Institute of Systems Science, National University of Singapore

MASTER OF TECHNOLOGY PROJECT REPORT

30.10.2022

—Movie Recommendation System

Team members

•LI YUHENG
•GUO HONGXI
•HUANG CHENXI
•PAULSON PREMSINGH SAMSON DHANSINGH

1 EXECUTIVE SUMMARY **2**

2 PROBLEM DESCRIPTION **4**

2.1 Project Objective 4

3 KNOWLEDGE MODELING **5**

3.1 Reasoning From Data 5

3.2 A Modeling Dataset 6

3.3 Knowledge Reasoning 7

4 SOLUTION **9**

4.1 System Architecture 9

4.2 System Scope 9

4.3 System Features 10

5 FUTURE IMPROVEMENTS **12**

6 CONCLUSION **12**

7 APPENDICES **13**

7.1 Project Proposal 13

7.2 Mapped System Functionalities against the knowledge, techniques and skills of modular courses 16

7.3 Installation and User Guide 17

7.3.1 INSTALLATION 17

7.3.2 USER GUIDE 19

7.4 Individual Project Report: Li Yuheng 26

7.5 Individual Project Report: Guo Hongxi 28

7.6 Individual Project Report: Huang Chenxi 29

7.7 Individual Project Report: Paulson Premsingh Samson Dhansingh 31

1 EXECUTIVE SUMMARY

Nowadays movies have become one of the most popular and crucial entertainment mediums in people's daily life. Also as a popular culture and art in contemporary society, the penetration, inclusiveness and coverage of movie and television art are unmatched by other arts. It affects social life by acting on people's thoughts and concepts. Movie and television appreciation can improve the aesthetic taste and artistic appreciation ability of people all around the world, and plays an important role in the quality education of people.

Meanwhile, for many people, movies not only provide material and spiritual enjoyment, but also attract them to travel to the filming location. The role of movie and television in the promotion of tourist destinations is also extensive and far-reaching.

With the rapid development of Internet and technology, more and more companies tend to build an online movie website by purchasing movies' copyright and provide users with paid online viewing services like Netflix, Prime Video, HBO Max, Tencent Video, etc. And every platform has its own recommendation system, which is very significant:

1. First, a fast recommendation system can increase user growth, increase user retention, activity, and length of stay on the web pages.
2. Most video streaming platforms use the membership-based profit model, Therefore, it is very important to increase the conversion rate of VIP and VIP retention. An accurate recommendation system can increase the purchase rate of VIP and attract users to continue watching movies that match their preferences.
3. A complete recommendation system can help save labor costs for companies. We can imagine that, with the increase of business complexity, the cost of the algorithm is basically constant, while the editing cost increases linearly. Therefore, the recommendation algorithm is widely used in extremely complex products or video companies with multiple product matrices to maintain, which can greatly reduce the Labor costs.

Let's take Netflix as an example, Netflix estimates that the personalized recommendation system saves its business as much as $1 billion a year. Building a good recommendation system faces many challenges, but it is undoubtedly of great value.

Using the techniques imparted to us in lectures, Our group use MySQL for building the whole database and deploy django to build interfaces, and use python post method to resolve user preference data sent back by the frontend(Jquery+CSS3+HTML5). The algorithm written by Python will give a feedback recommendation result back to the frontend.

Our team had an amazing time working on this project, and hopes the project can be used by users. For sure we will continue improving the movie recommendation system to make it be more international and more accurate.

2 PROBLEM DESCRIPTION

Every time when we finish watching a movie on an online movie platform, we would have a basic rate of the movie we just watched in our mind. And of course we are not willing to waste our time on watching movies that are not interested in, but also unwilling to spend much time on searching and finding a movie that we might love. So we need a recommendation to tell us which movie to watch in our next relaxing time, based on our rate and review on the previous movie.

But many of us have faced the problem that the recommendation result from the website is utterly irrelevant with ones we are truly interested in. We hold the view that many factors can cause such a situation, for example, some platforms take the users' click and watching time into consideration, and this may create a misleading that the user might just unexpectedly click the page or just casually have a look of this movie but not into it. So We believe the key factor in recommending the movie should be directly related to the users' own preference.

2.1 Project Objective

We hope to build a complete and user-friendly movie recommendation system and try to acquire the data from the user: if a user has watched the movie, we would like to invite him/her to give a score (overall 5 points) and submit to our database, and then our recommendation system can deploy the recommendation algorithm to recommend a movie. For example, If a user has watched a fiction movie *Batman:The dark knight rises*, and then he/she will give a rate, if the rate is above about 3.5, then the system will give back another fiction movies like *Superman* to the user; if the rate is around 1, then the system may recommend the user to watch a comedy movie.

It is worth noting that we are creating a recommendation system for users to conveniently and quickly find movies that they may be interested in, but not a real video stream platform. And we use a movie dataset from 2018 which is sufficient in the implementation of the recommended function. Commercially, we hope that the system can be deployed on the video platform as a microservice, users can use it directly on the web with the same account of the platform, and finally through the interface offered by the platform, users can directly click the recommended movie to watch. In this way, the system can also obtain some profits through distribution channels.

3 KNOWLEDGE MODELING

Before employing these patterns to produce new recommendations, we must create an engine that learns and detects patterns in a user's watching history.

- Data: Find and import the crucial libraries with movie datasets that already have worldwide ratings. ML systems need data.
- Analysis: Generic movie suggestions should be made using the dataset that is already available.
- Personalization: By submitting your own movie ratings, you can receive customized ratings.
- Strategy: Use a similarity-based collaborative filtering technique.
- Combination: To obtain a decent estimate across the ratings, combine suggestion lists. Both filtering models may now use the pooled dataset of movie ratings.

3.1 Reasoning From Data

Usually, there are two ways in knowledge reasoning: reasoning from data and reasoning from experts. In the way of reasoning from experts, knowledge acquisition can be hard, making the system increasingly complex. The objects we recommended are movies and we intend to recommend them to personal users. Vissarion Belinsky pointed out that Reading Shakespeare's drama shows that each person is a legitimate artistic subject, however low he stands in the social hierarchy and even in humanity as such[1]. We are building a personalized movie recommendation system, instead of general-purpose methods for describing and solving a broad class of useful problems.

