

MOVIE RECOMMENDER SYSTEM

IRS PROJECT REPORT

TEAM MEMBERS

GU ZHONGHAN CHEN PENGWEN

TANG YI WANG JIARUI

1 EXECUTIVE SUMMARY

With the rapid popularization of the Internet and the development of the film industry, the amount of movie information on the Internet is quite large, and people's demand for obtaining interesting movie information for entertainment is also increasing. Personalized movie recommendation systems have become a hot topic of discussion. The representation method of movie information is relatively complex, and various existing similarity calculation methods have their own advantages. The management of a large amount of movie data and user data also becomes increasingly complex as the number of data increases. How to combine the advantages of various algorithms to give users reliable movie recommendation results and allow users to access correct recommendation data has become an important issue. This system uses crawler to crawl movie data on movie websites, combines collaborative filtering algorithm, content-based recommendation method, statistics-based recommendation algorithm and other algorithms in the recommendation module, and combines offline recommendation to effectively solve the problem of system recommendation questions of accuracy. In the project, the design and implementation of front-end visualization pages, back-end business processing, and algorithms are implemented, and movies that users may prefer are recommended.

1.1 MARKET RESEARCH

There are few movie platforms or websites in the market. And most of them are movie rating lists based on scores from thousands of users, such as IMDB, Douban and Rotten Tomatoes. This type of movie website cannot make personalized recommendations according to the user's wishes.

However, our movie recommendation system fills a gap in the market. In addition to displaying the most popular movie trends on the homepage of the website based on the rating value and number of ratings, our system also requires users to make some choices to make targeted recommendations.

The target user groups of the movie recommendation system include: 1. Individual users. Individual users can use our system to find their favourite

movies. 2. Private cinema owner. Private cinemas usually provide viewing space for couples or friends to relax and entertain. And if the private cinema is equipped with our system, it will enable customers to get a more customized experience. 3. Video platform companies that hold relevant film copyrights. Our system can be sold and embedded to companies in need to provide movie recommendation services to their users.

2 PROBLEM DESCRIPTION

With the increasing maturity of Internet technology and the development of entertainment culture, people increasingly rely on the Internet to obtain entertainment information. In the era of the information explosion, the amount of information has risen sharply. Entertainment information is also increasingly abundant. How to find the information people need in the vast sea of data has become a hot research topic. Movies, as one of the main daily spiritual entertainment items, also have the problem of information overload. To solve this problem, this project proposes a personalized movie recommendation system.

2.1 PROJECT OBJECTIVE

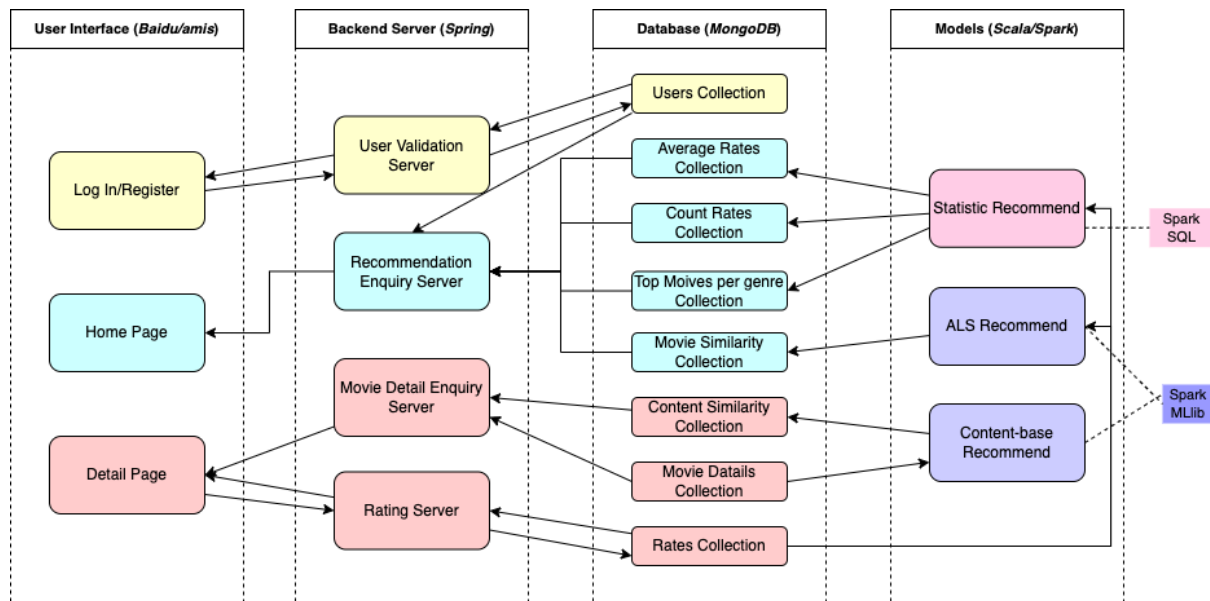
Our project aims at building a movie recommendation system, which contains two modules. One module is to show the popular movies, according to the value of ratings and the times of ratings respectively. In detail, the interface of our system shows a group of top-popular movies and a group of top-rating movies. This module mainly aims at providing users with the trend of movies. The other module is to make movie recommendations personally, according to the user's preference. For users who use the system for the first time, the system will ask users to select the genres they like, and then make recommendations based on their selection.

