get_movie_reviews

January 9, 2020

1 Get Reviews

1.0.1 Initial Movie Review Collection.

[1]:
```
# import os
# import pandas as pd
import requests
from api_keys import themoviedb_api_key
# from pprint import pprint
```

1.0.2 The New York Times

Efforts to get reviews from The New York times have been suspended since its API does not serve complete reviews, only review metadata.

[2]:
```
# api_key = nytimes_api_key
```

[3]:
```
# url = "https://api.nytimes.com/svc/movies/v2/reviews/search.json?"
# query = "avengers"
# query_url = url + "api-key=" + api_key + "&q=" + query
```

[4]:
```
# reviews = requests.get(query_url).json()
```

[5]:
```
# reviews
```

[6]:
```
# q = "https://api.nytimes.com/svc/movies/v2/reviews/search.json?
 ↪query=godfather&api-key=4mI9SosndaEigcAzs5zDFk8SPNvm7Wdd"
# r = requests.get(q).json()
```

[2]:
```
# r['results'][0]['link']['url']
```

1.0.3 The Movie Database

[30]:
```
# Initialize movie list and empty movie dictionary.
movie_list = ['The Avengers', 'Avengers: Age of Ultron', 'Captain America:␣
 ↪Civil War', 'Avengers: Infinity War', 'Avengers: Endgame']
movie_dict = {
    'movie_ids' : [],
    'titles' : [],
    'dates' : [],
    'average_score' : [],
    'vote_count' : [],
    'popularity' : [],
}
```

[31]:
```
# Method to search themoviedb and append information from the top hit into␣
↪movie_dict.
def get_movie_info(search_term):
    query_url = f"https://api.themoviedb.org/3/search/movie?
↪api_key={themoviedb_api_key}&query={search_term}"
    response = requests.get(query_url).json()
    movie_dict['movie_ids'].append(response['results'][0]['id'])
    movie_dict['titles'].append(response['results'][0]['title'])
    movie_dict['dates'].append(response['results'][0]['release_date'])
    movie_dict['average_score'].append(response['results'][0]['vote_average'])
    movie_dict['vote_count'].append(response['results'][0]['vote_count'])
    movie_dict['popularity'].append(response['results'][0]['popularity'])
```

[15]:
```
# search_term = "Avengers: Age of Ultron"
# query_url = f"https://api.themoviedb.org/3/search/movie?
↪api_key={themoviedb_api_key}&query={search_term}"
# response = requests.get(query_url).json()
```

[23]:
```
response['results'][0]
```

[23]:
```
{'popularity': 31.096,
 'vote_count': 14794,
 'video': False,
 'poster_path': '/t90Y3G8UGQp0f0DrP60wRu9gfrH.jpg',
 'id': 99861,
 'adult': False,
 'backdrop_path': '/rFtsE7Lhlc2jRWF7SRAU0fvrveQ.jpg',
 'original_language': 'en',
 'original_title': 'Avengers: Age of Ultron',
 'genre_ids': [28, 12, 878],
 'title': 'Avengers: Age of Ultron',
 'vote_average': 7.3,
 'overview': 'When Tony Stark tries to jumpstart a dormant peacekeeping program,
things go awry and Earth's Mightiest Heroes are put to the ultimate test as the
fate of the planet hangs in the balance. As the villainous Ultron emerges, it is
up to The Avengers to stop him from enacting his terrible plans, and soon uneasy
alliances and unexpected action pave the way for an epic and unique global
adventure.',
 'release_date': '2015-04-22'}
```

```
[32]: # Build movie dictionary.
      # (This is essentially just to get the relevant movie_ids needed in the review␣
       ↪getting step,
      #  but title and date information are retrieved as well.)
      for movie in movie_list:
          get_movie_info(movie)
```

```
[33]: # View movie_dict.
      movie_dict
```

```
[33]: {'movie_ids': [24428, 99861, 271110, 299536, 299534],
       'titles': ['The Avengers',
        'Avengers: Age of Ultron',
        'Captain America: Civil War',
        'Avengers: Infinity War',
        'Avengers: Endgame'],
       'dates': ['2012-04-25',
        '2015-04-22',
        '2016-04-27',
        '2018-04-25',
        '2019-04-24'],
       'average_score': [7.7, 7.3, 7.4, 8.3, 8.3],
       'vote_count': [21240, 14794, 15201, 16316, 10774],
       'popularity': [27.884, 31.096, 29.264, 71.641, 38.148]}
```

```
[7]: # Initialize empty reviews_dict.
     reviews_dict = {}
```

```
[8]: # Method to get list of reviews for a movie and build the reviews dictionary as
     # {movie_id : reviews_list}

     def get_movie_reviews(movie_id):
         query_url = f"https://api.themoviedb.org/3/movie/{movie_id}/reviews?
      ↪api_key={themoviedb_api_key}"
         response = requests.get(query_url).json()
         reviews_dict[movie_id] = [response['results'][i]['content'] for i in␣
      ↪range(len(response['results']))]
```

```
[9]: # Build reviews_dict.
     for movie_id in movie_dict['movie_ids']:
         get_movie_reviews(movie_id)
```

```
[13]: movie_id = 24428
      query_url = f"https://api.themoviedb.org/3/movie/{movie_id}/reviews?
       ↪api_key={themoviedb_api_key}"
      response = requests.get(query_url).json()
```

```
response['results'][0]
```

[13]:
```
{'author': 'xenocast',
 'content': 'With a movie like this you wonder how all of the otherwise, main
characters will work together and support the story.\r\n\r\nNo problems here.
While as might be expected, R. Downey Jr. comes across largely central, it is
still a good mix and IMHO the best scenes in the movie involve the generated
Hulk character.\r\n\r\nWith that kind of successful melding of characters,
Hollywood-scale egos and even computer generated characters; you have to give it
up to the writers and director to make this the successful film that it is.',
 'id': '4fce8f8819c29523880000de',
 'url': 'https://www.themoviedb.org/review/4fce8f8819c29523880000de'}
```

[15]:
```
# Check how many reviews were returned for each movie.
[len(reviews_dict[key]) for key in reviews_dict.keys()]
```

[15]:
```
[20, 20, 7, 13, 20]
```

Note: TheMovieDB API info at https://developers.themoviedb.org/3/getting-started.

[17]:
```
import json

json = json.dumps(reviews_dict)
f = open("reviews.json","w")
f.write(json)
f.close()
```

[ ]:
```

```