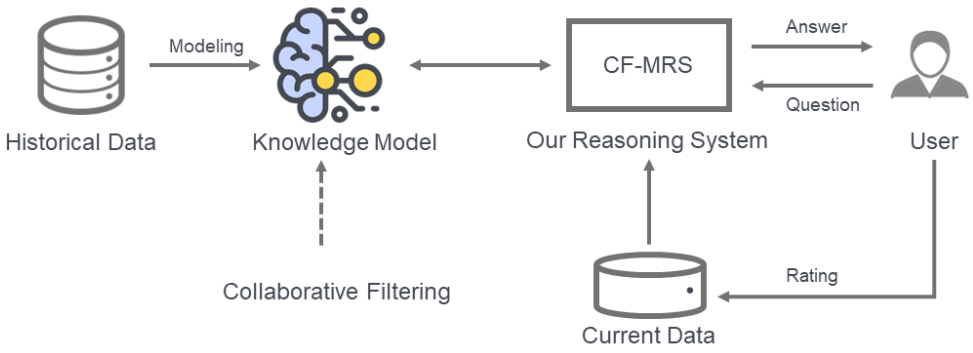


Fig.3.1 Knowledge Modeling: Reasoning From Data

[1] Response to The Muscovite Journal; see: Complete Works, vol. 10, 1956, p. 242

As shown in figure 3.1, we get a dataset and knowledge representation(section 3.2), use collaborative filtering modeling it to build our reasoning system, and repetitively refine it by users' new ratings(section 3.3).

3.2 A Modeling Dataset

We use the Scrapy Python web scraping library to collect the raw data from the IMDB website. To visualize and complement, we also obtain movie-poster data with 21908 movies from an open-source kaggle dataset[2].

imdbId	title	poster
3	Pauvre Pierrot (1892)	https://images-na.ssl-images-amazon.com/images/M/MV5BNGRkODFmNjYtYjU2Yy00YTI1LWFlYzc
5	Blacksmith Scene (1893)	https://images-na.ssl-images-amazon.com/images/M/MV5BNDg0ZDg0YWYtYzMwYi00ZjVlLWI5Yz
8	Edison Kinetoscopic Record of a Sneeze (1894)	https://images-na.ssl-images-amazon.com/images/M/MV5BNzFkN2JkNjEtZDZhNi00NmVmLTg4YW
12	The Arrival of a Train (1896)	https://images-na.ssl-images-amazon.com/images/M/MV5BZjE2MGVkMTAtMWIwYy00YzQ5LWE2
14	Tables Turned on the Gardener (1895)	https://images-na.ssl-images-amazon.com/images/M/MV5BMDMxMGI2ODctMjI2OS00YmQ5LWJj
192	Ella Lola, a la Trilby (1898)	https://images-na.ssl-images-amazon.com/images/M/MV5BZDgzZWZlNWQtNTE5YS00ODRmLTky
399	Jack and the Beanstalk (1902)	https://images-na.ssl-images-amazon.com/images/M/MV5BMjAzNTI3MzI0NI5BMI5BanBnXkFtZTcw
417	A Trip to the Moon (1902)	https://images-na.ssl-images-amazon.com/images/M/MV5BNGNjZjEyODgtMDhkNi00YjFlLWExOT
420	Alice in Wonderland (1903)	https://images-na.ssl-images-amazon.com/images/M/MV5BODQyZWYxZWYtNDRmZC00ZjQxLTIh
439	The Great Train Robbery (1903)	https://images-na.ssl-images-amazon.com/images/M/MV5BYzM5YTkzNzItYmQ4Ny00MWRjLWExM
455	The Music Lover (1903)	https://images-na.ssl-images-amazon.com/images/M/MV5BZDU1Mjk5YTEtOTcwYS00N2EyLTk0OD
498	Rescued by Rover (1905)	https://images-na.ssl-images-amazon.com/images/M/MV5BZTIkYmExMjMtYjU5Mi00NDYxLWE0YW

Fig.3.2 Movie-poster Dataset

userId	imdbId	rating	id
1	115641	1.0	12
1	67116	4.0	13
1	84827	4.0	14
1	80801	3.0	15
1	96446	2.0	16
1	120587	2.0	17
1	91064	2.5	18
1	81633	1.0	19
1	71230	3.0	20
2	113189	4.0	21
2	114388	5.0	22

Fig.3.3 User Rating Dataset

For model simplicity and low computational cost, we consider structural knowledge in the knowledge base. Structural knowledge can be regarded as a heterogeneous network with multiple types of entities and multiple types of links to express the structure of the knowledge base.[3] For our movie recommendation, the entities include movie items and

[2] https://www.kaggle.com/datasets/neha1703/movie-genre-from-its-poster?select=MovieGenre.csv

[3] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative Knowledge Base Embedding for Recommender Systems. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16). Association for Computing Machinery, New York, NY, USA, 353–362. https://doi.org/10.1145/2939672.2939673

corresponding attributes, i.e. ratings. And links describe the relationship between these entities. We intend to quantify similarity between item entities, which is useful for recommendation.

3.3 Knowledge Reasoning

For Knowledge Reasoning, we use a similarity-based collaborative filtering(CF) to recommend what our users most probably like. CF digs out a small number of users with similar tastes to the single user in massive data. More specifically, we use two branches in CF, user-based CF and item-based CF to recommend. Considering the balance between accuracy performance and model complexity, we calculate the user-ratings matrix by cosine similarity.

Given two user rating vectors A and B, the similarity is:

$$\text{similarity } = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

where A_i, B_i represent the components of A, B, respectively.
For the missing values in the user-rating matrix, we assume missing values are zeros.
After calculating the similarity, for a target user u, we sum the ratings of other users weighted by their similarity to the target user:

$$\text{Predicted rating } (u, i) \ = \ \frac{\sum_{v \in V} \text{CosSim}(u, v) \cdot r_{v,i}}{\sum_{v \in V} |\text{CosSim}(u, v)|}$$

where V is the set of other users, and i is the movie that needs to be given a predicted rating. Intuitively, the predicted rating is determined by the ratio of rating similarity to user similarity. The rating similarity is proportional to the predicted rating, and user similarity is inversely proportional to the predicted rating. For example, given a movie i with low rating similarity and high user similarity, the predicted rating of i is low, since you probably dislike what your similar users dislike.

However, in the implementation of our system, we reason out the movie that users probably like every time receiving recommendation requests, which becomes computationally prohibitive. Worse yet, it's going to get worse when used by large

numbers of users under our very limited computing resource. The huge pressure from computational load forced us to find a more effective way out.

Therefore, we use item-based CF to optimize our recommendation. item-based CF is similar to the user-based CF above, but from the point of view of the item itself, rather than the user. For example, users who like item A all like item C, then it can be known that item A is highly similar to item C, and user C likes item A, then it can be inferred that user C may also like item C.

In item-based CF, we transpose the user-rating matrix to get distances between items not users. Then we predict user *u*'s rating for a movie *i* using:

$$\text{Predicted rating } (u,i) \;=\; \frac{\sum_{m \in M} \text{CosSim}(i,m) \cdot r_{u,m}}{\sum_{m \in M} |\text{CosSim}(i,m)|}$$

where *M* is the set of *K* similar movies to *i*(among movies that *u* has rated).