We aim to provide the user with an easy-using but powerful user interface, which provides a variety of ways for the user to find movies that they may be interested in. Due to the commercial and convenience reason, we integrate our system into a website which is adaptive to phones and laptops, users can explore it only by a simple sign-in step. On our website, users are allowed to search movies with keywords, or visit the detail page of each movie and rate them. Taking the rapid

development of media technology into consideration, the movies in the system are from Movielens which only contains movies within this century.

3 RECOMMENDATION SYSTEM DESIGN

3.1 SYSTEM ARCHITECTURE



Above is the figure which briefly describes the system architecture of our project. The final system design may have some slight differences from the figure because to better illuminate the general idea we simplify the communications between layers and remove some functions which were designed for extensibility. Our whole project is deployed to the cloud server provided by Aliyun and can be divided into four parts: a user interface, a backend server, a database and the recommendation models.

We use Baidu amis to build our frontend, it is a framework based on React. By using such a framework, we can build websites with just JSON, which greatly increases efficiency. It is also highly flexible and can fit any screen size. Our backend server is deployed on a different port which is hidden from the customers, it is built using Java Spring. It provides APIs to the frontend and loads the data from our database. It will return a proper recommendation list depending on different types of users. For example, if there is a new user it will return a list based on the preferred genres user selected when logging in for the first time. But

for a user with a rating history, it will return the result based on latent-factor recommendation.

The database we are using is MongoDB. As mentioned before we use a source dataset from Movielens and manually scrap the details and rates of each movie from their website. After obtaining the database we run all three models written by scala using the Spark framework and store them in separate collections. By doing this, our backend can easily read them from the database and can return the recommendation list of movies to the frontend without doing much calculation. The statistical model and latent-factor model will run regularly and update the result in the database.

3.2 ALGORITHM FOR RECOMMENDATION

3.2.1 STATISTICS-BASED RECOMMENDATION

Our recommendation system represents two lists of popular movies for our users. One list contains those movies with the highest average rating scores, and the other one contains those movies with the most times of ratings. We use Scala and SparkSQL to do the statistics process, generate and update two collections in MongoDB. Then our system will read the collections in MongoDB to get the top 10 movies for each list and represent them to our users.

For new users, once they log in, our interface will ask them to select the genres they prefer. Then for personal recommendation, the system selects a few movies that belong to the genres, and with the highest average rating scores. To realize these, we first define each genre, and add the average rating score to the movie table, then calculate the Cartesian product between each movie vector and genre vector. Ultimately we generate and update a movie collection with the highest rating scores in each genre to MongoDB using Scala. Our system will read the collection from MongoDB once a new user selects a few genres, and represents the personal recommendation to the new user.

3.2.2 CONTENT-BASED RECOMMENDATION

Content-based recommendation systems recommend an item to a user based on a description of the item and a profile of the user's interest[1]. Generally speaking, our content-based recommendation utilizes the preference of users and the similarity between different movies to make recommendations. Once the system gets the rating matrix like Table 3-1, it could decide whether to recommend a new movie to each user.

