# MASTER OF TECHNOLOGY

## PROJECT REPORT

Movies Recommendation System

T E A M   M E M B E R S

Mu Yao          A0231527L
Gu XinCheng   A0231435N
Liu ZhaoYang   A0231463M

M A S T E R   O F   T E C H N O L O G Y

# 1. EXECUTIVE SUMMARY

In recent years, with the rapid development of information technology and social networks, the rapid popularization of mobile Internet, the data generated has also increased exponentially. In the vast amount of data and information, it is particularly difficult to filter the information that users are interested in because of different interests of different users. Personalized recommendation systems can make use of big data emerging from mobile Internet and social media to serve users and predict users' preferences more accurately according to their rating data for movies, so as to actively recommend products that users may be interested in.

To implement such a personalized recommendation system, the group creates a database of 26,000,000 ratings and 750,000 tag applications applied to 45,000 movies by 270,000 users. Then we use Python to build the front-end website and back-end recommendation system. Finally, a personalized recommendation-based movie recommendation website is built.

## 1.1 MARKET RESEARCH

For information consumers, the personalized recommendation systems can help users quickly filter out content of interest (e.g., the very popular shopping apps in China, Taobao, JD.com, etc. all make precise recommendations based on users' behaviors and references). It provides reference suggestions when users are confronted with unfamiliar fields (e.g., Dianping offers local business search, user generated reviews, detailed business information, features discounts and other merchant services), and acts as an assistant to users when their needs are unclear, providing them with cutting-edge information in a wider range of fields.

For information producers, the personalized recommendation system brings significant revenue effects. Amazon generates 30% of its annual revenue from personalized recommendations. Since 2008, the recommendation algorithm has added hundreds of thousands of hours of viewing time to YouTube every day and increased the number of video views by 50% each year.

## 2. PROBLEM DESCRIPTION

The application of recommendation technology in the field of film is significantly effective, and as an essential means of entertainment, film is appealing to a growing number of the general public. With the rapid development of the movie industry, various types of movies are increasingly available for the public to choose from. Among the large number of movies, users often do not have enough time to choose to browse and filter the movies they may like one by one due to the wide variety of movies and their long duration (even if they are previews).

Therefore, a personalized recommendation system can help users make decisions more quickly, which not only saves their time but also improves the favorability of information producers' products.

So, how can we learn the types of movies users like without violating their privacy? How to make inferences based on users' favorite movies? To understand and further solve the above and other problems the group first randomly experienced some of the more common video movie recommendation software in the market, and found that most movie recommendations are mainly based on big data, popular lists and other highly rated movies.

As a result, the group decides to build a personalized recommendation system while realizing the popularity-based recommendation.

## 2.1 PROJECT OBJECTIVE

In this project, the group will build a movie recommendation website including front-end website, back-end recommender system. Through the site, users can visually access the recommended movies or search for their favorite movies through the search engine, as well as other movies with similar genres.

The group collects a database of 26,000,000 rating scores and 750,000 tag applications applied to 45,000 movies by 270,000 users. Including tag genome data with 12 million relevance scores on 1,100 tags.

Four recommendation algorithms are used, namely, popularity-based recommender, content-based recommender, collaborative filtering, and hybrid recommender.

## 2.2 BUSINESS CASE

After completing the system, the group recommended our system to our close friends and relatives and invited them to experience the effect of the website. 40 people, including six young people, twenty middle-aged people and fourteen elderly people, were included in the experiment to ensure the sample was evenly distributed. The feedback we received was that more than eighty-five percent of the people thought our website was feasible, had a good experience, and were looking forward to further development and enhancement of the website design.

# 3 RECOMMEND SYSTEM

We are now in an era of information overload. For information consumers, it is very difficult to make the information useful to them from the vast information. Similarly, as an information producer, it is also very difficult to make one's product stand out in the wide sea of information data. Generally, people choose the movies they want to watch through the following three channels:

1. Asking people around them for advice or searching for recent popular and hot movies as reference;

2. People usually have their own favorite actors and directors, and based on this, they search for related movies and videos. The recommendation system can automate the above process by analyzing users' preferences, so as to recommend other movies for them. This approach is called content-based recommendation;

3. People choose movies by looking at the rankings and referring to popular preferences. Based on this, if the system finds a group of users with similar preferences, it can recommend movies that are more attractive to users than the popular rankings. This approach is called collaborative filtering.