In fact, in terms of the time to give recommendation results, using item-based CF is much slower than user-based CF, because the number of movies in the movie dataset is much larger than that of users. However, we are building a sustainably growing system. As the number of users grows, eventually better performance will be shown when using item-based CF.

4 SOLUTION

4.1 System Architecture

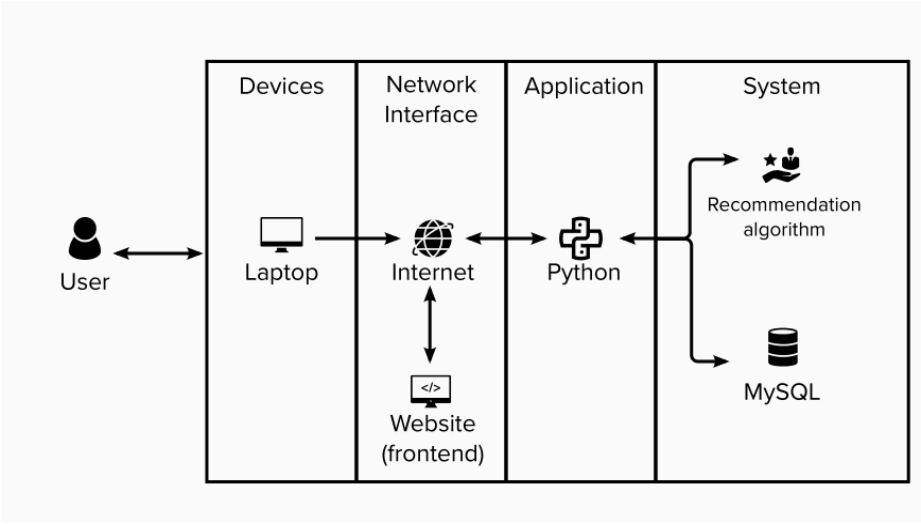


Fig.4.1 - System architecture

Web(Front-end):
We use Html, CSS, and Javascript,bootstrap for the front-end design.

Backstage(Backend):
Django framework is used to build the background interface, python is used to parse the user preference data(rate of movies) sent back by the front end using the get/post method, and the preference information is deduced by the recommendation algorithm to get the recommended movie list. After matching through the background SQL-based matching system, the detailed information of matching movies will be returned to the front end.

4.2 System Scope

Although it is sufficient for us to simulate and implement the function of a comprehensive movie recommendation system, the data of the project is restricted to all movies' information is before 2018, because we didn't have enough time to crawl the latest movie data.

And finally the project will provide a movie recommendation system that is based on the ratings of users, rather than according to the users' click times. And the whole system is deployed on localhost, users can use their localhost to access it.

And the scope of the system is also limited by the frontend design, for example the database can only get the rating data from users if they press the submit button after doing the rating.

4.3 System Features

Today is an era of information explosion, and the market is full of different types of movies. It becomes a complex problem for customers to choose a film to watch among various movies. Generally, people choose the movies they want to watch through movie rankings, friend recommendations, or recent popular movies on the homepage of video websites. However, the movie recommendation system can recommend more attractive movies to users through algorithms such as collaborative filtering.

Advantages:

(1) SYSTEM'S INTELLIGENCE & ROBUSTNESS

This movie recommendation system is very intelligent because it does not require users to answer any questions about their favorite movie. It can recommend a list of movies to users based on the user's rating of the films on the homepage through a collaborative filtering algorithm. Additionally, the system will also recommend a list of movies that the user and others with similar interests.

The system is also very robust. There are many fault tolerance settings in the system to improve its robustness. When the user inputs the wrong password or doesn't have an account, the system will prompt the user to enter the correct username and password or register a new account. Besides, users can also view the history of movies they have rated.

(2) SCALABILITY

There are many movies released every year, so updating the movie data (timely) is very important. This system stores movie data in a database, making it very convenient to update the movie information.

(3) EASY TO ACCESS

Users don't need to download any software. They only need to enter the correct URL to use this system, which is very convenient.

Disadvantages:

（1）It is suitable for occasions with only one user. If there are multiple registered users, only the previous person's data is deleted from the database, so that the second user can use the system.

(2) Every refresh is regenerating the recommendation list, which should be removed, and the refresh should not change.

（3）If a new user rates a few movies, he cannot get personalized recommendations immediately.

（4）The database can only get the rating data from users if they press the submit button after doing the rating.

5 FUTURE IMPROVEMENTS

(1)Expand the database

The database of the current system is not large enough and more data needs to be added to make the data more universal. At present, the database basically contains relatively popular movies and lacks niche movies. Therefore, niche movies should be added to provide users with more choices.

Add detailed information about the movie. The system only provides the primary movie name and poster. For a better user experience, movie introductions and related movie reviews can be added later. In addition, a viewing link of the movie can be added with authorization.

(2) Optimize the system

At present, the system does not support multiple users using the system to recommend movies at the same time. In order to achieve better usability, the algorithm/model can be optimized to achieve simultaneous use by multiple people.

(3)Try multiple algorithms

The algorithm currently used by the system is relatively simple, and only collaborative filtering. To improve the accuracy, multiple algorithms can be combined to apply in the system like NCF.

6 CONCLUSION

Through this project, our group members have a better understanding of the knowledge learned in the class. In the process of practicing, we learned how to build a complete system. For example, design the system, webpage, and SQL. Although we encounter many difficulties in the process, we finally solve them and complete the project.

Building the system itself reinforced some of the learning points taught in the courses like Reasoning System and Cognitive System. In fact, group members have never done a project related to the recommendation system and learned Django and CF before. To finish, we learned relative knowledge and built the system which is challenging but interesting.

7 APPENDICES

7.1 Project Proposal

Date of proposal: 30 Oct 2022
Project Title: Movie Recommendation System
Group Members (Name, Student ID): •HUANG CHENXI A0261955W •GUO HONGXI A0261887M •LI YUHENG A0261798L •PAULSON PREMSINGH SAMSON DHANSINGH A0261986M
Background/Aims/Objectives: Background: For video platforms: With the rapid development of the Internet and technology, more and more companies tend to build an online movie website by purchasing movies' copyright and provide users with paid online viewing services like Netflix, Prime Video, HBO Max, Tencent Video, etc. And every platform has its own recommendation system, which is very significant. First, a fast recommendation system can increase user growth, increase user retention, activity, and length of stay on the web pages. Most video streaming platforms use the membership-based profit model, therefore, it is very important to increase the conversion rate of VIP and VIP retention. An accurate recommendation system can increase the purchase rate of VIP and attract users to continue watching movies that match their preferences. A complete recommendation system can help save labor costs for companies. We can imagine that, with the increase of business complexity, the cost of the algorithm is basically constant, while the editing cost increases linearly. Therefore, the recommendation algorithm is widely used in extremely complex products or video

companies with multiple product matrices to maintain, which can greatly reduce the Labor costs.

