                  remove_stopwords ))
        #
        # Return the list of sentences (each
sentence is a list of words,
        # so this returns a list of lists
        return sentences
```

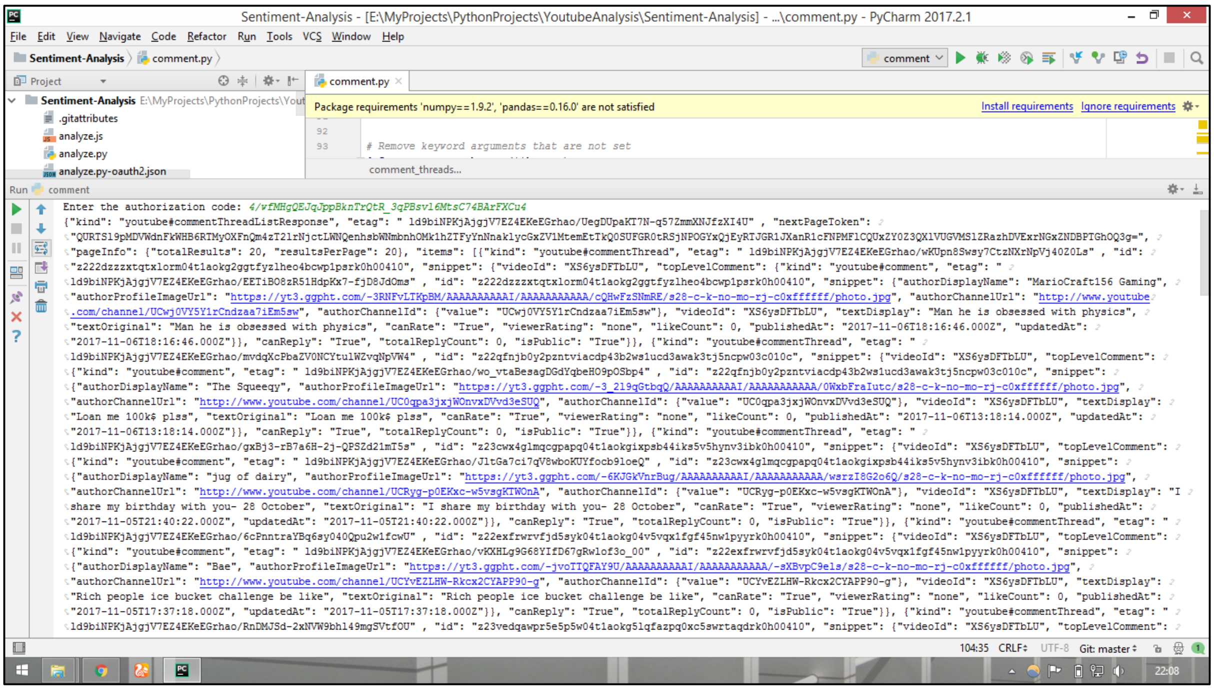
(5) Screenshots

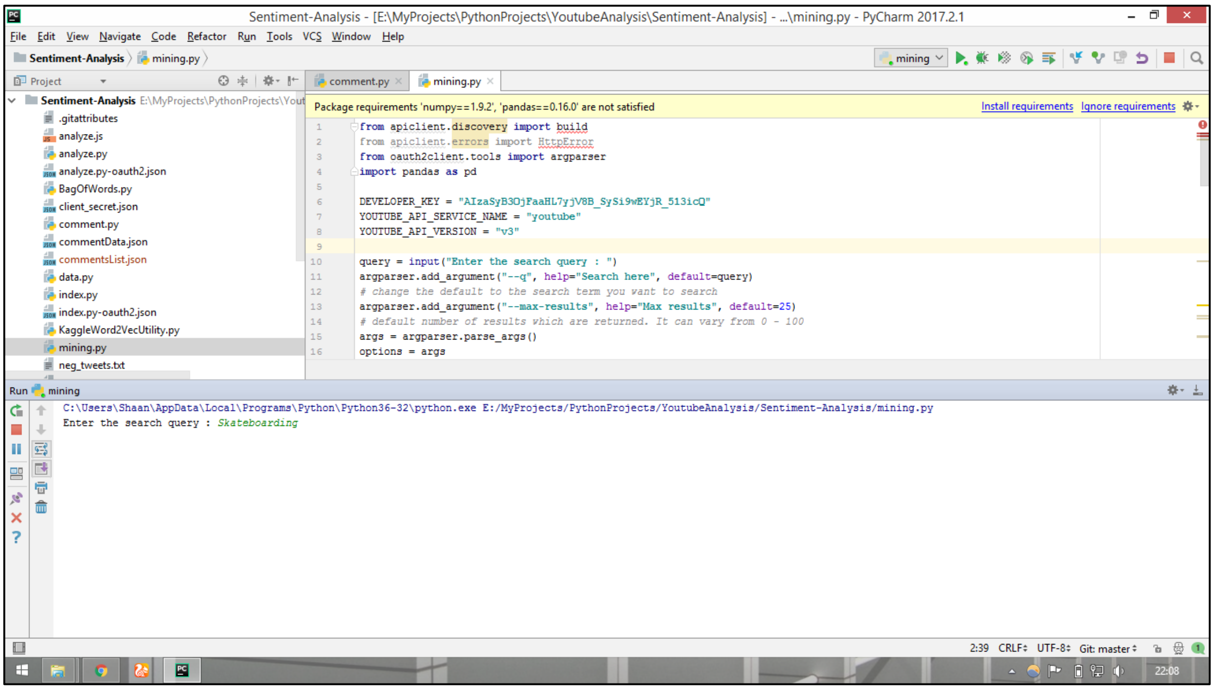


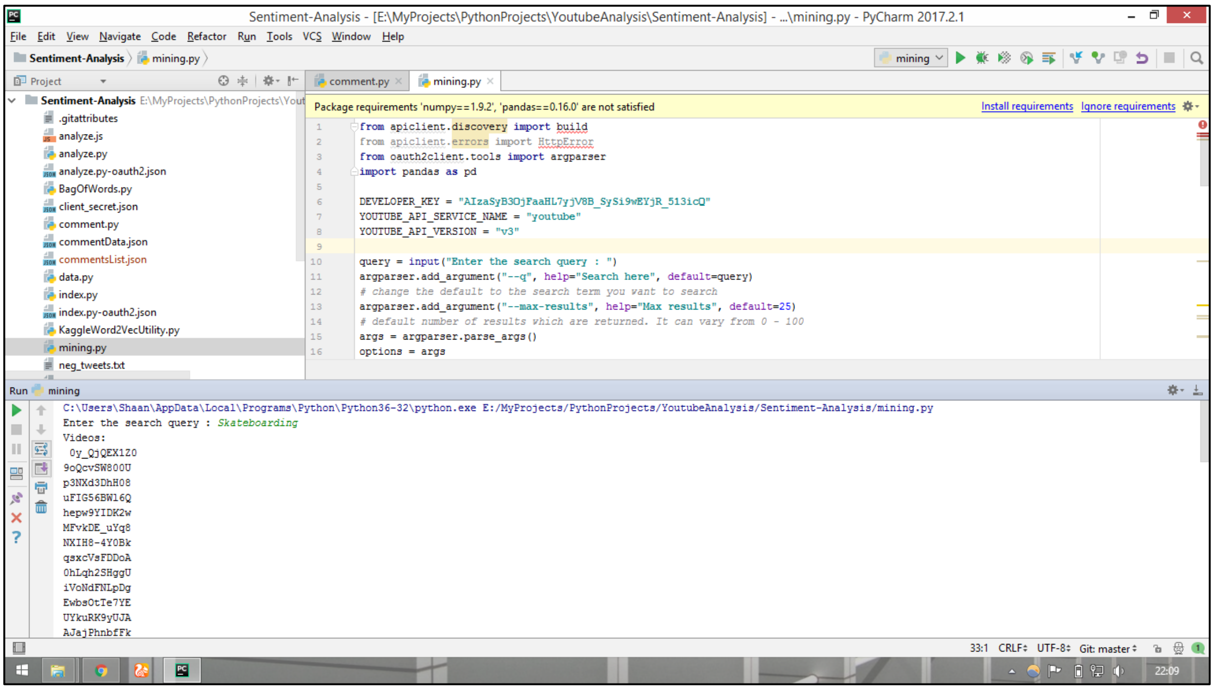


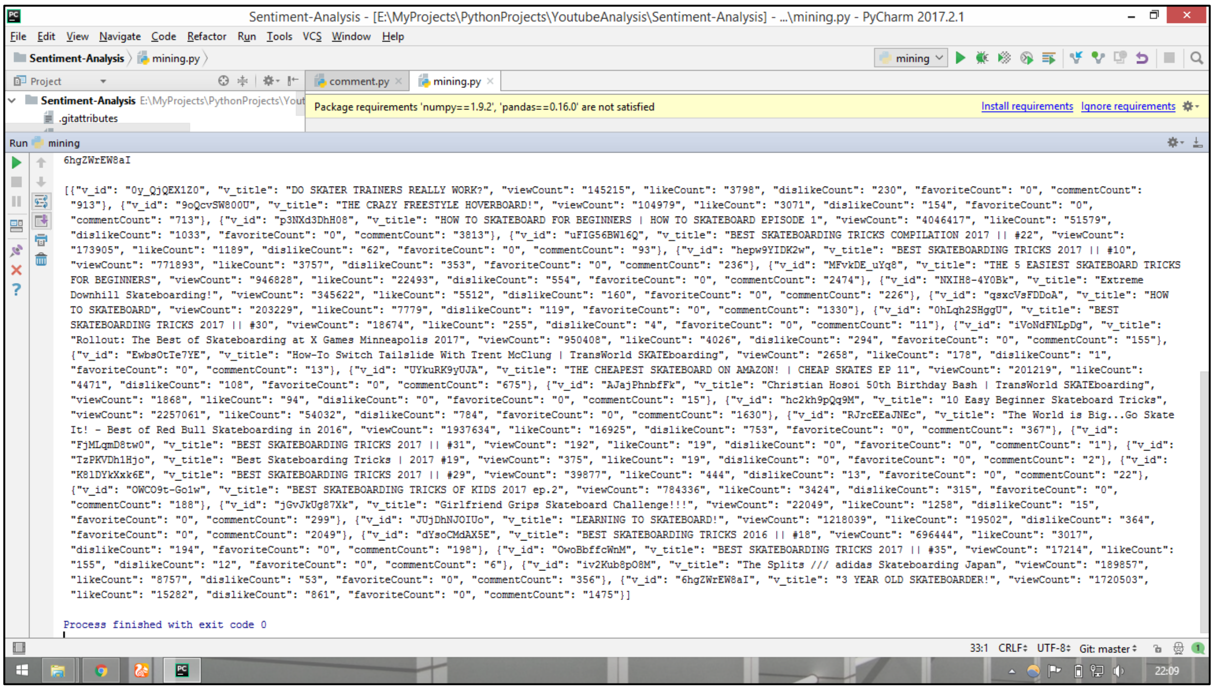

Sign in
Please copy this code, switch to your application and paste it there:
4/wfMHgQEJqJppBknTrQtR_3qPBsvl6MtsC74BArFXCu4


Sentiment-Analysis - [E:\MyProjects\PythonProjects\YoutubeAnalysis\Sentiment-Analysis] - ...\comment.py - PyCharm 2017.2.1
Package requirements 'numpy==1.9.2', 'pandas==0.16.0' are not satisfied
# Remove keyword arguments that are not set
def remove_empty_kwargs(**kwargs):
good_kwargs = {}
if kwargs is not None:
for key, value in kwargs.iteritems():
if value:
good_kwargs[key] = value
return good_kwargs
def comment_threads_list_by_video_id(client, **kwargs):
# See full sample for function
# kwargs = remove_empty_kwargs(**kwargs)
response = client.commentThreads().list(**kwargs).execute()
return print_response(response)
C:\Users\Shaan\AppData\Local\Programs\Python\Python36-32\python.exe E:/MyProjects/PythonProjects/YoutubeAnalysis/Sentiment-Analysis/comment.py
Please visit this URL to authorize this application:
Enter the authorization code: 4/wfMHgQEJqJppBknTrQtR_3qPBsvl6MtsC74BArFXCu4


Sentiment-Analysis - [E:\MyProjects\PythonProjects\YoutubeAnalysis\Sentiment-Analysis] - ...\comment.py - PyCharm 2017.2.1
