**SI507 Final Project Submission**

Catherine Hao | haorlin@umich.edu

Project code:

The link to Github repository for my SI507 final project code is:

https://github.com/Catherine-Hao/SI507-final-project

The README contains the description of each file in the repository, instructions for running my code as well as a brief description of how to interact with my program.

The Python packages for my project to work besides the built in packages include requests, flask, BeautifulSoup, webbrowser, pandas.

Data sources:

* Data #1: IMDb "Top 1000" Movies

<https://www.imdb.com/search/title/?groups=top_1000&ref_=adv_prv>

For this data source, I crawled and scraped the data of all 998 movies in 20 pages from IMDb web page. (There are 998 records of movies in this collection in total although the title saying it’s “Top 1000” movies), and caching is used in this process since this data is written into files. The data was first stored and displayed using panda data frame, and then imported into a CSV. file. The CSV. file is uploaded in the repository as movies.csv. The documentation of the movie is shown in Table. 1.

Table. 1 The fields of IMDb Top1000 Movie

|  |  |
| --- | --- |
| title | The title of the movie |
| imdb\_id | The IMDb id of the movie |
| imdb\_url | The IMDb url of the movie |
| year | The year when the movie is rwleased |
| certificate | The certificate rate of the movie |
| runtime | The runtime of the movie (unit: minute) |
| genre | The genres of the movie |
| director | The director of the movie |
| stars | The main cast of the movie |
| overview | The overview of the movie |
| rating | The IMDb rating of the movie (1-10) |
| metascore | The metacritic score of the movie (1-100) |
| votes | The number of IMDb votes |
| us\_gross | The box office of the movie (unit: million USD) |

Summary of data:

There are 1000 records available, and 1000 records retrieved. For creating the movie recommendation system, the fields including year, runtime, certificate, genre, director, stars and rating are used for conducting the query, since presumably users are more likely to have preferences in these attributes of the movie.

* Data #2: IMDb Reviews

https://imdb-api.com/en/API/Reviews/+{API\_KEY}+{IMDb\_id}

For this data source, I used the IMDb API which I haven’t used before that requires an API key. I applied for an API Key to access the data, which is: k\_nm384tne. By adding the API Key and id of IMDb, the 25 pieces of movie review data can be accessed, and caching is used in this process. The url link is displayed in a web browser, and the data can be displayed in JSON format, for example, the movie reviews for Inception is as follows:

{'username': 'SnoopyStyle',

'userUrl': 'https://www.imdb.com/user/ur2898520',

'reviewLink': 'https://www.imdb.com/review/rw7466621',

'warningSpoilers': True,

'date': '21 October 2021',

'rate': '8',

'helpful': '416 out of 703 found this helpful.',

'title': 'Villeneuve epic',

'content': "Denis Villeneuve directs this remake of the Frank Herbert classic sci-fi. It's the first part of two movies. Timothée Chalamet is the lead as Paul Atreides. The emperor has given his family the desert planet Arrakis where the most important spice is mined. It's a trap and everyone knows it.I've never read the book. I have seen the David Lynch version and the Sci Fi movies. I do miss some of the Lynchian weirdness. This version feels more Earth-bound without the weirdness. Arrakis has a lot more middle eastern influences. House Atreides is more Scottish. I do wish that they get rid of the damn bagpipes. One hopes that they would get rid of that noise in a few thousand years. As for Chalamet, he is the perfect prettiness for the role. He looks too pretty to be believed as a fighter which is exactly what every other character comments on about him. He has the ethereal look of an angelic hero in an ancient classical Greek story. I'm not sure if newbies could get lost in this labyrinthian world. It's not really that complicated. I think Villeneuve has made the right movie. He puts his skills into the epic feel of this franchise. It feels right... coming from a non-book-reader."},

...

This system would only output the reviews of the movie that is recommended, and to provide more information about the movie to help the users make further decisions on whether they would choose to watch the movie after they read the reviews from others. The documentation of the movie is shown in Table. 2.