Table 3-1 Rating Matrix

	Movie1	Movie2	Movie3	Movie4
Amy	4	3	2	NaN
Bob	2	3	NaN	5
Cathy	NaN	5	1	3

Here is how the algorithm makes the decision. For Amy, we need to decide whether to recommend Movie4 to her. First, we calculate the similarity between Movie4 and Movie1, 2, 3. Besides the table above, we also have a collection that contains all the movies and their properties, including the actors, directors, genres, publication dates, etc. Given Movie4, we can compare it with other movies using Cosine-Similarity, according to their attributes. Besides, we add tf-idf(term-frequency times inverse document frequency) weights to represent the importance of each value(text) of each property.

$$w(t, d) = \frac{tf_{t,d} \log(\frac{N}{df_t})}{\sqrt{\sum_i (tf_{t_i,d})^2 \log(\frac{N}{df_{t_i}})^2}} \quad (3-1)$$

The $tf*idf$ weight, $w(t, d)$, of a term t in a document d is a function of the frequency of t in the document ($tf_{t,d}$), the number of documents that contain the term (df_t) and the number of documents in the collection (N).

Once we get the similarity between Movie4 and Movie1, 2, 3, we can calculate the weight of each movie, and then estimate the rating of Amy towards Movie4 using those weights . If the rating were higher than a given threshold, then the

system should recommend Movie4 to Amy. For each user(e.g. Bob and Cathy), the recommendation process is the same as above.

In our project, for content-based recommendation, we first read the data from MongoDB, then choose the genres and directors as attributes to calculate the similarity between movies using the tf-idf model which needs training. In the Scala machine learning library, the tf-idf model is already given. According to the similarity, we generate a matrix including the similarity between each movie in the database. Then the system can make recommendations based on the similarity-matrix.

3.2.3 LATENT FACTOR MODEL FOR RECOMMENDATION

It is often the case that the rating matrix is sparse, which means only a few movies are scored by each user and content-based recommendation would not perform that well. Recommendation based on the latent factor model can solve this problem. The main idea of the latent factor model is to find or learn the latent factors, which indicate the relation between users and movies. Indeed, the latent factors could be unexplainable. The rationale of the latent factor model is shown in Figure 3.1, with the first matrix being the original rating matrix, the second one being the user matrix and the third one being the item matrix and LF being short for latent factor. The method to get the latent factors is called matrix factorization.

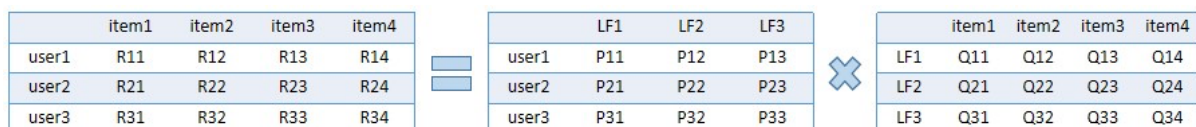


Figure 3.1 Rationale of the latent factor model

One of the precise methods of matrix factorization is Alternating Least Squares(ALS). The main idea of ALS is to fix one matrix and calculate the other one, then fix the other one and calculate the one fixed in the last iteration. During the training process above, the product of the two matrices will converge to the original rating matrix, using the MSE loss function and gradient descent methods.

The whole algorithm can be represented as below. With λ being the regularization factor, n_{ui} being the number of items the user i has rated and n_{vj} being the number of times the item j has been rated, $r_{i,j}$ being the value in the original rating

matrix, U being the user matrix, V being the item matrix, u_i^T and v_j are vectors in the U, V matrix respectively.

$$\arg \min_{U,V} \sum_{\{i,j|r_{i,j} \neq 0\}} (r_{i,j} - u_i^T v_j)^2 + \lambda (\sum_i n_{ui} \|u_i\|^2 + \sum_j n_{vj} \|v_j\|^2) \quad (3-2)$$

In our project, we use the Scala machine learning library, which involves the ALS class for training the latent-factor model. We use the grid-search method to find a group of hyper-parameters(rank and lambda) which make the model loss as low as possible. The hyper-parameter rank represents the dimension of the latent factor, which is 300 in our project; The hyper-parameter lambda represents the regularization factor, which is 0.1 in our project. After training the latent-factor model, we get another similarity-matrix for movies, which is different from the one in content-based recommendation. Based on this similarity matrix, the system will estimate the rating towards a new movie for users and decide whether to recommend it to each user after comparing the rating score with a threshold.