## 3.1 SYSTEM ARCHITECTURE

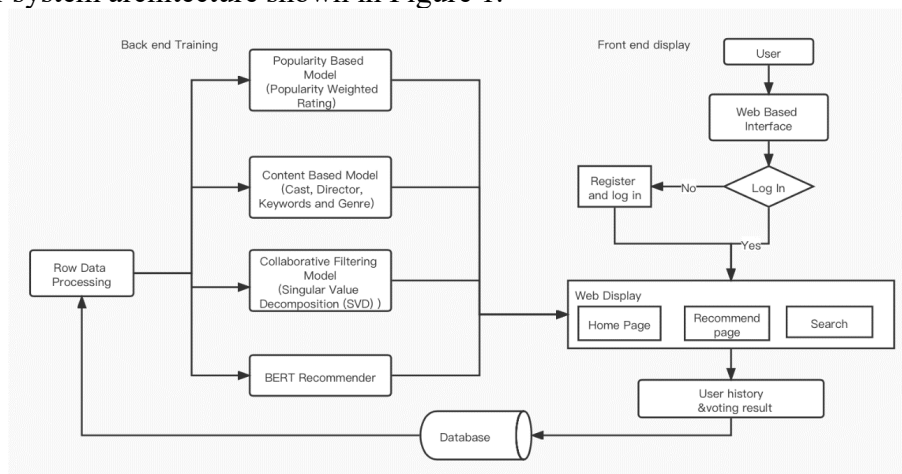The recommender system architecture shown in Figure 1.



Figure 1 System architecture

## 3.2 RECOMMENDER ALGORITHM

### 3.2.1 Popularity Based Recommender

Popularity-based recommender is a recommendation based mainly on the popularity of a movie, sometimes with reference to the movie genre as well. The basic idea is that movies that are more popular and well-received will have a higher probability of being liked by the general audience. However, this model does not give personalized recommendations based on the user's interest preferences.

Therefore, the implementation of this model is very trivial. The next thing the panel needs to do is to sort our collected movie data based on user ratings and movie popularity and show the top movies in the list. As an additional step, the group sets a genre parameter to get the top movies of a specific genre.

The model recommends the best rated movies to users by calculating the score of each movie and ranking the resulting scores.

At the beginning of the project, the group planned to use the average rating of the movies as the score, but through

discussion and research, it was found that some movies were not universally rated due to their high ratings but the small number of users who rated them. For example, a movie with an average rating of 8.9 but only 3 users rated it could not be considered better than a movie with an average rating of 7.8 but 40 users rated it. Therefore, using this method is not fair enough.

To solve this problem, the group used IMDB's weighted rating formula to construct the chart. Mathematically, it is expressed as follows:

$$Weighted\ Rating(WR) = \left(\frac{v}{v+m} \times R\right) + \left(\frac{m}{v+m} \times C\right)$$

Where, $v$ is the number of votes for the movie; $m$ is the minimum number of votes required to be listed in the chart; $R$ is the average rating of the movie; $C$ is the average number of votes for the entire report.

To determine the appropriate value of $m$, the minimum number of votes required is listed in the chart. For a movie to make it to the chart, it must have at least more votes than 85% of the movies in the chart. From this, the group defined a function to create type-specific icons and get the overall Top 100 chart accordingly.

Next, the algorithm yielded that a movie must have at least 270 votes on TMDB to be eligible for consideration for inclusion in the chart. Overall, on a scale of 10 out of 10, a movie with an average rating of 5.764 on TMDB would only qualify a total of 1502 movies for inclusion in our chart. For this reason, we will relax our default condition to the 75th percentile instead of the 85th percentile.

Finally, by displaying the top 10 romance movies in Figure 2 to test our approach, the result shows that despite romance being one of the most popular movie genres, it barely features in our generic top charts.