For users:
There are many movies released every year, which means that choosing a movie among various genres to watch is a bit difficult. Generally, people choose the movies they want to watch through movie rankings, friends' recommendations, or recent popular movies on the homepage of video paltforms. Not only is deciding which movies to watch through a movie recommendation system efficient, but also users are able to choose more attractive movies. Therefore, for users, a video website that supports the recommendation system will be a better choice.

Objectives:

Build a complete and user-friendly movie recommendation system and try to acquire the data from the user: if a user has watched the movie, we would like to invite him/her to give a score (overall 5 points) and submit it to our database, and then our recommendation system can deploy the recommendation algorithm to recommend a movie. For example, If a user has watched a fictional movie Batman: The dark knight rises, then he/she will give a rate, if the rate is above about 3.5, then the system will give back another fictional movie like Superman to the user; if the rate is around 1, then the system may recommend the user to watch a comedy movie.

Project Descriptions:

Front-end:
The project consists of a front end and a back end. For the front end, we use HTML, CSS3, and JavaScript to design the webpage and realize/implement three main functions:

1. User Interface
(1) Create a new account
For new users, they have to create a new account before using the movie recommendation system, which is helpful to store their information like rating history. Then the data of the user's account such as username, email address, and password will be transferred to the database and stored.

(2) User login and logout interface
After registering an account, the user is able to log his account and use the movie

system. When the user gets the recommended movie list, he can log out.

2. Rating the movie that the user has watched before
The user can rate the movie listed on the index webpage by selecting the number of stars. Different numbers of stars represent different evaluations of the movie like recommendation, like, general, dislike, etc. Then the data will be transferred and stored in the database .

3. Show the recommended results to users
Run the data into the model, get the result, and transfer it to the web page. Then the user will get a list of recommended movies.

Backstage (Backend):
Django framework is used to build the background interface, python is used to parse the user preference data (rate of movies) sent back by the front end using the get/post method, and the preference information is deduced by the recommendation algorithm to get the recommended movie list. After matching through the background SQL-based matching system, the detailed information of matching movies will be returned to the front end.

Methods:

Collaborative filtering:
Collaborative filtering is a recommendation algorithm designed based on user behavior. Specifically, it finds a certain similarity (like the similarity between users or similarity between items) through group behavior and makes decisions or recommendations for users through similarity. In addition, collaborative filtering includes two operations of collaboration and filtering. Collaboration is the collection of feedback, evaluations, etc. from all users by interacting with the website continuously. As for filtering, through the information obtained through/by collaboration, filters screen out the items that users are interested in from a large number of items.

CF digs out a small number of users with similar tastes to a single user in massive data. More specifically, we use two branches in CF, user-based CF and item-based CF to recommend.

User-based CF is to find neighbor users (similar users) based on the user's preference for items, and then recommend the things that neighbor users (similar users) like to the current user. Item-based CF is to find similar items based on the user's preference for items, and then recommend similar items based on the user's historical preferences.

7.2 Mapped System Functionalities against the knowledge, techniques and skills of modular courses

Modular Courses	Mapped System Functionalities
Machine Reasoning	1. The system uses existing users ratings knowledge to reason. In terms of reasoning form, the system uses analogical reasoning(similarity-based reasoning). 2. The system aims to solve a synthetic task. The system is a reactive reasoning system (data driven) system. 3. For storage, the system uses a structured database and structured query language technique. 4. For knowledge acquisition, the system uses a data mining model to discover knowledge.
Cognitive Systems	The system uses 3V - framework to apply AI: ● Viable: Data-driven recommendation ● Valuable: deliver movies that fit users' tastes in a auto-decision way ● Vital: analyzing large amounts of users movie ratings
Reasoning Systems	1. The system implements an explainable AI: Similarity-based Reasoning. 2. Collaborative Filtering approach is used, including user-based CF and item-based CF. 3. The system measures the similarity by cosine similarity. Also, the system handles missing values by ignoring pairwise complete when measuring.

7.3 Installation and User Guide

7.3.1 INSTALLATION

Refer to Github README.md file[4]

There are two ways to use this project.

- **Easy usage**
- **Custom usage** - Convenient for custom development. You need to follow the packaging instructions to package manually. Unless you are familiar with Python and Django, this approach is not recommended.

Easy usage

- **Download**

  Download the ZIP file and unzip it locally

- **Create Database**
  1. Configure MySQL connection:

     host='127.0.0.1', port=3306, user='root', password='root'

  2. Import data: execute *moviedata.sql* in MySQL
- **Run**
  1. Double click *\movieRMD\movieRMD.exe*
  2. Browser access http://127.0.0.1:8000/

Custom usage

1. Perform **Download** & **Create Database** in **Easy usage**.
   Make *\source-code\movierecommend* the project file

   **NOTE: Make sure the project directory includes the static folder. If not, be sure to create a new static folder, and then create a new static_root folder under the static folder!*

[4] https://github.com/Stanley7096/IRS-PM-2022-10-30-IS04PT-GRP-MovieRecommender#section-5--user-guide

2. *Optional check 1*:
   \django_auth_example\urls.py includes codes:

```
from django.conf.urls import static
from . import settings
// The following code must be on a new line
urlpatterns += static.static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
```

   \django_auth_example\setting.py includes codes:

```
STATIC_ROOT = os.path.join(BASE_DIR, 'static', 'static_root')
```

3. Execute the **python manage.py collectstatic** command to collect static files into static_root

4. Execute the **pyinstaller manage.spec** command to package

5. Copy the */static* and */template* folders to the */dist/manage directory*

6. Copy the */users/static* folder to */dist/manage/users*

7. *Optional check 2*:
   Go back to the project directory.
   movieRMD.py should include codes:

```
import os
os.system('manage.exe runserver 8000 --noreload')
input()
```

8. Choose any of the following ways to execute

// Execute with custom icon
pyinstaller -F --icon=jieni.ico movieRMD.py
// Execute without custom icon
pyinstaller -F movieRMD.py

