

# OMDB & Twitter API Mash-up Documentation (Option 2)

## About

- Pulls movie information from the OMDB API and tweets from the Twitter API to learn more about the actors and what's being said about them
- Takes a list of movie names
- Creates a .txt files with some interesting findings about the movies
- With SQL, creates a database with 4 tables for the movies, tweets, users who posted the tweets, & users who were mentioned in the tweets

## How to Run it

- Enter into the directory that contains the .py file and run in Terminal using python3
- Note: twitter\_info.py should be in the same directory as the .py file for Twitter authentication to work

## Imports

- **unittest** – for testing purposes
- **itertools** – implements iterator tools to help run queries
- **collections** – specialized container data types for running queries
- **requests** – to make a call to the OMDB API url
- **tweepy** – access to entire twitter RESTful API methods
- **twitter\_info** – other file in folder for Twitter authentication
- **json** – to parse through returned information from OMDB & Twitter APIs
- **sqlite3** – to allow us to make our database

## Files Included

### FinalProject.py

- Run this file to make calls to OMDB and Twitter APIs to learn something cool about the movies we've inputted

### Final\_project\_206.db

- The database containing all tweet, user, user mentions, and movie data we retrieved

Check\_Out\_The\_Results.txt

- .txt document to allow the user to see our findings in a clear, formatted manner

SI206\_final\_project\_cache.json

- Holds our cached information for tweets

User\_final\_project\_cache.json

- Holds our cached information for the users

OMDB\_final\_project\_cache.json

- Holds our cached information for the movies

**Note:** Different cache files were created because some inputs would be repeats across the four functions. To avoid information being overwritten, new cache files were required.

twitter\_info.py

- Holds our token information for Twitter authentication

## Functions

OMDB(movie\_name)

- Takes a required movie name string
- Returns the JSON movie information from OMDB RESTAPI

get\_tweets(input\_string)

- Take a required string of the name of the leading actor of our movie
- Returns a dictionary of 15 tweets' info

user\_info(dictionary)

- Takes a required dictionary of twitter information
- Returns a dictionary of information about the users that posted those tweets

usermentioned\_info(dictionary)

- Takes a required dictionary of twitter information
- Returns a dictionary of the information about the users that were mentioned in the tweets
- Used to construct our social network in our database

## Classes

### Movie

- This class represents a movie
- Required inputs to constructor include a dictionary of movie data, the title of the movie, the director, and the IMDB rating

Methods:

- `__init__()`
  - Required inputs to constructor include a dictionary of movie data, the title of the movie, the director, and the IMDB rating
  - Instance variables created for each
- `__str__()`
  - No additional inputs
  - Returns a string formatted to include basic information about the movie in a readable format
- `listactors()`
  - No additional inputs
  - Returns a list of all the actors in the movie
- `numlanguages()`
  - No additional inputs
  - Returns an integer which is the number of languages that are in that movie

## Database Creation

### Tables & what's in each row/column:

Tweets – each row represents information about a tweet

- Text
- ID
- User who posted the Tweet

- Movie search term
- Number favorites
- Number retweets

Users\_Mentioned – each row represents information about a user mentioned in the tweets

- User Mentioned ID
- User Mentioned screen name
- Number of favorites by user mentioned

Movies – each row represents information about a movie we searched

- Movie ID
- Title of movie
- Director of movie
- Number of languages the movie has
- IMDB rating of movie
- Top billed actor of movie

Users – each row represents information about the user who posted a tweet

- User ID
- User screen name
- Number of favorites that the user made

## Database Manipulation

The code also runs queries to:

- Get tweets with more than 50 retweets
- Get all the movies along side the number of retweets each of the 15 tweets got for their leading actor
- Get all the tweets that got likes and how many likes they got
- Get all the users who favorited more than 10,000 times

Useful because it gives us more insight as to which tweets are actually popular, which movie tweets are most popular, which tweets about certain actors are most popular, and the most active users.

Considering a movie to watch? These queries can help.

## Why API Mash-Up?

- It is important to be able to extract information from the web
- This program can allow us to do large scaled research about what people are saying about different actors
- When we analyze who these people are interacting with, we begin to construct a social network
- We can learn more about a movie than just the basic information that is given to us - this new information can help us decide if we want to watch a movie or not

## What's Happening and Where:

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

Line 124

- Line(s) on which your class definition(s) begin(s):

Line 244

- Line(s) where your database is created in the program:

Lines 320 - 350

- Line(s) of code that load data into your database:

Lines 353 - 468

- Line(s) of code (approx) where your data processing code occurs — where in the file can we see all the processing techniques you used?

Lines 474 - 526

- Line(s) of code that generate the output.