It is concluded that the algorithm gives the same recommendations to every user, regardless of their personal tastes. If users go a step further and search the charts by genre, they still won't get the best recommendations.

| | title | imdb_id | year | vote_count | vote_average | popularity | genres | wr |
|---|---|---|---|---|---|---|---|---|
| 9336 | Dilwale Dulhania Le Jayenge | tt0112870 | 1995 | 661 | 9 | 34.457024 | [Comedy, Drama, Romance] | 8.061515 |
| 2813 | Fight Club | tt0137523 | 1999 | 9678 | 8 | 63.869599 | [Drama] | 7.939311 |
| 4779 | The Lord of the Rings: The Fellowship of the Ring | tt0120737 | 2001 | 8892 | 8 | 32.070725 | [Adventure, Fantasy, Action] | 7.934105 |
| 291 | Pulp Fiction | tt0110912 | 1994 | 8670 | 8 | 140.950236 | [Thriller, Crime] | 7.932469 |
| 313 | The Shawshank Redemption | tt0111161 | 1994 | 8358 | 8 | 51.645403 | [Drama, Crime] | 7.930027 |
| 6781 | The Lord of the Rings: The Return of the King | tt0167260 | 2003 | 8226 | 8 | 29.324358 | [Adventure, Fantasy, Action] | 7.928940 |
| 350 | Forrest Gump | tt0109830 | 1994 | 8147 | 8 | 48.307194 | [Comedy, Drama, Romance] | 7.928273 |
| 5673 | The Lord of the Rings: The Two Towers | tt0167261 | 2002 | 7641 | 8 | 29.423537 | [Adventure, Fantasy, Action] | 7.923685 |
| 255 | Star Wars | tt0076759 | 1977 | 6778 | 8 | 42.149697 | [Adventure, Action, Science Fiction] | 7.914340 |
| 1214 | Back to the Future | tt0088763 | 1985 | 6239 | 8 | 25.778509 | [Adventure, Comedy, Science Fiction, Family] | 7.907247 |

Figure 2: Top 10 romance movies

### 3.2.2 Content Based Recommender

The algorithm focuses on recommending similar products based on the user's previous preferences. It consists of user attributes and product attributes, the former including the user's inherent attributes and the user's historical product interaction information, and the latter being the description of the product's own attributes, so that recommendations can be achieved by simple cosine similarity. At the same time, this algorithm will have better results for products of the same type with similar description dimensions, and more general results for various types of products.

The core problem to be solved by this approach is whether the recommendation is scalable. If it simply recommends similar products based on the user's previous preferences, the value of the whole recommendation system is very limited, but it would be better if it could accurately recommend other products in different categories.
For the algorithm, the main idea of the group was to calculate the similarity between movies based on certain metrics and to recommend movies that are most similar to a particular movie that the user likes.

After discussion, the group built the content-based recommenders based on movie cast, crew, keywords and genre.

Firstly, we put the cast, crew, genre and credits in one data box. For the crew, we will select only the director as our feature. For the cast the selection is a bit trickier. Lesser-known actors and secondary characters don't really influence

how a film is perceived. Therefore, we had to choose only the main characters and their corresponding actors. The final group decided to arbitrarily select the top three actors that appeared in the credits list.

Next, we created a metadata dump for each movie, which included genre, director, main actors, and keywords. A count vectorizer was then used to create the count matrix. The cosine similarity was then calculated and the most similar movie was returned. Mathematically, it is defined as follows:

$$cosine(x, y) = \frac{xy^T}{||x|| \, ||y||}$$

For the preparation of genre and credit data, we stripped spaces from all features and converted to lowercase. The director is mentioned 2 times to give it more weight relative to the whole cast. For keywords, we first perform a small amount of pre-processing. Frequency counts (ranging from 1 to 610) were calculated for each keyword that appeared in the dataset. For keywords that appear only once, we do not have any use for them. Therefore, these can be safely removed. Finally, we will convert each word into its stem.