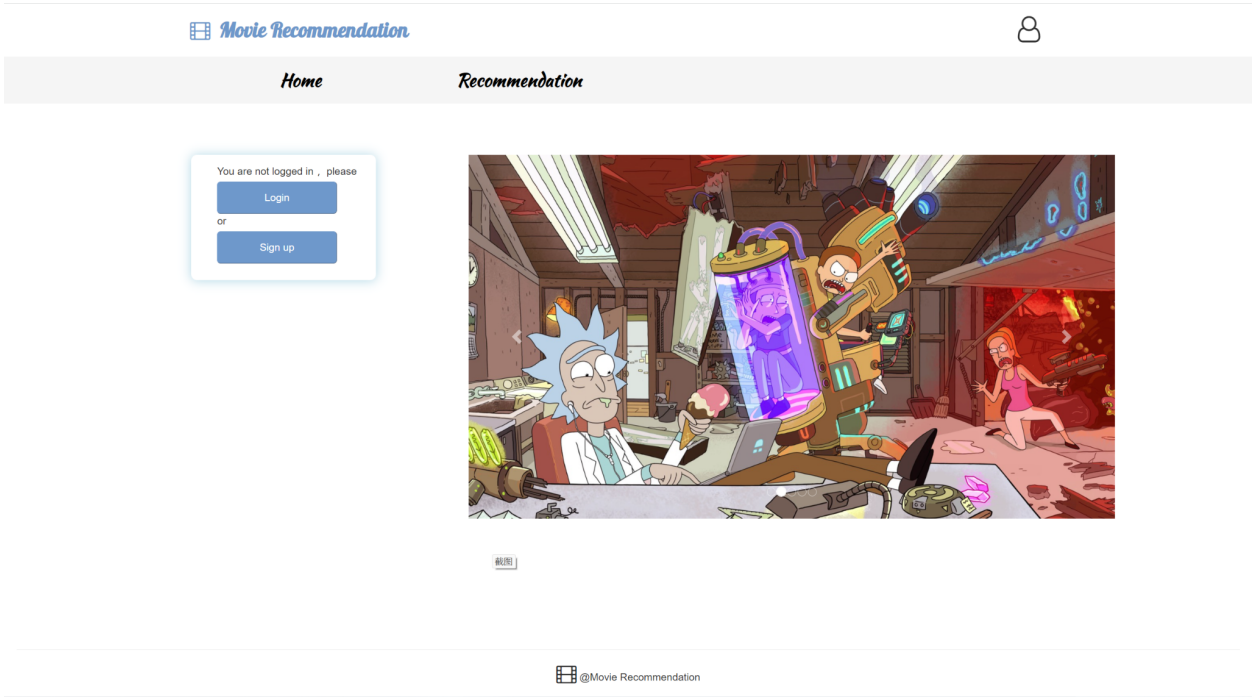
9. A *movieRMD.exe* will be generated in the dist folder and copy it to *dist/manage/*

10. Zip the dist folder, rename and replace the existing *movieRMD* folder

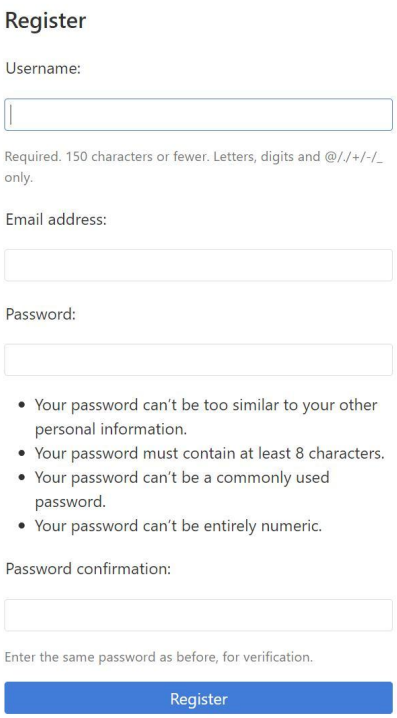
11. Perform **Run** in **Easy usage**

7.3.2 USER GUIDE

Enter the URL in the browser: http://127.0.0.1:8000/



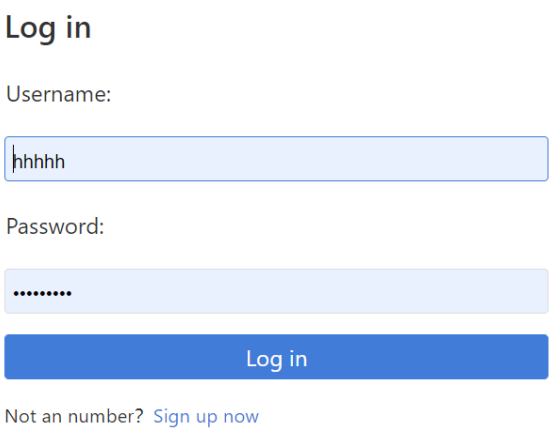
Click ‘sign up’ button to create an account



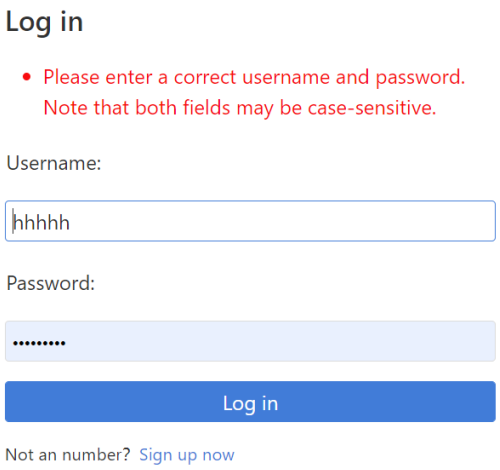
If the password doesn't meet the requirement, it will have an alert .

- This password is too short. It must contain at least 8 characters.
- This password is too common.
- This password is entirely numeric.