4 CONCLUSION

In this project, we are building a movie recommendation system and helping the user to discover the latest movie matching their preference. Our system is designed using a four-layer structure and is fully deployed online. The backend will return a pre-calculated recommendation result by our model and display it on the frontend. Our model will also run regularly to obtain the latest recommendation result on users' recent ratings.

For the part of algorithms, we make analysis to our database using statistics knowledge, and select movies with the highest rating scores and with the most rating times respectively. We use the content-based algorithm to recommend similar movies for each movie, according to genres and directors of movies. For personal recommendation, we use latent-factor to calculate the similarity of each movie, during which we use the ALS method to learn the latent factors. Then based on the similarity of movies that users prefer, the system makes recommendations for each user personally.

At present, our recommendation system has one drawback: it is not capable of making recommendations in real-time. In the future, it is promising for our project to add a function to realize a real-time module for recommendation.

APPENDIX OF REPORT A

Project Proposal

APPENDIX OF REPORT: PROJECT PROPOSAL

Date of proposal: 10/10/2022
Project Title: Movie Recommendation System
Group ID (As Enrolled in LumiNUS Class Groups): Group 8 Group Members (name , Student ID): Chen Pengwen A0261632L Gu Zhonghan A0261825A Wang Jiarui A0261860E Tang Yi A0261851E
Sponsor/Client: <i>(Company Name, Address and Contact Name, Email, if any)</i> None.
Background/Aims/Objectives: Our project aims at building a movie recommendation system for users to explore and find their favourite movies. The system will contain two modules. One module is to show the popular movies, according to the value of ratings and the times of ratings respectively. This module mainly aims at providing users with the trend in movies. The other module is to make movie recommendations personally, according to the user's preference. For users who use the system for the first time, the system will ask users to select the genres they like, and then make recommendations based on their selection.
Requirements Overview: <ul style="list-style-type: none">● Research ability● Programming ability● System integration ability
Resource Requirements (Hardware, Software and any other resources) <ul style="list-style-type: none">● Cloud Server: Aliyun Cloud Server (Ubuntu 20.04) -- 2 vCPU 4GB Memory● Reasoning systems: Statistics-based recommendation, Collaborative filtering, Latent-factor algorithm, Optimization● Language: Scala 2.11.8, Java 8● Deep learning tools: spark-sql_2.11, spark-mllib_2.11● Database: MongoDB 5.0.13● Environment: Node.js, Nginx● Framework: Springboot 2.6.3, Baidu/amis● Tools: Maven, Cron

Procedures	Objective	Key Activities
Requirement Gathering and Analysis	The team should meet with ISS to scope the details of project and ensure the achievement of business objectives.	<ol style="list-style-type: none"> 1. Gather & Analyze Requirements 2. Define internal and External Design 3. Prioritize & Consolidate Requirements 4. Establish Functional Baseline
Technical Construction	<p>To develop the source code in accordance to the design.</p> <p>To perform unit testing to ensure the quality before the components are integrated as a whole project</p>	<ol style="list-style-type: none"> 1. Setup Development Environment 2. Understand the System Context, Design 3. Perform Coding 4. Conduct Unit Testing
Integration Testing and acceptance testing	To ensure interface compatibility and confirm that the integrated system hardware and system software meets requirements and is ready for acceptance testing.	<ol style="list-style-type: none"> 1. Prepare System Test Specifications 2. Prepare for Test Execution 3. Conduct System Integration Testing 4. Evaluate Testing 5. Establish Product Baseline
Acceptance Testing	To obtain ISS user acceptance that the system meets the requirements.	<ol style="list-style-type: none"> 1. Plan for Acceptance Testing 2. Conduct Training for Acceptance Testing 3. Prepare for Acceptance Test Execution 4. ISS Evaluate Testing 5. Obtain Customer Acceptance Sign-off
Delivery	To deploy the system into production (ISS standalone server) environment.	<ol style="list-style-type: none"> 1. Software must be packed by following ISS's standard 2. Deployment guideline must be provided in ISS production (ISS standalone server) format 3. Production (ISS standalone server) support and troubleshooting process must be defined.

APPENDIX OF REPORT B

Mapped System Functionalities against knowledge,
techniques and skills of modular courses: MR, RS, CGS

APPENDIX OF REPORT: MAPPED SYSTEM FUNCTIONALITIES AGAINST KNOWLEDGE

KNOWLEDGE-BASED REASONING TECHNIQUES:

Data pre-processing, including feature selection, feature scaling, data cleaning, etc.

