Movie recommendation system based on Top 250 movies

Jingjie Wan

1. Project code

Link: https://github.com/jingjie-wan/SI507

Please refer to the README file for more details

2. Data source

**2.1 IMDb Top 250 Movies**

The first data source is a webpage of the top-rated 250 movies in Internet Movie Database(IMDb).

Source: http://www.imdb.com/chart/top

Format: HTML

Access & Cache:

I accessed the data by scraping. I use **requests** library to make a HTTP request to the above URL. Then I use **BeautifulSoup** to parse and extract movie information from the response content.

I used cache so HTTP request to the website only have to be made once. To be more specific, I saved the text of the web response in a html file. If the file already exists locally, the program directly read the file.

Summary of Data:

* #Records available: 250
* #Records retrieved: 250
* Description:

Every record contains basic information of one of the top 250 movies. Here are the important fields of each record:

* The rank (of rating) of the movie (*place*)
* The IMDB number of the movie, which is unique and Is used widely to identify movies in many databases (*IMDB\_number*)
* The title of the movie (*title*)
* The rating of the movie on IMDB website (*rating*)
* The year that the movie released (*year*)
* The director of the movie (*direct*)
* The main actors and actress (in list) (*stars*)

**2.2 Open Movie Database**

Since the first data source only provides limited information, I used the second database which contains detailed information about movies to complement the movie data. The two datasets can be merged by the IMDB number.

Source: https://www.omdbapi.com/

Format: JSON

Access & Cache:

I accessed the data by Web API which requires API key (http://www.omdbapi.com/?apikey=[yourkey]&). I used **requests** library again while I can only use IMDB number from the first database to make request to one movie's data at a time. I converted these json data to dictionary (which keys are the IMDB numbers of the movies).

To cache the data, I turned the dictionary containing data of each movies to json format and saved it as json file. If the program finds the file locally, it would directly read the file instead of making all the requests again.

Summary of Data:

* #Records available: About 1 million
* #Records retrieved: 250
* Description:

Every record contains detailed information a movie, including runtime, awards, rated, genre and so on. Here are the important fields of each record:

* The IMDB number of the movie (enable it to be merged with the first database to get complete information) (*IMDB\_number*)
* The runtime of the movie (*runtime*)
* The genres of the movie (in list) (*genre*)
* Languages and countries of the movie (in list) (*language*, *country*)
* whether the film has been nominated oscar award (*nominated\_oscar*)
* Box office (in dollar) (*box\_office*)

3. Data Structure

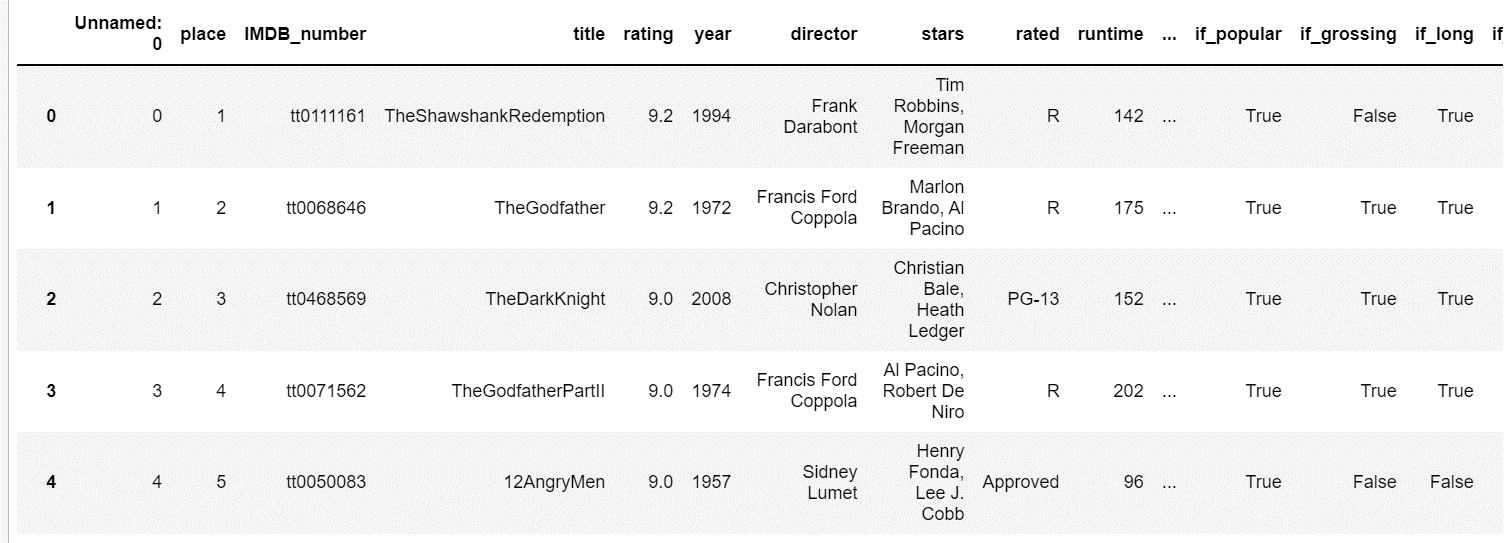
The data structure I used (tree) is described in detail in README file in the *code* folder.