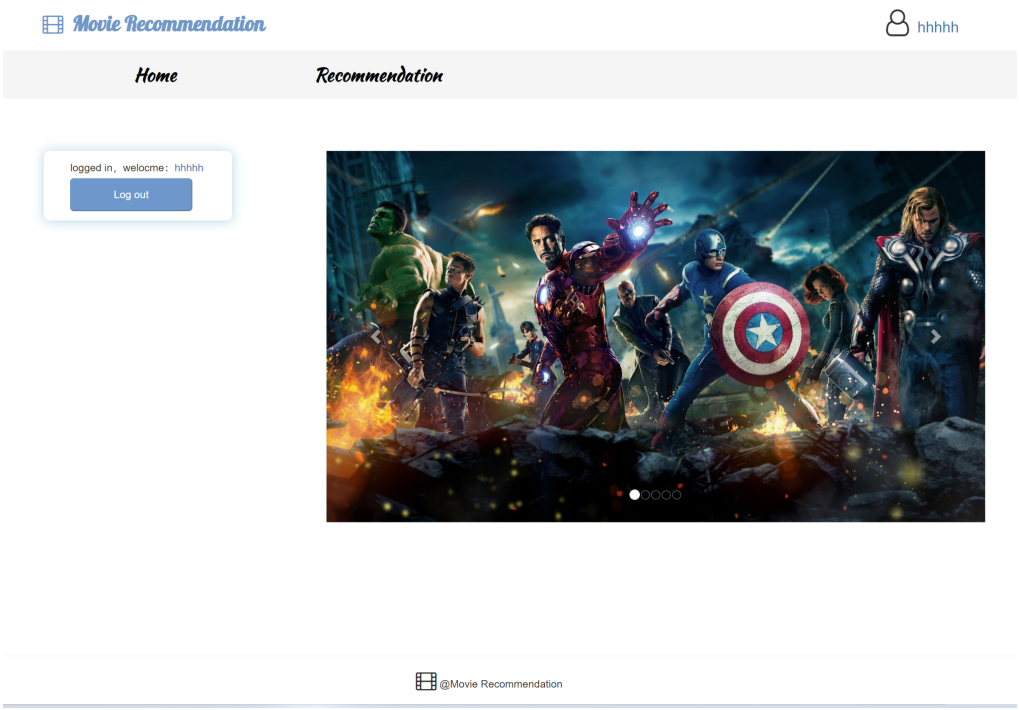
Click 'Log in' button to login



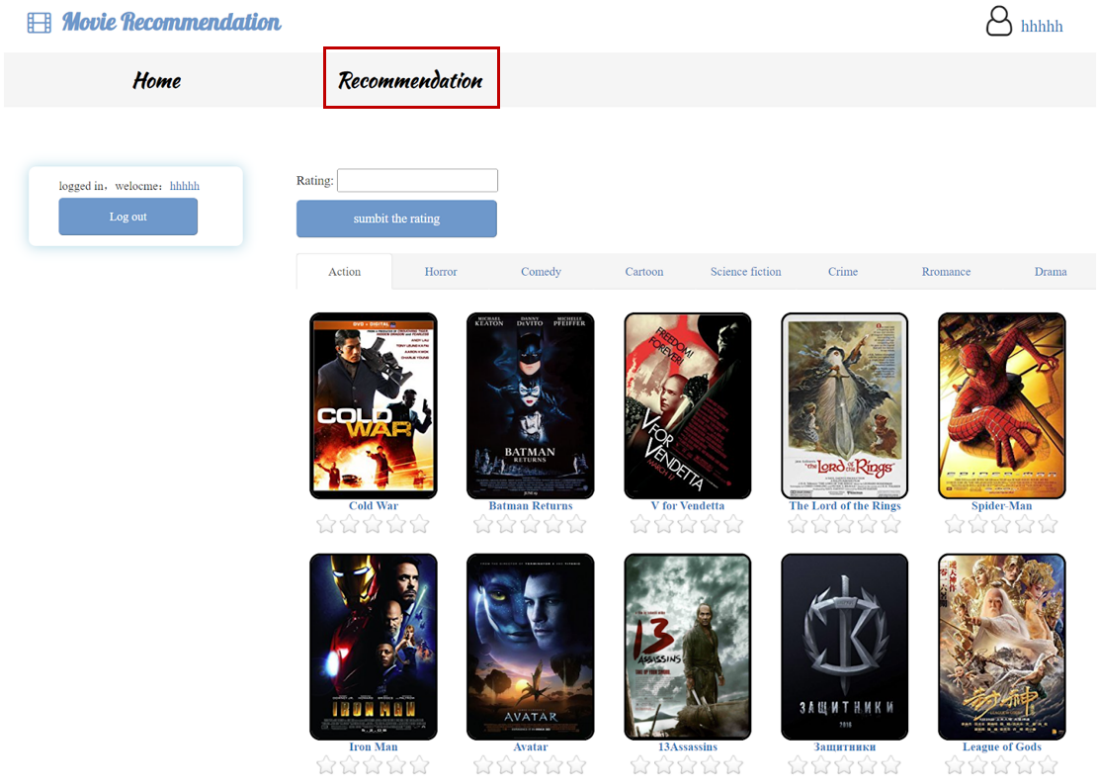
If the username and password are wrong. It will have alert

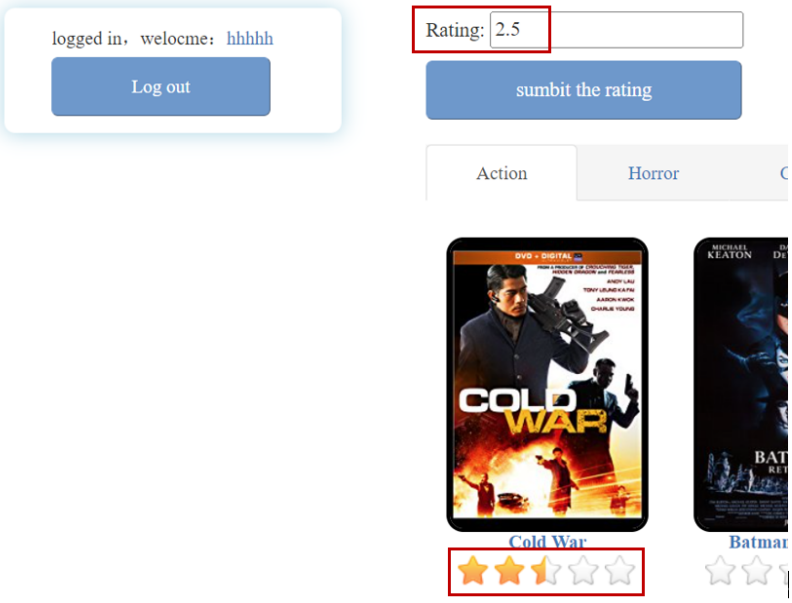


If the username and password are correct. It will jump to the ‘Home’ page.