Finally, to test the model, we try different weights for features (director, actor, genre), limit the number of keywords that can be used in the soup, weigh genres according to frequency, show only movies in the same language, etc.
The conclusion is that content-based recommendations are only able to recommend movies that are similar to the specified movie genre and cannot provide recommendations across genres. The engine is not truly personal and does not capture the user's personal interests and preferences, and users tend to get similar recommendations when searching.

### 3.2.3 Collaborative Filtering

Collaborative filtering, including both collaborative and filtering operations. The so-called collaboration is to use the group's behavior to recommend. For a recommendation system, through the continuous collaborative actions of users, the recommendations for users will eventually become more and more accurate. Filtering refers to finding the user's preferred option from the feasible recommendation options. Specifically, the idea of collaborative filtering is to find some kind of similarity through the behavior of a group and make decisions and recommendations for users through this similarity.

Collaborative filtering recommendation algorithms are divided into two categories, namely, user-based collaborative filtering algorithms and item-based collaborative filtering algorithms. Since movie production is basically growing steadily, and with the rapid development of the Internet era, more and more users are using movie watching as an entertainment pastime activity, the number of users is growing rapidly. Therefore, in this system, we use a collaborative filtering algorithm based on item.

The item-based CF algorithm mainly calculates the similarity between items, and then generates a recommendation list for users based on the similarity of items and their historical behaviors. However, the item-based CF algorithm does not calculate the similarity directly based on the attributes of the items themselves, but by analyzing the user's behavior to calculate the similarity between items. For example, there are no similarities between cell phones and cell phone cases except for their shapes, so it seems impossible to calculate the similarity directly. But let's consider the problem from another perspective. If there are many users who bought a cell phone and a cell phone case at the same time, can we consider the cell phone and the cell phone case to be more similar?

While, if the item is too popular and the similarity between the item and the popular item is very high, then it may cause the algorithm to always recommend the popular item, and to avoid that problem, mathematically, a formula for calculating the item similarity is as follows:

$$W_{uv} = \frac{|N(u) \cap N(v)|}{\sqrt{|N(u)||N(v)|}}$$

Where, the denominator $|N(u)|$ represents the number of users who like item $u$. The numerator $|N(u) \cap N(v)|$ is the number of users who like both item $u$ and $v$.

We will use the Surprise library to minimize RMSE (root mean square error) and give good recommendations by algorithmic singular value decomposition (SVD).

### 3.2.4 Recommender using self-attention

If we treat each movie as a "word" in language, and the task of recommender system is to predict the next "word" given a sequence of "word" list, so it is similar to the language model. In the area of natural language processing (NLP), there are vary language models, such as LSTM, ElMo, BERT, Transformer, etc. Among these models, BERT is the state-of-art model in NLP, which using the attention mechanism.

Attention was proposed in 2017 by Vaswani, A., et al. It can be calculated in parallel, which is faster than the Recurrent Neural Network. The calculation equation as follow:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d^k}}\right)V$$

where $Q$, $K$ and $V$ represent query, keys and values respectively, $d^k$ is the embedding size.

We designed a BERT-like model, and using masked movie to train this model. The model architecture shown in Figure 3.
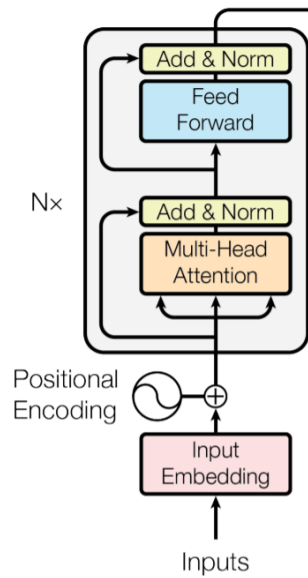


Figure 3. Model architecture

Actually, it is same with BERT, and the positional encoding, we also using the functions proposed by Vaswani, A., et al., shown in follow:

$$PE_{(pos,2i)} = sin\left(pos \ / \ 10000^{2i/d_{model}}\right)$$

$$PE_{(pos,2i+1)} = cos\left(pos \ / \ 10000^{2i/d_{model}}\right)$$

where $pos$ is the position and $i$ is the dimension, $d_{model}$ is the embedding size.

