Taylor Lundeen
April 24th, 2017
SI 206

Final Project README: Option 2

The Program's Overall Purpose:

- This code exists to make it easy for users to access data pertaining to certain movies from the OMDB API as well as data pertaining to tweets about those movies and the user's who tweeted/ were mentioned in tweets about them.
- This program will output a text file that gives a brief summary of the details of each movie, a list of the most common characters found in the tweets about the movie, and the locations from which Twitter users tweeted about the movie.
- This program will create a database- finalproject.db- which has three tables within it. One table for data pertaining to Movies, one table for data pertaining to Tweets about the movie, and one table pertaining to Users who tweeted about the movie or who were mentioned in tweets about the movie.

The Program's Dependencies:

- This code requires a python file called twitter_info.py that holds data required for proper authorization to access Twitter data using the Tweepy API wrapper.
  - Within this file you must have variables holding strings that represent the consumer_key, consumer_secret, access_token, and access_token_secret.
- This code requires that you have installed the following modules using pip:
  - requests
  - unittest
  - tweepy
  - json
  - sqlite3
  - itertools
  - collections

Files Included in the Program:

- The sqlite3 database: "finalproject.db"
- The json cache file: "SI206_finalproject_cache.json"
- The Summary Statistics .txt file: "Lundeen_FinalProject_SummaryStats.txt"

How to run the code:

- In order to run the code, you must have the twitter_info.py file in the same directory as the program's python file.
- From there, simply navigate into that directory using your command line and type in python Lundeen_206_finalproject.py into your command line and hit enter.

Summary of the Program's Functions:

- get_tweets:
    - **Input**: a string representing the term you wish to search for and access data pertaining to tweets about. (*required)*
    - **Return value**: a dictionary from the cache holding the response dictionary from your request to Tweepy API pertaining to your search term. This dictionary holds the data about the Tweets you searched for.
    - **Behavior:** This function also checks to see if data pertaining to the inputted search term is already in the cache and if it is it returns the given dictionary. If it is not already in the cache, this function will make a request to the Tweepy API and will cache the data returned from the request.
- get_user_info:
    - **Input:** a string representing a given Twitter user's user id (*required*), and a string representing a Twitter user's screename (*required*)
    - **Return Value:** a dictionary from the cache holding the response dictionary from your request to the Tweepy API pertaining to the user you searched for.
    - **Behavior:** This function also checks to see if data pertaining to the inputted screenname is already in the cache and if it is it returns the given dictionary. If it is not already in the cache, this function will make a request to the Tweepy API and will cache the data returned from the request.
- get_movie_data:
    - **Input:** a string representing a movie title. (*required*)
    - **Return Value:** a dictionary value from the cache holding the response dictionary from your request to the OMDB API pertaining to the movie title you searched for.
    - **Behavior:** This function also checks to see if data pertaining to the inputted movie title is already in the cache and if it is it returns the given dictionary. If it is not already in the cache, this function will make a request to the OMDB API and will cache the data returned from the request.

Summary of the Program's Classes:

- Movie class:
    - **Instance Represents:** One instance of this class will represent information about a movie as obtained by the OMDB API.
    - **Required constructor input:** a response dictionary from the OMDB API. This is the return value of the get_movie_data function mentioned above.
    - **Other Methods:**

        - __str__: The __str__ method will access the self.title, self.director, and self.actors instance variables and input them into a string that will be printed when an instance of the class is created. The string will follow this format: "The movie {self.title}, was directed by {self.director} and stars {self.actors}." Requires no additional input.
        - get_top_actor: This method returns the top billed actor from the list of actors in the movie. Requires no additional input.

Database Creation:

- Movies Table:
    - Columns:
        - ID- Primary Key
        - title (string)
        - director (string)
        - num_languages (integer)
        - IMDB_rating (string)
        - top_actor (string)
        - plot (string)
- Tweets Table:
    - Columns:
        - text- string holding the Tweet text
        - tweet_id- Primary Key
        - user_id- reference to Users table
        - movie title- reference to Movies table (string)
        - num_favorites (integer)
        - num_retweets (integer)
- Users Table:
    - Columns:
        - user_id- Primary Key
        - screenname (string)
        - num_faves (integer)
        - num_followers (integer)
        - location (string)

Data Manipulation Code:

- The data manipulation in this code was used to make connections in the data. The most common words in tweets about each movie were found using the collections module's Counter method. List comprehension and dictionary accumulation was used to count the locations in which the tweets were tweeted from.

Specific Lines to Note:

    - Line(s) on which each of your data gathering functions begin(s):

        - get_tweets- Line 53
        - get_user_info- Line 72
        - get_movie_data- Line 95
    - Line(s) on which your class definition(s) begin(s):
        - Movie class definition- Line 121
    - Line(s) where your database is created in the program:
        - Database creation- Line 143- Line 168
    - Line(s) of code that load data into your database:
        - Lines 173- 256

    - Line(s) of code (approx) where your data processing code occurs — where in the file can we see all the processing techniques you used?
        - Data Processing-
            - List comprehension: Lines 202, 266, 288, 312, 330, 352, 373,
            - Counter method: Lines 278, 301, 322
            - Dictionary accumulation: Line 360
    - Line(s) of code that generate the output: Lines 393-433

Why I chose to do this project:

- I chose to do this project because I find making connections with data very interesting. I thought that Twitter would be a great avenue to find trends in posts regarding certain movies. I also was excited to find such trends about movies I enjoy.