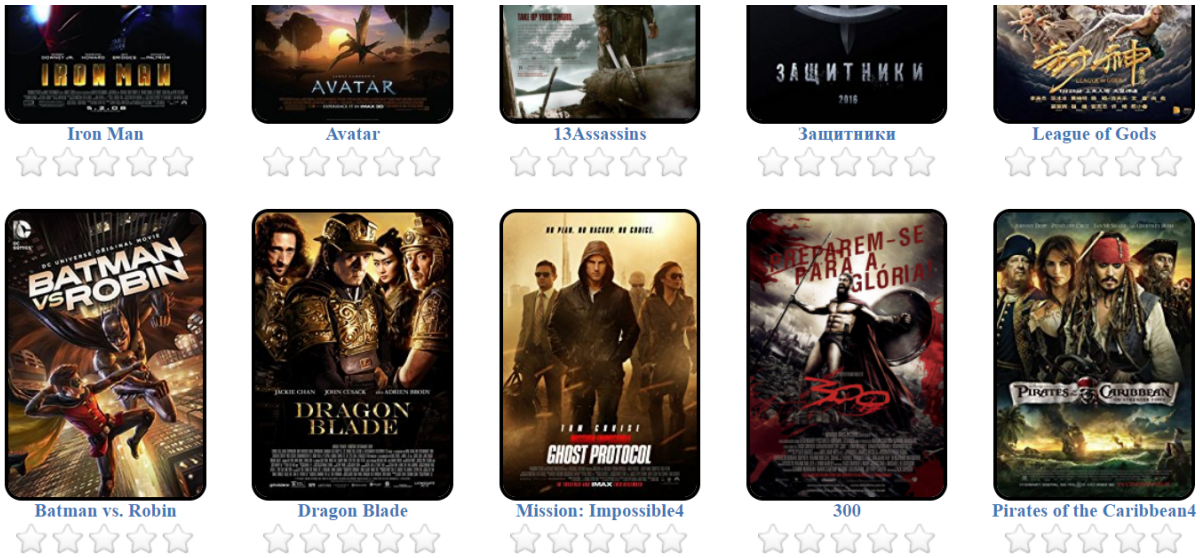


Click the ‘Recommendation’ link, It will jump to the Rating page. Users can rate the movie he/she has seen before through selecting the number of the star.





Click the ‘View recommended movies’ buttonto get the recommended movies list. According to the users’rating ,the system will automatically recommend a list of movies for users.



Movie Recommendation hhhhh

Home Recommendation

Recommend Result:

People with similar interests to you also liked the following movies:

Pulp Fiction (1994) Jurassic Park (1993) Speed (1994) True Lies (1994) Se7en (1995) The Mask (1994)

View recommended movies

@Movie Recommendation

Movie Recommendation hhhhh

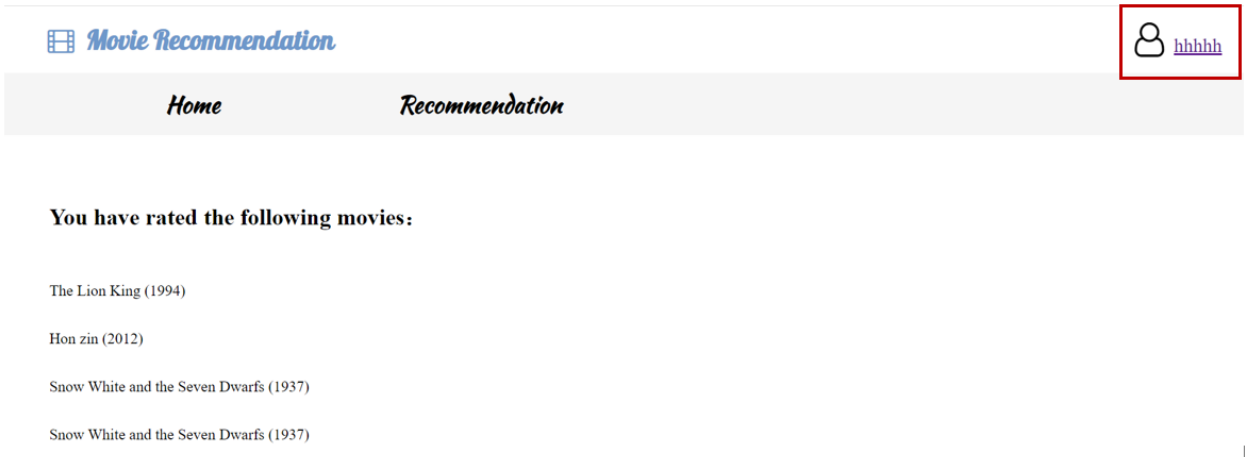
Home Recommended

Movies similar to your favorite movies:

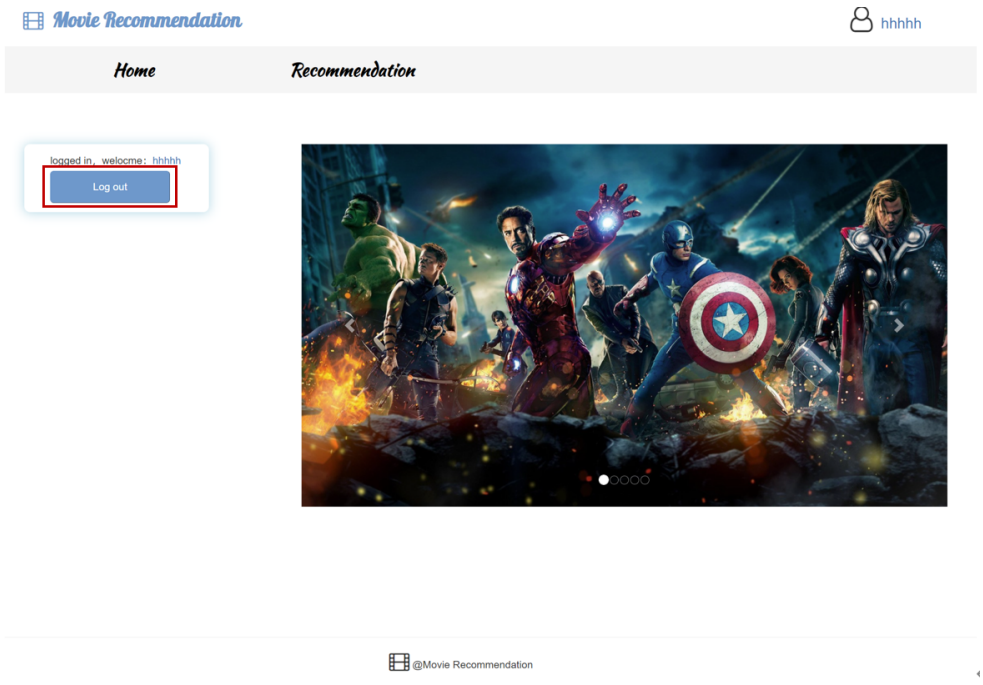
Jurassic Park (1993) Pulp Fiction (1994) Speed (1994) Dances with Wolves (1990) The Mask (1994) True Lies (1994)

Batman (1989)

Click the user name to jump to message page which provide the users' rating history.



Finally the user can log out of the system



7.4 Individual Project Report: Li Yuheng

Li Yuheng A0261798L

1. Your personal contribution to the project.

This project experience is quite rewarding and interesting for me, and with the great help of my team members, I can immerse myself into the project from the very beginning to the report part.