Machine learning methods, including selecting appropriate loss function, grid-search for tuning hyper-parameters, traversing the input for the train() method, etc.

Using statistical knowledge to recommend popular movies.

BIG DATA MINING RULES:

We build a database in MongoDB, and use the Scala language to process the big data(rating matrix, movie tables, etc.). For personal recommendation, we use the content-based algorithm and the latent-factor algorithm. In detail, we select cosine-similarity for each movie; we select the ALS method to learn the latent factors; we add appropriate regularization for the model, etc.

SYSTEM DESIGNED WITH COGNITIVE TECHNIQUES:

To interact with our users, we design a website with a user interface. Users can search for keywords(titles, genres, actors, directors, etc.) of movies. For new users, our system will ask them to select their preference for movie genres.

APPENDIX OF REPORT C

Installation and User Guide

APPENDIX OF REPORT: INSTALLATION AND USER GUIDE

CONTENT

- 1. Use The Online Movie Recommendation System**
- 2. Deploy The System Yourself**
- 3. Customize The System**

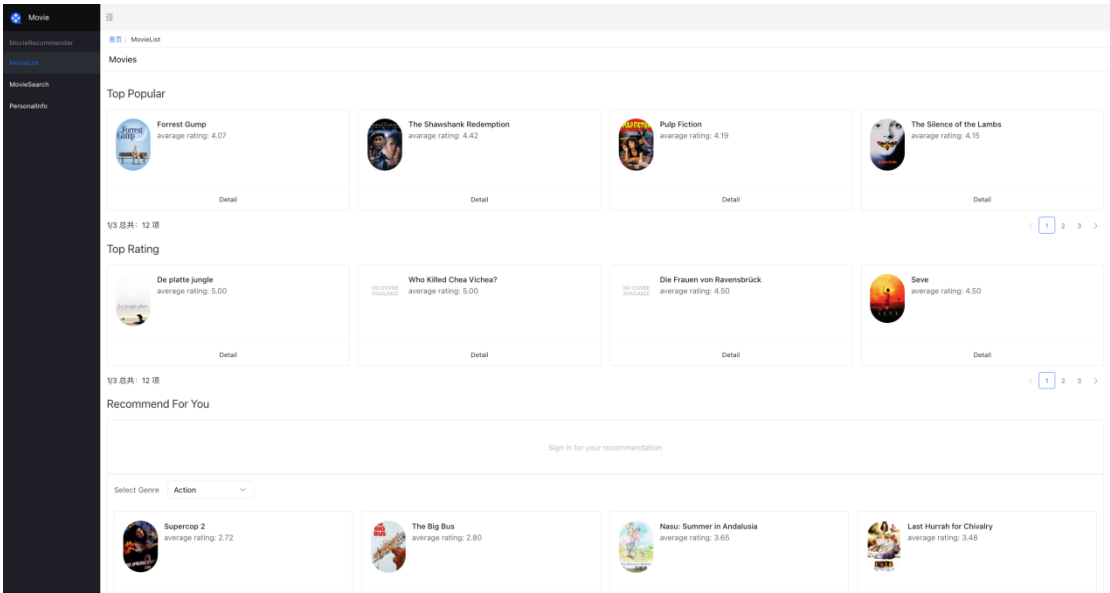
1. Use The Online Movie Recommendation System

Prepare

A device which has a modern web browser.