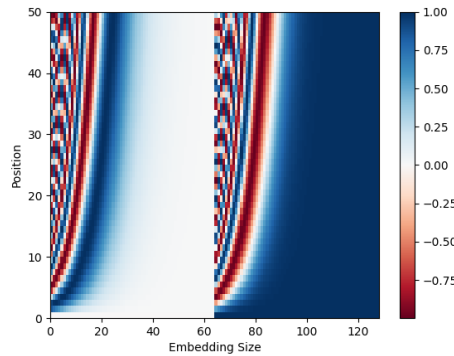We can visualize this positional embedding in Figure4.

Figure 4. Positional embedding

During training, we use the same idea of BERT's Masked Language Model training method, we randomly mask the input sequence movies, and the model need to predict what is the raw movie that been masked. And during inference, we can add a mask token to the end of a sequence, then the last prediction of the model is the recommended movie, and we can use top-N result. Figure 5 shown the training and inference strategy.
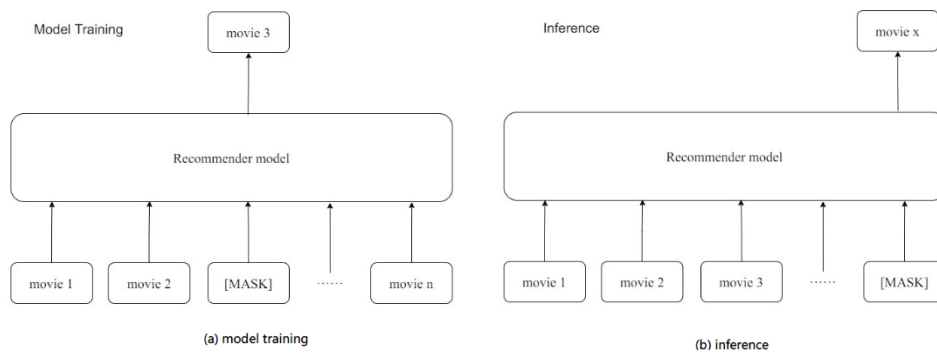


Figure 5. Model training and inference

### 3.2.5 Hybrid Recommender

The hybrid recommendation algorithm promises to make better recommendations for users and improve recommendation quality and user experience by using multiple recommendation algorithms working in concert to avoid the problems of individual algorithms.

The hybrid recommender brings together the techniques we have implemented in our content-based and collaborative filter-based engines.

The user's ID and the title of the movie will be used as input, and the output ranks similar movies based on the expected rating of that particular user.

It can be concluded that for our hybrid recommender, although the movies are the same, we have different recommendations for different users. Thus, our recommendations are more personalized and user specific.

## 4.CONCLUSIONS AND FUTURE WORK

Overall, the group built a total of 4 different recommendation engines for back-end model：

Simple Recommender. By using the overall votes and average votes from TMDB to build general and genre-specific top movie rankings. Weighted scores from IMDB were used to calculate the ratings and finally to rank them.

Content-based recommender. A content-based engine (actors, crew, genre and keywords) was built to make predictions. A simple filter was also designed to give greater priority with more votes and higher ratings.

Collaborative filtering. Single value decomposition using Surprise Library. The RMSE obtained is less than 1 and the engine gives an estimated rating for a specific user and movie.

Recommender using BERT architecture. The deep learning-based method provided an end-to-end method, which do not need to calculate similarity.

Hybrid engine: Combines the ideas of content and collaborative filtering to provide movie suggestions to a particular user based on an estimated rating calculated internally for that user.

## 5. REFERENCES

Resnick, P., & Varian, H. R. (1997). Recommender systems. Communications of the ACM, 40(3), 56-58.

Su, X., & Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. Advances in artificial intelligence, 2009.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W., & Jiang, P. (2019, November). BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In Proceedings of the 28th ACM international conference on information and knowledge management (pp. 1441-1450).