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Sentiment-Analysis
comment.py
Project
Sentiment-Analysis E:\MyProjects\PythonProjects\Yout
.gitattributes
analyze.js
analyze.py
analyze.py-oauth2.json
Package requirements 'numpy==1.9.2', 'pandas==0.16.0' are not satisfied
Install requirements
Ignore requirements
Remove keyword arguments that are not set
comment_threads...
Run comment
Enter the authorization code: 4/vfMHgQEJqJppBknTrQtR_3qPBsvl6MtsC74BArFXCu4
104:35 CRLF: UTF-8 Git: master
22:08


Sentiment-Analysis - [E:\MyProjects\PythonProjects\YoutubeAnalysis\Sentiment-Analysis] - ...\mining.py - PyCharm 2017.2.1
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Sentiment-Analysis
mining.py
Project
comment.py
mining.py
Sentiment-Analysis E:\MyProjects\PythonProjects\Yout
.gitattributes
analyze.js
analyze.py
analyze.py-oauth2.json
BagOfWords.py
client_secret.json
comment.py
commentData.json
commentsList.json
data.py
index.py
index.py-oauth2.json
KaggleWord2VecUtility.py
mining.py
neg_tweets.txt
Package requirements 'numpy==1.9.2', 'pandas==0.16.0' are not satisfied
Install requirements
Ignore requirements
from apiclient.discovery import build
from apiclient.errors import HttpError
from oauth2client.tools import argparser
import pandas as pd
DEVELOPER_KEY = "AIzaSyB3OjFaaHL7yjV8B_SySi9wEYjR_513icQ"
YOUTUBE_API_SERVICE_NAME = "youtube"
YOUTUBE_API_VERSION = "v3"