Enter The Website

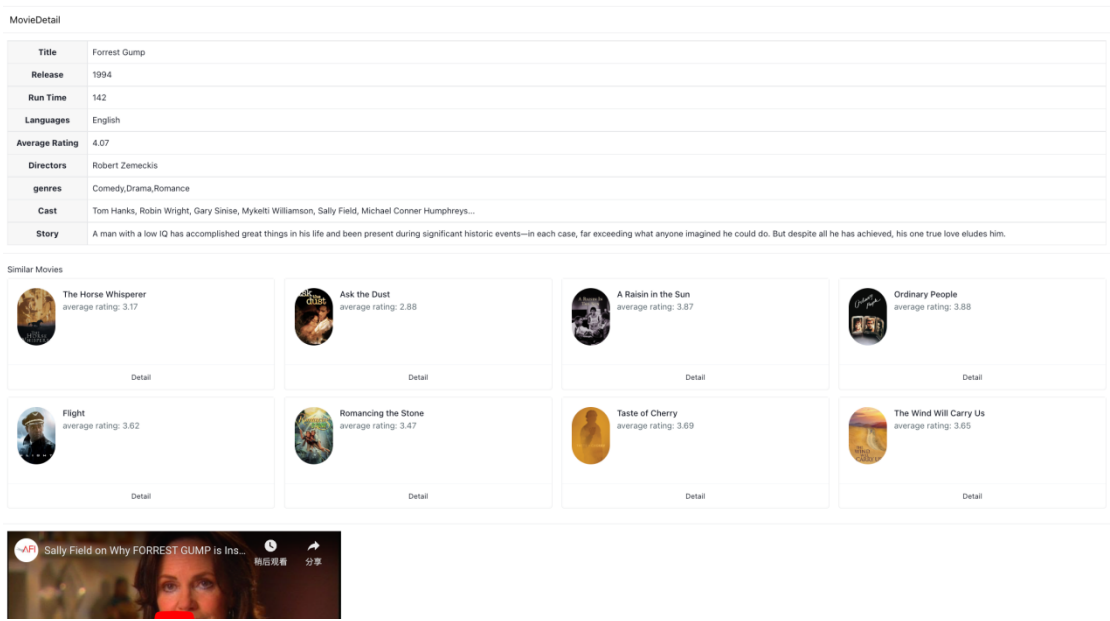
Launch a web browser, go to <https://nus2.com>



Now you can see different movie lists. But “Recommend For You” is empty, because you haven’t logged in.

MovieDetail

Each movie card has a “Detail” button. Push the button, then you enter the movie detail page.



On this page, you can not only get the basic information, posters and YouTube video about this movie but also get a similar movie list which is calculated by our algorithm.

MovieSearch

Title	Poster	Release Year	Language	Runtime	Director	Genres	
Terminator Velocity		1994	English, Russian, Italian	102	Deren Sarafian	Action	Detail
Terminator 2: Judgment Day		1991	English, Spanish	137	James Cameron	Action, Thriller, Science Fiction	Detail
The Terminator		1984	English	108	James Cameron	Action, Thriller, Science Fiction	Detail
Terminator 3: Rise of the Machines		2003	English	109	Jonathan Mostow	Action, Thriller, Science Fiction	Detail
The Terminal		2004	Russian, English, French, Russian, Spanish	128	Steven Spielberg	Comedy, Drama	Detail

On MovieSearch page, you enter the keyword, and then get a movie list as the searching result.

PersonalInfo

Personal

Sign In

username

password

[Sign In](#) [Sign Up](#)

Rating History

Title	Poster	Your Rating	Rating Time
-------	--------	-------------	-------------

In PersonalInfo page, if you are a new user, you are supposed to sign up to release more functions.

Hello, tangyi2000

[Sign Out](#)

Choose you prefer genres.

Select your prefer genres

[Submit](#)

After you sign up and sign in, you can select your preferred genres, which help us to recommend movies for you.

Personal

Hello, tanggy2000


Sign Out

Choose you prefer genres.

Action Adventure


Submit

Rating History

Title	Poster	Your Rating	Rating Time	
Forrest Gump		4	2022-10-29 19:47:42	Detail


You can also rate a movie on MovieDetail page, and your rating history is showed on your PersonalInfo page.

Recommend For You




Three Billboards Outside Ebbing, Missouri
average rating: 4.00

[Detail](#)




Escape to Victory
average rating: 3.24

[Detail](#)



The Great Escape
average rating: 4.09

[Detail](#)



Cosmos: A Personal Voyage
average rating: 4.34

[Detail](#)

1/5 总共: 20 项

< 1 2 3 4 5 >

And now, you can have your own recommend list in the MovieList page.

2. Deploy The System Yourself

All of these above are guidance for Ubuntu 20.04 System.

Please pay attention to the software and system versions, we do not guarantee that the same results can be obtained under different versions.