Lines 536 - 603

## Running the Program:

```
FinalProject — bash — 119x37
atoteja@Alpanas-MacBook-Pro-3:~/desktop/FinalProject$ python3 FinalProject.py
getting new data from the web for Batman Begins
getting new data from the web for National Treasure
getting new data from the web for Moneyball
getting new data from the web for Christian Bale
getting new data from the web for Nicolas Cage
getting new data from the web for Brad Pitt
-----
These are all the tweets with more than 50 retweets:
['RT @Impeach_D_Trump: Christian Bale: The world is watching Donald Trump read a Dictatorship for Dummies book \n\nRETWEET if You Agree', 'RT @thedailybeast: Christian Bale on President Trump: "It's like we're watching somebody reading a Dictatorship for Dummies book" https://t...', 'RT @hardball: "I was really embarrassed that I wasn't aware" of the Armenian genocide before the film -- Christian Bale #ThePromise #Armeni...', 'RT @PrisonPlanet: Christian Bale calls Trump a dictator.\n\nThis is really going to put his Hollywood career at risk.\n\nHow brave!\n\nHow daring...', 'RT @PrisonPlanet: Christian Bale calls Trump a dictator.\n\nThis is really going to put his Hollywood career at risk.\n\nHow brave!\n\nHow daring...', 'RT @Impeach_D_Trump: Christian Bale: The world is watching Donald Trump read a Dictatorship for Dummies book \n\nRETWEET if You Agree', 'RT @Impeach_D_Trump: Christian Bale: The world is watching Donald Trump read a Dictatorship for Dummies book \n\nRETWEET if You Agree', 'RT @Impeach_D_Trump: Christian Bale: The world is watching Donald Trump read a Dictatorship for Dummies book \n\nRETWEET if You Agree', 'RT @Impeach_D_Trump: Christian Bale: The world is watching Donald Trump read a Dictatorship for Dummies book \n\nRETWEET if You Agree', 'RT @WeLoveRobDyrdek: Your week presented by Nicolas Cage reactions. https://t.co/dX14Not9j', 'RT @80sMoviePosts: Brad Pitt in the 80s https://t.co/bf3dMrjGR', 'RT @80sMoviePosts: Brad Pitt in the 80s https://t.co/bf3dMrjGR', 'RT @80sMoviePosts: brad pitt in the 80s https://t.co/znQ4D4kcZf']
-----
Which leading actor is most popular?
Here are all the movies along side the number of retweets for their leading actor:
{'Batman Begins': [0, 0, 0, 4, 7, 7, 61, 482, 482, 578, 578, 578, 578, 668], 'Moneyball': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 13, 305, 12236, 12236], 'National Treasure': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 2459]}
-----
Here are all the tweets that actually got some likes...and how many likes they got:
[{'#U Eso ya seria más perfecto que Nicolas Cage https://t.co/0ovl0eBHjE', 1}, {'Need a #creativity boost today? Check out our #contentcreation and #blogging tips from #NationalTreasure #NicCage! https://t.co/0fAyKvSQEh', 1}, {'Misty Nopeland is a national treasure. Thank you Nicolas cage.', 2}]
-----
```

```
FinalProject — bash — 119x37
43)>
  for node in nodes:
//anaconda/lib/python3.5/site-packages/cryptography/x509/oid.py:21: ResourceWarning: unclosed <socket.socket fd=14, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM, proto=6, laddr=('35.2.9.152', 61624), raddr=('104.244.42.66', 443)>
43)>
  for node in nodes:
//anaconda/lib/python3.5/site-packages/cryptography/x509/oid.py:21: ResourceWarning: unclosed <socket.socket fd=15, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM, proto=6, laddr=('35.2.9.152', 61625), raddr=('104.244.42.66', 443)>
43)>
  for node in nodes:
//anaconda/lib/python3.5/site-packages/cryptography/x509/oid.py:21: ResourceWarning: unclosed <socket.socket fd=16, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM, proto=6, laddr=('35.2.9.152', 61626), raddr=('104.244.42.66', 443)>
43)>
  for node in nodes:
//anaconda/lib/python3.5/site-packages/tweepy-3.6.0-py3.5.egg/tweepy/binder.py:222: ResourceWarning: unclosed <socket.socket fd=18, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM, proto=6, laddr=('35.2.9.152', 61627), raddr=('104.244.42.66', 443)>
ok
test_usersmentioned_info (__main__.APItests) ... Using cached data for Christian Bale
Using cached data
ok
test_constructor (__main__.MovieClassMethodsTests) ... getting new data from the web for Finding Nemo
ok
test_listactors (__main__.MovieClassMethodsTests) ... Using cached data for Finding Nemo
ok
test_numlanguages (__main__.MovieClassMethodsTests) ... Using cached data for Finding Nemo
ok
test_str (__main__.MovieClassMethodsTests) ... Using cached data for Finding Nemo
ok
test_users1 (__main__.Sqltask) ... ok
test_users2 (__main__.Sqltask) ... ok

-----
Ran 12 tests in 5.396s

OK
atoteja@Alpanas-MacBook-Pro-3:~/desktop/FinalProject$
```