query = input("Enter the search query : ")
argparser.add_argument("--q", help="Search here", default=query)
# change the default to the search term you want to search
argparser.add_argument("--max-results", help="Max results", default=25)
# default number of results which are returned. It can vary from 0 - 100
args = argparser.parse_args()
options = args
Run mining
C:\Users\Shaan\AppData\Local\Programs\Python\Python36-32\python.exe E:/MyProjects/PythonProjects/YoutubeAnalysis/Sentiment-Analysis/mining.py
Enter the search query : Skateboarding
2:39 CRLF: UTF-8: Git: master
22:08


Sentiment-Analysis - [E:\MyProjects\PythonProjects\YoutubeAnalysis\Sentiment-Analysis] - ...\mining.py - PyCharm 2017.2.1
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Sentiment-Analysis
mining.py
comment.py
Package requirements 'numpy==1.9.2', 'pandas==0.16.0' are not satisfied
Install requirements
Ignore requirements
.gitattributes
analyze.js
analyze.py
analyze.py-oauth2.json
BagOfWords.py
client_secret.json
comment.py
commentData.json
commentsList.json
data.py
index.py
index.py-oauth2.json
KaggleWord2VecUtility.py
mining.py
neg_tweets.txt
from apiclient.discovery import build
from apiclient.errors import HttpError
from oauth2client.tools import argparser
import pandas as pd
DEVELOPER_KEY = "AIzaSyB3OjFaaHL7yjV8B_SySi9wEYjR_513icQ"
YOUTUBE_API_SERVICE_NAME = "youtube"
YOUTUBE_API_VERSION = "v3"
query = input("Enter the search query : ")
argparser.add_argument("--q", help="Search here", default=query)