## 6. APPENDICES

## APPENDICES A: PROJECT PROPOSAL

| |
|---|
| **Project Title:** <br><br> Movies Recommendation System |
| **Group ID:** <br><br> No.16 |
| **Group Members:** <br><br> Gu Xincheng    A0231435N <br><br> Liu Zhaoyang   A0231463M <br><br> Mu Yao         A0231527L |
| **Project Objectives:** <br><br> The proposed recommendation system will utilize various recommendation methods and algorithms to form an optimized hybrid system and is designed to be implemented for public users. |

**Project Descriptions:**

**Problem Statement**

1. Information overload

In today's society, with the rapid development of Internet technology, the massive amount of data and information makes information consumers overwhelmed, and it is especially difficult to select the information that interests the user from the vast amount of information.

2. The need for individual expression of information producers

In the vast sea of information data, information producers are eager to make their products stand out, in other words, not to be drowned in the complex and verbose information products.

3. Most users do not have clearer selection criteria

Generally speaking, if the user has a clear demand, can search for information through the search engine directly. Then for users who do not have a clear target, but expect to explore a variety of new areas, personalized recommendations can provide a broader range of cutting-edge information, to bring users a more novel experience.

**Methodology**

The implementation of this project consists of the following methods and combine them to a hybrid system:

1.Popularity-based recommendation

The Popularity based recommender based on movie popularity and (sometimes) genre. The basic idea is that movies that are more popular and more critically acclaimed will have a higher probability of being liked by the average audience. This model does not give personalized recommendations based on the user.

2.Content-based recommendation

It focuses on recommending similar products based on the user's previous preferences. The system consists of user attributes and product attributes, the former including the user's inherent attributes and the user's historical product interaction information, and the latter being the description of the product's own attributes, so that recommendations can be achieved by simple cosine similarity. At the same time, this algorithm will have a better effect for the same type of products with similar description dimensions, and a more common effect for various kind of products.

3. Collaborative filtering

Collaborative filtering, which includes both collaborative and filtering operations. The so-called collaboration is to use the behavior of the group to make decisions (recommendations). For recommendation systems, the recommendations to users will eventually become more and more accurate through the continuous collaborative action of users. Filtering refers to finding (filtering) the user's preferred solution (topic) from the feasible decision (recommendation) solutions (topics).

Specifically, the idea of collaborative filtering is to find some kind of similarity (similarity between users or similarity between topics) through the behavior of a group and make decisions and recommendations for users through this similarity.

4. Deep learning based

We will try to use the idea of BERT, a state-of-art model in NLP, to build a recommender system with self-attention.

5. Hybrid system

Feature combination uses feature data from multiple recommendation algorithms as the original input and uses one of the algorithms as the primary algorithm to generate the final recommendation results.

In the case of collaborative filtering and content-based recommendation, a collaborative filtering algorithm can be used to assign a feature to each sample, and then content-based recommendation uses these features and content-related features to build a content-based recommendation algorithm.

The hybrid approach of feature combination of collaborative filtering and content-based recommendation enables the recommender system to utilize collaborative data without relying on it exclusively, and therefore reduces the sensitivity of the system to the number of users operating on a given topic. Even if there are few user actions for a topic, the topic can be recommended well.

# APPENDIX B: INSTALLATION & USER GUIDE

## 1. Install Python

Go to https://www.python.org/ download the installer. Follow the installation guide. Alternatively, you can use Anaconda to install the python, you can follow this link: https://docs.anaconda.com/anaconda/install/index.html.

This guide will use Anaconda as the tools.

## 2. Install Packages

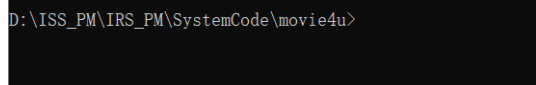If you use the Anaconda, you might want to create a new environment first. Please follow the official guide.

https://docs.anaconda.com/anaconda/user-guide/getting-started/

Change the directory to the source code folder, using

**cd path to source code folder**

or you can open the commands within the source code folder.
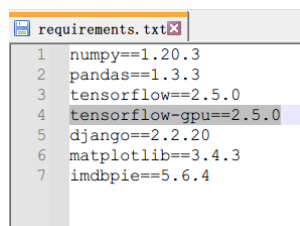
As shown in Fig1.