In actual development, I was mainly responsible for the backend design and database (Django and MySQL), and also helped other team members with the algorithms design. Also I was always actively involved in the team meeting to discuss what we should do next, and always gave my suggestions and made a project plan for team members. And I was very honored to work with such enthusiastic and interesting members, and enjoyed the whole process with them.

2. What you have learnt from the project.

From this project, I have learnt the basic and important part of an actual reasoning system, and I gained crucial understanding and knowledge mainly from the backend, database part.

I firstly learnt the basic usage of Django, Django is a high-level Python web framework and Django itself is based on the MVC model, that is, the Model (model) + View (view) + Controller (controller) design pattern. And I learnt the basic method to do the request: GET and POST methods are two HTTP request methods. The GET method is used to get some information from resources, and the POST method is used to submit some form data that needs to be processed and get responses.

So for my job, I just need to invoke the algorithm part in the back-end and make sure the data can be got or posted and be stored in the database.

As for the database part, I worked together with my teammate hongxi, and we learnt to use navicat to import raw data into tables in the created database and use MySQL Workbench to manage the whole database.

3. How you can apply this in future work-related projects.

The knowledge and skills acquired could be applied at my workplace to build reasoning systems with the combination of frontend and backend program. Like in the course Pattern Recognition System, we have to train a neural network model and use this model to build a system. In the next project we can totally use the knowledge from this project to do the frontend and backend work.

Also I can apply the knowledge from here to do data analysis by virtue of the complexity of raw data in every project. So it is very crucial for me to go deep in this area.

7.5 Individual Project Report: Guo Hongxi

GUO HONGXI A0261887M

1. Your personal contribution to the project.

In this project, I'm responsible for designing the front end which consists of HTML, CSS, and JavaScript. My main work is to design the webpage including the layout and style of the page. Specifically, implement basic page jumping, page column division, and divide the page into different modules, each of which implements different content and functions. For example, the user login module requires the user to input the correct user name, and email address password to register a new account. After signing up, they can log in with the correct username and password. The navigation bar module provides different links to other pages. The movie evaluation module collects the user's rating of the movie.

Besides, I also implement front end and back end interactions and other work like writing reports and making a video.

2. What have you learnt from the project?

Doing this project helped me get familiar with how to build a recommender system. In addition, I also learned some basic knowledge of Django and bootstrap and how to use them to build front-end web pages including static web page layout design and dynamic function implementation. In the process of practice, I have a deeper understanding and application of the interaction between the front and back ends, such as the transmission of data between the front and back ends.

Additionally, I learned to cooperate better with team members, communicate with others effectively, overcome difficulties and achieve our ultimate goal.

3. How you can apply this in future work-related projects.

This project provides me an opportunity to improve my ability to build the front-end.And the knowledge I acquired in this project can be applied in future work like the construction of a data visualization platform and design more user friendly and functional websites.

7.6 Individual Project Report: Huang Chenxi

Huang Chenxi A0261955W

1. Your personal contribution to the project.

In this project, my work involves structured database creating, knowledge modeling, recommendation algorithms design and corresponding coding for the back end. For system implementation, I deploy the project deliverables: web application. I also took part in the project code review for quality control. In terms of the group project report, I elaborated section 7.2 appendices-mapped system functionalities, section 7.3.1 and section 3 knowledge modeling, including section 3.1, section 3.2, section 3.3. My personal contribution is shown in the table below.

Contribution	Detail
Structured database creating	• Data collecting • Data preprocessing • IMDB movies and users ratings datasets creating
Knowledge modeling	• Measuring cosine similarity between users ignoring pairwise complete • Reasoning by user-based collaborative filtering and item-based collaborative filtering
Coding	• Coding in Python for back-end recommendation • Joint debugging with Django front-end part
Deployment	• Deploying the web application
Quality control	• Taking part in code review
Group project report	• Elaborating section 7.2 appendices-mapped system functionalities, section 7.3.1 • Elaborating section 3 knowledge modeling, including section 3.1, section 3.2, section 3.3

2. What you have learnt from the project.

In application development, I have never built a Python-based web application before. Instead, I am more familiar with Java-based web application development. So I learnt how to build a web application in Python and Django (Python web framework). Given so many powerful open-source AI libraries and toolkits in Python(e.g. Tensorflow, Keras, Pytorch) being popular, and the growing demand for data analytics, machine learning related applications, it is important to master python-based application development skills.

In a practical view, I also learnt that the two ways of user-based collaborative filtering and item-based collaborative filtering bring different performance in an actual movie recommendation system. In the early stage, it is suffering for users waiting to get recommendation results when the system applies user-based collaborative filtering. Because the number of movies in the movie dataset is much larger than that of users. However, as the number of users grows, eventually the user-based collaborative filtering can reduce more burden on the system than item-based collaborative filtering.

3. How you can apply this in future work-related projects.

For the future work-related projects, this project provides an excellent example of a similarity-based recommendation system in web application format.

From a developer view, this work can be extended to develop other forms of executable products, such as mobile application, web application using public IP deploying on a cloud server.

From an AI designer view, the current algorithm can be modified for faster or more accurate recommendation. Otherwise, it can be replaced by another efficient model.

7.7 Individual Project Report: Paulson Premsingh Samson Dhansingh

Paulson Premsingh Samson Dhansingh - A0261986M

1. Your personal contribution to the project.

My primary focus was involved in development and design of the web application using Django. I also worked on the database using MySQL. I was also part of planning of the project at different phases mainly in the algorithm development part of the entire system. I collaborated in the knowledge modeling part of the project report and in making and processing the video.

My team members and I were effectively involved in brainstorming the ideas at every point where we encountered issues. I spent a considerable amount of time learning and implementing the specific tasks of the project.

2. What you have learnt from the project.

This project has mainly helped me evolve in the field of reasoning systems and explainable AI. It has also helped in gaining knowledge and experience in prediction analysis.

Before implementation of the project I was unaware on how to execute the collaborative filtering approach. I used the open-source dataset and learnt to implement the user-based collaborative filtering approach. I also learnt to predict the blanks in the utility matrix which was the goal of the recommendation system using this approach.

The project particularly helped me in sharpening the skills and additional training opportunities and more opportunities to practice. Concerning my area of expertise, I helped my team reach the ultimate project goal successfully.

3. How you can apply this in future work-related projects.

The learning experience of different technologies has helped me in advancing my skills needed in a workplace.

Almost every form of website can be (and has been) built using Django, from wikis and content management systems to social networks and news websites. It can send material in practically any format and integrate with any client-side framework. This helps me to further opportunities and in troubleshooting the problems if any are encountered. As an AI engineer, I will think and improve the efficiency of the algorithm.