Table. 2 The fields of IMDb Top1000 Movie

|  |  |
| --- | --- |
| username | The name of the user |
| userUrl | The url of this IMDb user |
| reviewLink | The IMDb url of this review |
| warningSpoilers | If the review is a spoiler (True/False) |
| date | The date of this movie |
| rate | The user’s rate of the movie (1-10) |
| helpful | The genres of the movie |
| title | The title of the review |
| content | The content of the review |

Summary of data:

There are 25 records available, and 25 records retrieved for each movie. The recommendation system would provide the user these review data in a readable format and help users make the decision.

Data Structure:

The basic structure of this system is a tree. I have accessed 1000 movie data from IMDb, organizes the attributes of the data into a tree, and then asks users questions (about year, runtime, genre, ratings, etc) based on their preferences until it provides one recommended movie that meets the user options.

Then, the program would provide the users with the top reviews for the movie so that they could be informed of the actual feedback and make further decisions based off of those reviews.

A tree is made up of a set of tuples (of tuples (of tuples ... )). For example, based on all the information about the data, I tired to create a tee and made a movie recommendation system. Here’s the tree with seven questions and eight answers:

Tree = \

("What type of certificate do you want for the movie, such as PG-13, R, M?" ,

("What type of genre do you want for the movie, such as Drama, Sci-Fi, Comedy? (You can input multiple genres with \" \" in between)" ,

("What year do you want for the movie? (you can input a time period like \"1990-2000\", or one certain year like \"1995\")",

("What rating do you want for the movie to have at least?" ,

("Who do you want for the director of the movie? (You can input multiple names with " " in between)" ,

("Who do you want to be in the cast of the movie? (You can input multiple names with " " in between)",

("How long do you want for the movie to last? (you can input a time range like \"80-120\")",

({movie\_runtime}, None, None),

({movie\_no\_preference}, None, None)),

({movie\_cast}, None, None)),

({movie\_director}, None, None)),

({movie\_rating}, None, None)),

({movie\_year}, None, None)),

({movie\_genre}, None, None)),

({movie\_certificate}, None, None))

By running the file which reads the JSON of the tree, the tree would look more organized like this:

Do you have preference for the movie's certificate?

+-Yes: Do you have preference for the movie's genre?

| +-Yes: Do you have preference for the released year of the movie?

| | +-Yes: Do you have preference for the movie's rating?

| | | +-Yes: Do you have preference for the movie's director?

| | | | +-Yes: Do you have preference for the movie's cast?

| | | | | +-Yes: Do you have preference for the movie's runtime?

| | | | | | +-Yes: The movie is {movie\_runtime}

| | | | | | `-No: The movie is {movie\_no\_preference}

| | | | | `-No: The movie is {movie\_cast}

| | | | `-No: The movie is {movie\_director}

| | | `-No: The movie is {movie\_rating}

| | `-No: The movie is {movie\_year}

| `-No: The movie is {movie\_genre}

`-No: The movie is {movie\_certificate}

Interaction and Presentation Options:

This system uses Flask to conduct interactive and presentation functions. Brief instructions for how a user would interact with your program. First, for every attribute the movie, the system would first ask whether they have requirement for that certain attribute, and then would be more specific about what their requirement is. Then, after this is done, the system would randomly select a movie that meet the users’ need. If there is not a movie that would satisfy the user, the system would recommend the user to try again. Lastly, the system would inform the user about the recommended movie’s title and would jump to the movie’s IMDb web page for further information. It would also provide with a list of 25 reviews of the movie.

Tree = \

("Do you have preference for the movie's cerificate?" ,

("Do you have preference for the movie's genre?" ,

("Do you have preference for the released year of the movie?",

("Do you have preference for the movie's rating?" ,

("Do you have preference for the movie's director?" ,

("Do you have preference for the movie's cast?",

("Do you have preference for the movie's runtime?",

("{movie\_runtime}", None, None),

("{movie\_no\_preference"}, None, None)),

("{movie\_cast"}, None, None)),

("{movie\_director"}, None, None)),

("{movie\_rating"}, None, None)),

("{movie\_year"}, None, None)),

("{movie\_genre"}, None, None)),

("{movie\_certificate"}, None, None))