You can download the files from:

<https://github.com/MichaelGu718/IRS-PM-Group-8-Movie-Recommendation-System>

change every “{password}” in the code with “nusiss”

First, download package information from all configured sources.

sudo apt-get update

2.1 Back End

Install Java 8.

sudo apt install openjdk-8-jdk

confirm Java 8 is installed successfully.

java -version

Download the “MovieServer.jar”, and enter the download directory.

nohup java -jar MovieServer.jar </dev/null &>/dev/null &

Now the spring application is running on localhost:8088.

2.2 Front End

install Node.js and npm

sudo apt install nodejs npm

confirm that Node.js is installed successfully.

nodejs --version

Download the MovieRecommendSystemFrontEnd.zip and unzip it.

unzip MovieRecommendSystemFrontEnd.zip

Go into the directory, install the dependencies and start the application.

cd MovieRecommendSystemFrontEnd.zip

npm i

npm start

2.3 Reverse Proxy

Then, you need to configure the reverse proxy by using Nginx.

Install Nginx.

sudo apt install nginx nano -y

Edit Nginx config file.

sudo vim /etc/nginx/sites-enabled/default

In the server{} block, add content:

location /api/ {

rewrite ^/b/(.*)\$ /\$1 break;

proxy_set_header Host \$host;

proxy_set_header X-Real-IP \$remote_addr;

proxy_set_header X-Forwarded-For \$proxy_add_x_forwarded_for;

proxy_pass http://localhost:8088/;

}

location / {

rewrite ^/b/(.*)\$ /\$1 break;

proxy_set_header Host \$host;

proxy_set_header X-Real-IP \$remote_addr;

proxy_set_header X-Forwarded-For \$proxy_add_x_forwarded_for;

proxy_pass http://localhost:3000/;

}

Then save it.

Now visit “localhost”, you can see the page.

3. Customize The System

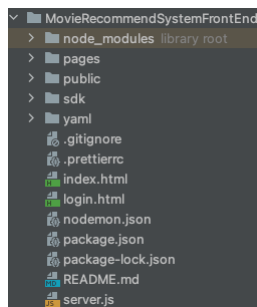
Our system may not meet all your needs. So you can make changes and improvements based on the existing code.

3.1 Front End

Baidu amis

We use Baidu amis framework to develop the frontend.

Amis is a low-code front-end framework launched by Baidu. It provides rich components and powerful renderers, using JSON configuration to generate pages.



If you want to learn how to use amis, you can refer to the online documentary.

<https://baidu.github.io/amis>

Other Possibilities

Since our system is on the separation of back-end and front-end, you can re-develop the front end, using whatever framework you want, like React, Vue, Angular and so on. You can also develop Android or iOS applications. You only need to do http requests to the APIs with needed parameters or request body, get the response data, and present it to the users.

3.2 Back End

We use Java as our back-end development language and SpringBoot Framework. Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications that you can "just run".

We use Maven as our project management and comprehension tool. Dependencies needed are written in the pom.xml files.

You can download the project code, make modifications, and use maven to package it again for deployment.

3.3 Data

We have already provided the MongoDB database in a cloud server. If you want to use your own database, you need to change the URL in the “application.properties”, and Scala files.

```
#mongodb
spring.data.mongodb.uri=mongodb://movieAdmin: [REDACTED]@nus2.com:37018
spring.data.mongodb.database=movierecommender
```

APPENDIX OF REPORT D

Individual Reports

INDIVIDUAL REPORT: GU ZHONGHAN (A0261825A)

(1) personal contribution to group project

In this project I participated in the frontend building job, I design and build the movies' details page and part of the manual bar of the home page using the Baidu amis framework. I also research the recommendation technology and implement the content-based algorithm to recommend similar movies on the movie's details page.

(2) what learnt is most useful for you

Although I used to learn how to build frontend by React, I still find that using Baidu amis is quite pleasant. Amazingly, I could use just JSON to build things like the menu bar, which usually takes me a lot of time to adjust the CSS. I could navigate my previous knowledge of frontend to this project and help my group members to solve problems, this greatly improve my comprehension of the frontend knowledge and my collaboration skill.

I also learned how the content-based recommendation algorithm works. I can use both cosine similarity and tf*idf to calculate the similarity of movies and recommend them to users based on their rating history. Besides, I am now able to use the Spark framework and understand the basic grammar of Scala which is an essential technique when handling big data.

