Sindhu Giri

Jackie Cohen

SI 206: Data Oriented Programming

April 25, 2017

Final Project—Option 2 Documentation

1. What option did you pick (1, 2, or 3)
Option 2

2. What does it do?
This project caches data from the Twitter and OMBD api and inputs this information as three tables into a database. Then there are queries made into the database; data processing techniques are used to determine whether there is difference between the retweets/ favorite count for movies with an IMDB rating greater than 6 and retweets/favorite for all the movies I searched for. This information along with additional data is provided in summary statistics text file.

3. How do you run it?
You need to run the .py file in the terminal. You also need the twitter_info.py file in the same directory to be able to run the file.

4. What are its dependencies?
- No specific dependencies
- The import statements are listed at the top of my 206_finalproject.py file
- Any twitter_info.py file should work

5. What files are included?
- 206_finalproject.py—Primary python file that contains all of the code
- SI206_finalproject_cache.json—Cache file that contains all of the cached data from the requests to the Twitter and OMDB APIs
- summarystats.txt—Text file with the summary statistics of all of the data processing from the database
- proj_doc.PDF—Documentation for the final project in PDF format
- twitter_info.py—Python file that contains the consumer_key, consumer_secret, access_token, and access_token_secret in order to request to the Twitter API
- project3_tweets.db—Database that is created with the cached data from the Twitter and OMDB APIs

6. **Functions**
1. get_tweets (lines 55-76):
   i) Input: required—phrase used for unique identifier, optional—none
   ii) Return Value: Tweets—a list that contains all of the tweet data from the response text

iii) Behavior: Checks to see if the unique identifier is in the CACHE_DICTION. If it is the response_text is the key of the CACHE_DICTION [unique identifier] value. If it isn't, the tweepy module searches the twitter api for the unique identifier and makes the value of the unquie identifier key the response_text. Then it dumps all of this data into the CACHE_DICTION. Then there is a for loop to find the data under response_text["statuses"] and all of this data is appended to the list tweets.

2. get_user_tweets (lines 106-122):
   i) Input: required—phrase used for unique identifier, optional—none
   ii) Return Value: Response—the values of the unique identifier key in CACHE_DICTION
   iii) Behavior: Checks to see if the unique identifier is in the CACHE_DICTION. If it is the response_text is assigned to the key value of CACHE_DICTION [unique identifier]. If it isn't, the tweepy module searches the twitter api for the unique identifier and makes the value of the unquie identifier key the response_text. Then it dumps all of this data into the CACHE_DICTION.

3. get_OMBDinfo (lines 138-155):
   i) Input: required—phrase used for unique identifier, optional—none
   ii) Return Value: Response—the values of the unique identifier key in CACHE_DICTION
   iii) Behavior: Checks to see if the unique identifier is in the CACHE_DICTION. If it is the response_text is assigned to the key value of CACHE_DICTION [unique identifier]. If it isn't, the tweepy module searches the OMDB api for the unique identifier and makes the value of the unquie identifier key the response_text. Then it dumps all of this data into the CACHE_DICTION.

7. **Classes**

1. Tweet (lines 78-102):
   i. Instance: "user" will hold a string of the screen name of the person who wrote the tweet
   ii. Input: tweet_list, movie_titles
   iii. Methods:
      1. user_mentions—no additional input, finds the users mentioned in a given tweet, appends the user mentions to a list called user_id, returns user_id
      2. tweet_stuff—no additional input, makes a tuple of all the instance variables in the class Tweet, returns the tuple

2. User (lines 124-134):
   i. Instance: "num_favs" will hold a integers of the favorite counts of users on tweets
   ii. Input: user_tweets
   iii. Methods:

1. user_stuff— no additional input, makes a tuple of all the instance variables in the class User, returns the tuple

3. Movie (lines 124-134):
    i. Instance: "director" will hold a string of the director of a movie
    ii. Input: movie_diction
    iii. Methods:
        a. lst_actors— no additional input, splits the list of actors according to a comma, returns the list of actors that contains separate strings for each actors' name
        b. num_languages—no additional input, splits the list of language according to a comma and then finds the length of the list , returns an integer of the number of languages
        c. billed_actor—no additional input, invokes the lst_actors method in order to find the first actor in the list, returns the first actor in the lst_actors method
        d. omdb_stuff—no additional input, makes a tuple of all the instance variables in the class Movie, returns the tuple

8. **Database Creation**
1. Tweets (lines 199-203):
    i. tweet_id—integer, primary key
    ii. text—text
    iii. user—text
    iv. retweets—integer
    v. user_mentions—text
2. Users (lines 206-210):
    i. user—text, primary key
    ii. screen_name—text
    iii. num_favs—integer
3. Movies (lines 213-217):
    i. id—text, primary key
    ii. title—text
    iii. director—text
    iv. imdb_rating—text
    v. billed_actor—text
    vi. num_languages—integer

9. **Data Manipulation Code**
- The code makes inner join queries to find connectness between retweets and favorite counts and movies with an imdb_rating greater than 6. Then, the Collections library is used to find the most common retweets, num_favs for an imdb_rating greater than 6.
- All of this data processing is useful because it develops relationships between seemingly unrelated information. Thus, I found connections between data tables in order to answer my overall prompt.

- It compares to see whether searching with the constraint of the imdb_rating greater than 6 changes the retweet/num_fav amount.
- A user should expect to understand how having a higher rated movie (the imdb_rating greater than 6) mentioned within a tweet influences the amount of time a tweet is retweeted or the amount of times a user favorites the tweet.

10. **Why did you choose to do this project?**

I chose to do this project because I worked with the Twitter API in the SI 106 final project. I wanted to deepen my understand to process additional data from the Twitter API and find meaningful connections as it relates to the movies on OMDB.