Fig1 cd source code folder

Then, install the required packages using:

**pip install - r requirements.txt**

If you do not have a GPU on your computer or you do not install CUDA and cuDNN, please delete the line

**tensorflow-gpu==2.5.0**

in requirement.txt, shown in Fig 2.



Fig 2 Delete this line

## 3. Download the pre-trained sim matrix

Please refer to SystemCode\movie4u\data\sim_matrix\README.MD, the download link will be shown there.

## 4. Start the project

In the commands, type:

**python manage.py runserver**

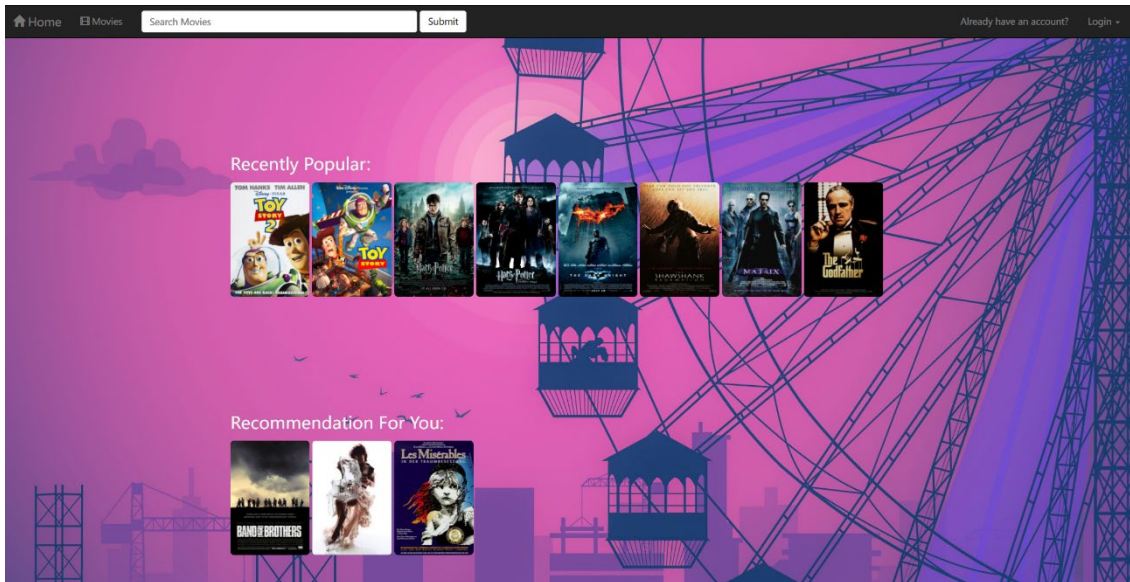Then, open 127.0.0.1:8000 in your browser, you can see the project has been started. Like Fig 3.



Fig 3 Project started

If you got such error shown in Fig 4, please activate your environment first!



Fig 4. Error

## 5. Explore the project

You can modify everything you want to this source code, like using other recommendation methods. And explore it by yourself!

## APPENDIX C: PERSONAL REPORT

## Gu Xincheng:

## Personal contribution to the project

In this project, I am mainly responsible for the design, code writing, model training and recommendation accuracy tuning of the recommendation system in the back-end part of this movie recommendation system.

In the design part, I designed the overall flowchart of the recommendation system back-end model, and each sub-functional module, and I determined the specific implementation method of each functional module.

In the code implementation part, I wrote the code for three sub-systems of the back-end system, namely Popularity Based Recommender, Content Based Recommender, and Collaborative Filtering. Finally, I successfully implemented the three sub-systems in a working manner. Also, I collected high quality datasets from the Internet for training and accuracy optimization.

## Learnt from the project

First, in the theoretical part, I learned the complete steps of building a model, as well as the principles and specific implementation methods of recommendation systems, such as Content-Based Recommenders, Collaborative Filtering (Singular Value Decomposition (SVD)), and how to implement the evaluation of recommendation systems.

Second, in the hands-on part, I became more familiar with how to use Python's powerful library to accomplish personal goals. For example, I have become more familiar with using the panda library to process static data, using the Surprise library to implement Collaborative Filtering, using Python's Django framework architecture to build web pages, etc.,

By writing code, my programming ability has been improved, which will help me a lot in the future.

