DLI Accelerated Data Science Teaching Kit

Module 2 Lab:
Data Collection via API

[25 pt] Collection Movie DB (TMDb) data

You will use "The Movie DB" API to: (1) download data about movies and (2) for each movie, download its 5 similar movies.

You will write some **Python 3** code (not Python 2.x) in **script.py** in this question. You will need an API key to use the TMDb data. Your API key will be an input to **script.py** so that we can run your code with our own API key to check the results. Running the following command should generate the CSV files specified in part b and part c:

python3 script.py <API_KEY>

Please refer to this tutorial to learn how to parse command line arguments. Please DO NOT leave your API key written in the code.

Note: The Python Standard Library and the requests library are allowed. Python wrappers (or modules) for the TMDb API may **NOT** be used for this assignment. Pandas also may **NOT** be used --- we are aware that it is a useful library to learn. However, to make grading more manageable and to enable our TAs to provide better, more consistent support to our students, we have decided to restrict the libraries to the more "essential" ones mentioned above.

a. How to use TheMovieDB API:

- Create a TMDb account and request for an API key at https://www.themoviedb.org/account/signup
  - After signing up, you need to request an API Key. To do so, log into your TMDb account and click Settings
  - Click API on left panel, and request the API key by generating it.
  - Select "Developer" when asked about the type of API key that you wish to register for. Accept the terms of use.
  - Fill out the form
  - Once you have submitted the form, you should be able to see your API key under the same API tab.
- Refer to the API documentation https://developers.themoviedb.org/3/getting-started/introduction, as you work on this question.

Note:

- The API allows you to make **40 requests every 10 seconds**. Set appropriate timeout intervals in your code while making requests. We recommend you think about how much time your script will run for when solving this question, so you will complete it on time.
- The API endpoint may return different results for the same request.

b. [10 points] Search for movies in the "**Comedy**" genre released in the year 2000 or later. Retrieve the 300 most popular movies in this genre. The movies should be sorted from most popular to least popular. **Hint**: Sorting based on popularity can be done in the API call.

- Documentation for retrieving similar movies:
  https://developers.themoviedb.org/3/discover/movie-discover
  https://developers.themoviedb.org/3/genres/get-movie-list
- Save the results in **movie_ID_name.csv**.
  Each line in the file should describe one movie, in the following format --- NO space after comma, and do not include any column headers:

  movie-ID,movie-name

  For example, a line in the file could look like:

  353486,Jumanji: Welcome to the Jungle

  Note:
  - You may need to make multiple API calls to retrieve all 300 movies. For example, the results may be returned in "pages," so you may need to retrieve them page by page.
  - Please use the "primary_release_date" parameter instead of the "release_date" parameter in the API when retrieving movies released in the year 2000 or later. The "release_date" parameter will incorrectly return a movie if any of its release dates fall within the years listed.

c. [15 points] For each of the 300 movies, use the API to find its 5 similar movies. If a movie has fewer than 5 similar movies, the API will return as many as it can find. Your code should be flexible to work with however many movies the API returns.

- Documentation for obtaining similar movies:
  https://developers.themoviedb.org/3/movies/get-similar-movies
- Save the results in **movie_ID_sim_movie_ID.csv**.
  Each line in the file should describe one pair of similar movies --- NO space after comma, and do not include any column headers:

  movie-ID,similar-movie-ID

  Note: You should remove all duplicate pairs after the similar movies have been found. That is, if both the pairs A,B and B,A are present, only keep A,B where A < B. For

example, if movie A has three similar movies X, Y and Z; and movie X has two similar movies A and B, then there should only be four lines in the file.

A,X
A,Y
A,Z
X,B

You do not need to fetch additional similar movies for a given movie, if one or more of its pairs were removed due to duplication.

Deliverables:

- **movie_ID_name.csv:** The text file that contains the output to part b.
- **movie_ID_sim_movie_ID.csv:** The text file that contains the output to part c.
- **script.py:** The **Python 3** (not Python 2.x) script you write that generates both **movie_ID_name.csv** and **movie_ID_sim_movie_ID.csv**.