# change the default to the search term you want to search
argparser.add_argument("--max-results", help="Max results", default=25)
# default number of results which are returned. It can vary from 0 - 100
args = argparser.parse_args()
options = args
Run mining
C:\Users\Shaan\AppData\Local\Programs\Python\Python36-32\python.exe E:/MyProjects/PythonProjects/YoutubeAnalysis/Sentiment-Analysis/mining.py
Enter the search query : Skateboarding
Videos:
0y_QjQEX1Z0
9oQcvSW800U
p3NXd3DhH08
uFIG56BW16Q
hepw9YIDK2w
MFvkDE_uYq8
NXIH8-4Y0Bk
qsxcVsFDDoA
0hLqh2SHggU
iVoNdFNLpDg
EwbsOtTe7YE
UYkuRK9yUJA
AJajPhnbfFk
33:1 CRLF UTF-8 Git: master
22:09


Sentiment-Analysis - [E:\MyProjects\PythonProjects\YoutubeAnalysis\Sentiment-Analysis] - ...\mining.py - PyCharm 2017.2.1
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Package requirements 'numpy==1.9.2', 'pandas==0.16.0' are not satisfied
Install requirements
Ignore requirements
Run mining
6hgZWrEW8aI
[{"v_id": "0y_QjQEX1Z0", "v_title": "DO SKATER TRAINERS REALLY WORK?", "viewCount": "145215", "likeCount": "3798", "dislikeCount": "230", "favoriteCount": "0", "commentCount": "913"}, {"v_id": "9oQcvSW800U", "v_title": "THE CRAZY FREESTYLE HOVERBOARD!", "viewCount": "104979", "likeCount": "3071", "dislikeCount": "154", "favoriteCount": "0", "commentCount": "713"}, ...
Process finished with exit code 0
33:1 CRLF UTF-8 Git: master
22:09

(6) References