## How to apply this in future work-related projects

In practice, this project experience helped me to be familiar with the whole process of knowledge modeling and could understand the process of a project more comprehensively than just one part of it. Of course, becoming more familiar with the programming code and algorithms will help the project to be completed.

More than that, most of the future projects are not individual tasks but group tasks, so learning to get along with group members is a very important skill. Through this group work, I learned better how to grind with group members with different personalities to complete a project together. I think this is very important for future work.

**Mu Yao:**

## Personal contribution to the project

In this project, I was mainly responsible for collecting and processing data in the pre-project stage. During the project, I assisted the team members in designing and training the model, building the front-end website, and writing the final report.

## Learnt from the project

Through this project, I have honed my ability to write code and accumulated programming experience. In the process, I gradually became familiar with various libraries in Python, and learned the recommendation system framework, such as Surprise, which comes with SVD, user-based, item-based collaborative filtering algorithms and other recommendation algorithms with simple and powerful interfaces.

I also learned how to use Django to create web pages and studied how to implement the front and back-end build of a complete web page. At the same time, I learned a lot about other scholars' research in the field of recommendation systems in the process of writing the report.

More importantly, in this project, I learned to cooperate better with team members, communicate effectively, improve together, overcome difficulties and achieve the final goal.

## How to apply this in future work-related projects

The experience of this project has taught me to apply the theoretical knowledge I have learned more flexibly in practice. It helped me to be familiar with the whole process of knowledge modeling, to build a more complete knowledge system, and to systematically understand what I learned in the course.

At the same time, I accumulated experience in communication and cooperation with people, and learned more guidelines for dealing with people, which will help me to complete my work more efficiently in various group projects and work scenarios in the future, and get along better with people of different personalities.

**Liu Zhaoyang:**

## Personal contribution to the project

I build the BERT recommender model, and train the model with different hyper parameters. Preprocess the dataset that can be used for bert recommender model training. Merge the recommendation methods to build a hybrid recommender system. Finish some parts of final report and user guide document.

## Learnt from the project

The basic idea of start a project. Usually, we do not need build everything from scratch, we need to find some references, do some search in recent works which related to our project. With these, we can quickly build a small system to test whether the idea of the project is feasible or not before it is too late. If we do not do this, it might bee too late when we find something of our project is wrong. At beginning, we just start our project without any research, and certainly, we wasted 2 weeks. But after that, we did research, did some small tests, and everything went well.

The knowledge of IRS course. In class, we learnt many methods, but just understand the principle of the methods, when it comes to the real-world problems, I find it is difficult to directly apply the methods we learnt from class, we need to make some changes to the method to adapt the real-world problems. And in the practice, we can fully understand these methods and apply these methods to real problems.

Coding. During this project, I have learnt a lot of tricks in model training and coding, and have tried many different API when coding. For example, in the past I only use the sequential and functional API of keras, but in order to build the BERT-like recommender model, I need use more powerful and flexible APIs, namely the model subclassing. By coding and debugging, I think I have familiar with the model subclassing.

Teamwork. This is a team project, only one person cannot finish it, during the whole progress, I have learnt the power of teamwork. In a team, if you do not know how to solve a problem, the others may know, and with communications, you can get a better understanding of how to solve such problem than just search on the Internet, I think this is the biggest benefit of teamwork, you can learn more than you do it individually. And during a team, everyone is working hard to achieve the final goal, and this atmosphere will also push you forward.

## How to apply this in future work-related projects

In the future, when I start a project, I need to consider the feasibility at very beginning, as I have already suffered a loss. Besides, I will communicate with the team mates frequently since sometimes others' ideas may be helpful for the project, which will be useful for project start.

During work, I can apply what I have learned in IRS, to solve the real problems. The knowledge can not only apply on the problem itself, but also can apply on anywhere that is suitable.

The coding ability is not that important, but it is necessary, so I will continually improve the coding ability and use this to make the project progress faster.

Make a good use of the power of team, more communication means more efficiency, thus listen to others' opinion more and ask more when I facing a problem.