(3) how you can apply the knowledge and skills in other situations or your workplaces

I could use my knowledge of the frontend to help my company to build a beautiful and powerful frontend and call the APIs of the backend expertly. Moreover, although, different companies may use a variety of frameworks, I could comprehend them in a short time and quickly engaged in the development job.

I will also use the knowledge of the recommendation algorithm to help my company to do better recommendations using the description of their products. In addition, since some platform may use this algorithm to do recommendation if my company want to promote their products on such a platform, I could give some advice to my company about how to write a better description of their products.

INDIVIDUAL REPROT: CHEN PENGWEN (A0261632L)

During the project, my main contribution is to select an appropriate algorithm and code to realize the calculating process, and tune the hyper-parameters of the model to make personal recommendations for users. In addition, I also participate in the work to update our database.

The most important knowledge I learnt in this project is the rationale of latent factor, and how it works in a recommendation system. As I learnt from courses, collaborative-filtering algorithms including user-based and item-based have one drawback: they can not handle a sparse rating matrix. After literature research, I find that recommendations based on latent factors can easily solve the problem. Furthermore, I also learnt how to calculate the latent factors using alternating least squares(ALS) method.

Coding is also a vital part. First I managed to learn the codes of experts or professionals. Then I found one that was suitable for our project, and I made some modifications to make it work with our data. The ALS method is contained in the ScalaMLlib, so I just called the class, and set a group of parameters, then used grid search to get the suitable hyper-parameters.

I would like to apply the knowledge of machine learning(about data, model, decision) to deal with classification and regression problems. Also, every algorithm I learnt in courses or the project inspired me about the rules of processing data and the rationale of prediction.

INDIVIDUAL REPORT: TANG YI (A0261851E)

In this project, I am responsible for back-end development, database configuration, and server deployment for the entire project. The backend provides a total of 15 APIs, delivering movie information, user registration and login, recommendation list and so on. I configured the cloud server environment, started the database on the cloud server and deployed the front-end and back-end code, which enables all Internet users to experience our movie recommendation system through our URL, “nus2.com”.

The most important thing I learned is how to build a reliable backend service. If you want to use your algorithm for users, you need a user interface page, so we need to develop a front-end page, and the results calculated by our algorithm are stored in the database, and the information about movies and users should also be stored in the database, so we need back-end services to process user requests from the front-end, get the results from the database, return them to the front-end, and then display them to the user by the front-end. Through this project, I have mastered how to build a reliable backend service with java language and spring framework.

In addition to that, I learned how to deploy frontend, backend and database to the server. The configuration of the server environment is quite complicated. You need to download various dependent software, modify various configuration files, such as implementing a reverse proxy, and finally run the code on the server.

I now understand how a complete set of AI-related systems works, and I also understand how development engineers and algorithm scientists write and develop an AI-related application. So I hope to apply the development knowledge and algorithm knowledge I have learned to my future work to develop or improve a perfect user-friendly AI system

INDIVIDUAL REPORT: WANG JIARUI (A0261860E)

In this project I am mainly responsible for part of the recommendation algorithms and the front end building job. I completed the statistics-based recommendation algorithm at the core of the recommendation system.

In this project, I have gained a lot and progressed. Firstly, I deepened my understanding of recommendation systems through this project. When selecting data sets and models, we clarified the data required by different models and the most applicable scenarios by comparing the differences, advantages, and disadvantages between various models. At the same time, coding from scratch to implement a recommendation algorithm has greatly deepened my mastery of the algorithm principle from the data level. Secondly, the process of data preprocessing gave me more experience in processing data of advanced data types. In this project, the process of searching for data and learning while preprocessing the data set in this project allowed me to learn many methods and techniques that I didn't master before. Finally, from a macro perspective, I also deeply realized the gap between a project's idea and realization. Machine learning, deep learning, and various algorithms require data support. We have many fantastic ideas, many of which are amazing. But in the actual trial, it was found that there was no data source support, which made it difficult to achieve. A novel idea means that there is no support from the data collected by the predecessors, while sufficient data means that the current application has been realized by the predecessors countless times. Innovation will only be reached by a breakthrough in technology.

The knowledge and skills acquired could be applied at my workplace in the future to build reasoning systems that can solve cognitive overload issues faced by my company. The recommendation algorithms learnt could also be applied to solve a different kinds of recommendation problems. In addition, how to use Git to achieve efficient teamwork is another thing I learned from this project.