[1] Riloff, Ellen and Janyce Wiebe. 2003. Learning extraction patterns for subjective expressions. In EMNLP.

[2] Subasic, Pero and Alison Huettner. 2001. Affect analysis of text using fuzzy semantic typing. IEEE Trans. Fuzzy Systems, 9(4):483–496.

[3] Cormen, Thomas H., Charles E. Leiserson, and Ronald L. Rivest. 1990. Introduction to Algorithms. MIT Press.

[4] Das, Sanjiv and Mike Chen. 2001. Yahoo! for Amazon: Extracting market sentiment from stock message boards. In Asia Pacific Finance Association Annual Conf. (APFA).

[5] Durbin, Stephen D., J. Neal Richter, and Doug Warner. 2003. A system for affective rating of texts. In KDD Wksp. on Operational Text Classification Systems (OTC-3).

[6] Joachims, Thorsten. 2003. Transductive learning via spectral graph partitioning. In Intl. Conf. on Machine Learning (ICML).

[7] Montes-y-G´omez, Manuel, Aurelio L´opez-L´opez, and Alexander Gelbukh. 1999. Text mining as a social thermometer. In IJCAI Wksp. on Text Mining, pages 103–107

[8] Morinaga, Satoshi, Kenji Yamanishi, Kenji Tateishi, and Toshikazu Fukushima. 2002. Mining product reputations on the web. In KDD, pages 341– 349. Industry track.

[9] Turney, Peter D. and Micharel L. Littman. Measuring praise and criticism: Inference of semantic orientation from association. ACM Transactions on Information Systems, 2003.

[10] Utsumi, Akira. Verbal irony as implicit display of ironic environment: Distinguishing ironic utterances from nonirony. Journal of Pragmatics, 2000. 32(12): p. 1777-1806.