Tree is saved in Tree.json.

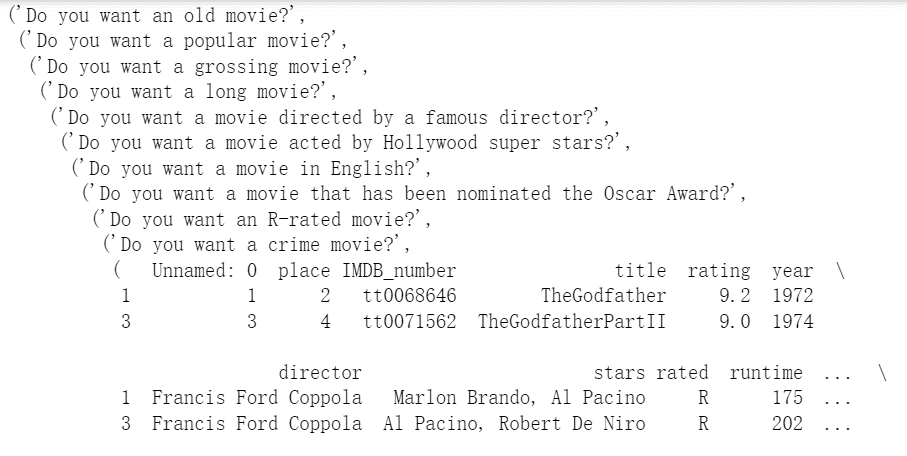
Get\_tree.py reads Tree.json and loads the tree.

Screenshots:

