**README**

**Option2: API mash up**

**File name: 206\_data\_access.py**

This program helps to identify movie lovers on Twitter by searching the director of certain movies and calculating the percentage of “true” lovers. The “true movie lover” is defined as users who have “movie”, “film” and “cinema” in his or her description.

Since the program will give the percentage of users who have some knowledge about film, it can be used to study user behavior regarding films. For example, what is the percentage of users who post film-related tweets for the trend? What is the percentage of users who contribute constructive information regarding a film?

For this program, the input has to be a list of films (not restricted in the number of films). The output is the number of films the user searches for, the number of users who mention the director of each film, number of “true movie lovers” and the percentage of “true movie lovers” over users.

Three databases are created in this program, namely Movies, Tweets and Users.

**Instructions:**

To run this program, the user has to input a list of movies (line 168). After running it, a csv file named movie\_lover\_rate will be generated in the same directory and it will contain the results.

To run this program, the user does not have to install anything. Files needed are as following:

1. twitter\_info.py

Files included:

1. SI206\_cache.json

**Functions and classes**

The program also includes the following functions:

def getWithCaching(url, params):

input: url of the website, parameters needed

return: response text depending on the user’s search parameters

def canonical\_order(d):

input: a dictionary

return: a sorted dictionary by alphabetical orders

def requestURL(baseurl, params={}):

input: base url and the parameters:

return: the full url needed

def get\_movie\_info(movie):

input: name of a movie

return: a dictionary of the movie information from OMdb

def sort\_lst():

No input required

Return: a sorted list of all the movie information according to the rating, from high to low

def get\_director\_tweets(input):

input: name of a director

return: a dictionary of the tweet related to the director from Twitter

def get\_user(input):

input: name of the user

return: a dictionary of the user information from Twitter

The program includes the following classes:

class Movie(object):

One instance of the class represents a movie object. The constructor needs an input of a dictionary of the movie information. The Movie class has the following methods:

\_\_str\_\_(): it will format the movie title to recommended + movie title if the rating of the movie if above 7.5.

load\_movie\_data(): It will load all the movie data into the database set up in the beginning and also return a list of director names of the movies.

class Tweet(object):

One instance of the class represents a tweet about a certain director. The constructor needs an input of a dictionary of the tweet information. The Tweet class has the following methods:

load\_tweet\_data(): It will load all the tweet data into the database set up in the beginning. No return value for this method.

class User:

One instance of the class represents a user who is related to the tweet. The constructor needs an input of a dictionary of the user information. The User class has the following methods:

load\_user\_data(): It will load all the user data into the database set up in the beginning. No return value for this method.

**Database creation**

The database has three tables, namely Movies, Tweets and Users

Movies:

Imdb\_id: the id of the film

Title: the title of the film

Director: the director of the film

Num\_lang: number of languages in the film

Imdb\_rating: the imdb\_rating of the film

Tweets:

Tweet\_id: the id of the tweet

Retweet\_count: the number of retweets of the tweet

Text: the content of the tweet

Screen\_name: the screen\_name of the user who posts the tweet

User\_id: the id of the user

Favorite\_count: the number of favorites of the tweet

Users:

User\_id: the id of the user

Favorite\_count: the number of favorites of the user

Description: the self-description of the user

Screen\_name: screen name of the user

Follower\_count: the number of followers of the user

**Data manipulation code**

The program will show the percentage of “true movie lovers” for any movie, regardless of the number of movies. No matter how many movies input, a number will be generated. This can be very useful in finding the trend/taste of movie lovers. For example: we can identify the percentage of movie lovers who give comments on different genres of movies (blockbuster movies, cult movies, classics) or movie of different ratings (especially recent movies). Our prediction is that the higher the rating, the higher the percentage.

The program will also return all the movie information in a sorted manner (from high to low according to imdb\_rating)

**Purpose of the project**

I want to study the popularity/quality of a recent movie not only by its rating but by the percentage of movie lovers who comment on the movie/director. In fact, if there is a relationship between the rating and percentage, we can establish a new rating system for movies based on its social media popularity.

* + Line(s) on which each of your data gathering functions begin(s): line 123, 192, 253
  + Line(s) on which your class definition(s) begin(s): line 137, 216, 284
  + Line(s) where your database is created in the program: line 62-86
  + Line(s) of code that load data into your database: line 197, 251, 301
  + Line(s) of code (approx) where your data processing code occurs — where in the file can we see all the processing techniques you used?: line 320-326, 174-181
  + Line(s) of code that generate the output. Line 329-334