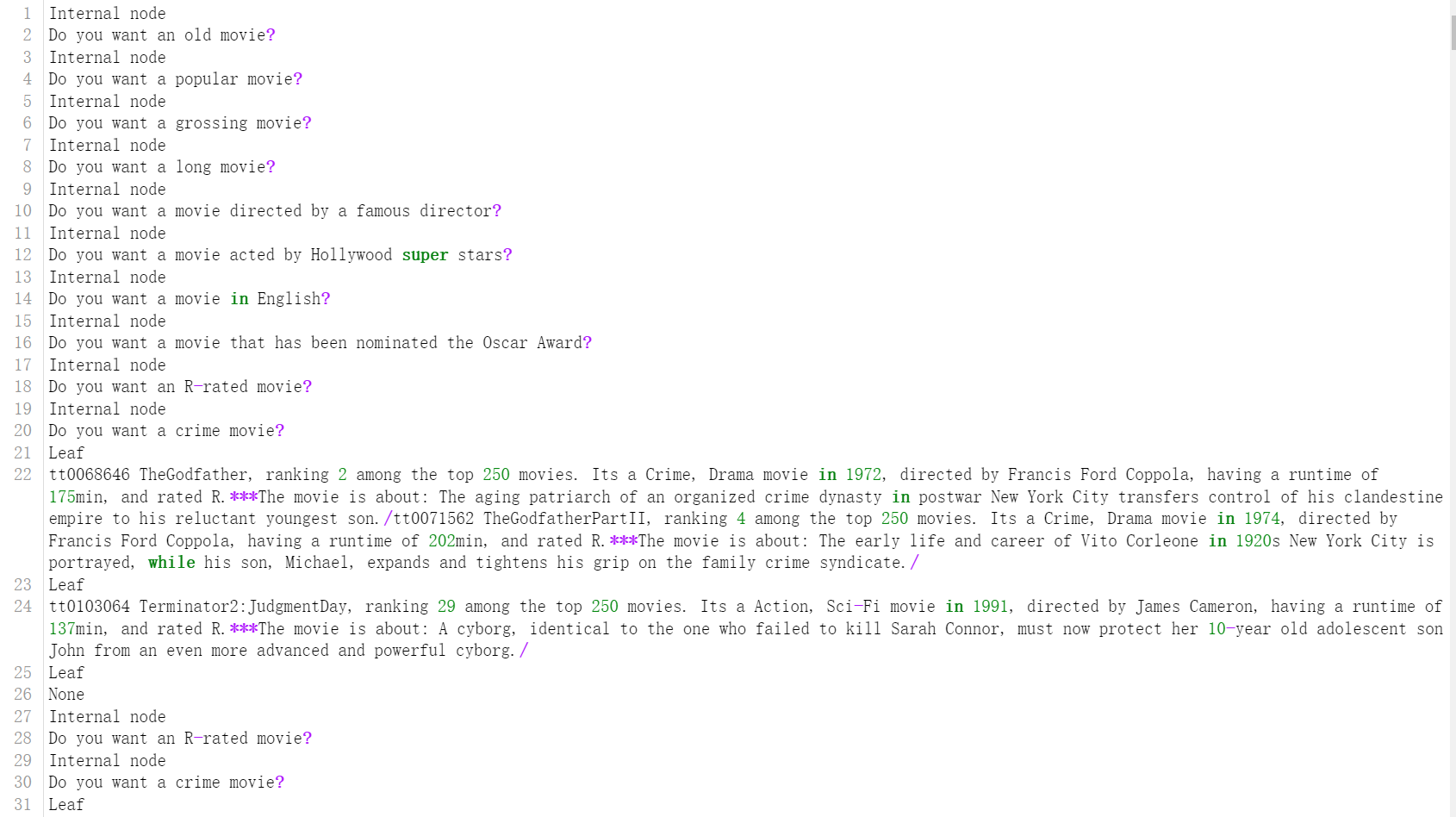
* Data (the first several columns are information of movies, followed by columns of boolean used to construct the tree)



* Data structure:
  + Tree shown in tuple (part of )



* + Tree shown in string (in json file) (part of )



4. Interaction and Presentation Options

After answering a series of questions about the requirements for movies, the user will be given four options for displaying and selecting the recommended movies.

* Firstly, two options are given: (1) see the recommended movies in simple mode (just the titles); (2) in detailed mode (including their titles, places among 250 movies, genres, released year, directors, runtime, etc.)
* Next, another two options are given: (1) see the plot of a specific movie; (2) launching a browser which jumps to the IMDB website of a specific move.

and JSON file with your graphs or trees

I mainly used command line prompts for interaction and presentation. Launching a browser that jumps to a specific movie website is also used as a presentation method.

To interact with my program, the user should follow the instructions in the command lines. The user will answer 'yes' or 'no' to a series of questions about their requirements of the recommended movies (e.g. 'Do you want a popular movie?'). Then the user can enter a number (1 or 2) to choose the demonstration modes of the recommended movie list. Thirdly, the user can choose to see the plot or browse the website of a specific movie in the list by entering the number of the movie in the list. Finally, the user can choose whether to play with the recommendation system again.

5. Demo Link

https://drive.google.com/file/d/1aBT3LEaTNbjq9eoqgST4zsLVv\_v2l3uP/view?usp=sharing

*Note: Please change the clarity on the bottom right corner to 1080P to see the command lines! (The default clarity of google drive is 720P)*