

## 摘要

本系统采用了 django 框架，分为前台和后台两个模块，用户在前台可以查看不同类型的电影的详细信息，也可以通过搜索功能寻找自己想了解的 movie。用户在前台可以查看评分最高的十部电影，也可以对电影打分。管理员可以在后台完成对电影，评分和用户数据的增删改查。当用户需要查看的电影没有在数据库中时，本系统会启动爬虫对豆瓣电影网站上的数据进行抓取并保存到数据库中。当用户对自己看过的电影进行评分后，本系统会收集用户的评分记录，然后根据协同过滤算法来计算并选择与用户相似度较高的电影推荐给用户。使用协同过滤算法可以根据历史用户的记录来为用户推荐与用户兴趣契合度高的冷门电影，这样的系统推荐的电影的命中率更高。电影推荐系统可以节约用户时间，增强用户体验。

**关键词：**推荐算法 爬虫 django 框架

## Abstract

The system adopts Django framework, which is divided into two modules: foreground and background. Users can view the details of different types of movies in the foreground, or search for the movies they want to know through the search function. Users can view the top ten movies in the front desk and also rate them. The administrator can add, delete, and check the movie, rating, and user data in the background. When the movie users need to view is not in the database, the system will start the crawler to grab the data on Douban movie website and save it in the database. When users rate the movies they have seen, the system will collect the user's score records, and then calculate and select the movies with high similarity to users according to the collaborative filtering algorithm. By using collaborative filtering algorithm, we can recommend the popular movies that match the users' interests according to the records of historical users, and the hit rate of the movies recommended by this system is higher. Movie recommendation system can save user time and enhance user experience.

**Keywords:** recommended algorithm web spider django

# 目录

摘要.....	I
Abstract.....	I
1 前言.....	1
1.1 项目背景.....	1
1.2 设计目的和意义.....	1
2 系统分析.....	2
2.1 需求分析.....	2
2.2 可行性分析.....	2
2.3 系统用例分析.....	2
3 项目设计.....	4
3.1 设计模式.....	4
3.2 系统结构.....	4
3.2.2 前台管理模块.....	4
3.2.3 后台管理模块.....	5
3.3 系统流程设计.....	6
3.3.1 用户登录.....	6
3.3.2 电影搜索.....	6
3.3.3 获取电影推荐.....	7
3.4 数据库设计.....	8
3.5 电影推荐算法介绍.....	10
3.5.1 基于物品的协同过滤算法.....	10
3.5.2 算法流程.....	10
3.5.3 推荐算法评测.....	13
4 项目实施.....	14
4.1 系统前台模块.....	14
4.1.1 用户登录.....	14
4.1.2 用户注册.....	15
4.1.3 电影类型.....	15
4.1.4 电影搜索.....	16
4.1.5 电影详情.....	16
4.1.6 电影评分.....	17
4.1.7 电影推荐页面.....	17
4.2 系统后台模块.....	19
4.2.1 密码修改.....	19
4.2.2 历史记录.....	19
4.2.3 管理员添加.....	20
4.2.4 电影信息管理.....	20
4.2.5 评分信息管理.....	21
4.2.6 用户信息管理.....	21
5 总结.....	22
5.1 项目总结.....	22
5.2 个人收获与体会.....	23

参考文献.....	24
致谢.....	错误!未定义书签。

# 1 前言

## 1.1 项目背景

当今世界经济高速发展，社会空前进步，人们的精神生活也越发的丰富多彩起来，许许多多的年轻人喜欢通过看电影的方式来消磨自己的时间，开阔自己的眼界<sup>[1]</sup>。与此同时随着世界经济的繁荣发展，现有的电影的数量也越来越多。在信息爆炸的今天，获取信息的途径和方式多种多样，人们花费时间最多的不再是去哪获取信息，而是要在众多的信息中寻找自己感兴趣的信息，这就是信息超载问题<sup>[2]</sup>。为那么如何从浩如烟海的影片库中找到适合自己的电影成为了一个令人头疼的问题，为此推荐系统应运而生。

个性化推荐算法可以使用到各种的网站上，好的推荐算法不仅可以节约用户的时间<sup>[3]</sup>，而且当用户使用推荐系统的时间越来越来长，用户对系统的依赖度和忠诚度就会更高，从而可以为网站带来更高的收益<sup>[4]</sup>。

## 1.2 设计目的和意义

基于协同过滤的电影推荐系统可以说是根据用户的喜好和历史行为来为用户进行推荐的，这种推荐是交给系统自行完成的不需要用户提供明确的要求，即用户获得的推荐是系统从购买模式或浏览行为等隐式获得的<sup>[5]</sup>，不需要用户努力地找到适合自己兴趣的推荐信息，但是因为许多协同过滤算法或多或少都存在冷启动问题，在用户刚完成注册时无法为其推荐合适的电影。本系统通过对电影的评分进行排序，向用户推荐最热门的排行榜单。因此基于协同过滤算法的电影推荐系统可以通过分析用户的历史行为来提高用户满意度，增加用户黏性。具有一定的开发价值。

## 2 系统分析

### 2.1 需求分析

通过对国内各大视频网站和电影推荐网站的考察，本系统应该具备以下的功能：

- (1) 页面简洁明了，用户可以方便的浏览电影信息；
- (2) 系统可以展示数据库中电影的信息，使得用户可以了解这些电影的基本信息；
- (3) 具有电影的分类，用户可以搜索特定类型的电影；
- (4) 可以为电影评分，并记录用户的历史行为；
- (5) 可以根据用户对不同电影的评分为用户推荐相似度高的电影；
- (6) 具有评分最高的电影的推荐功能；
- (7) 管理员可以在系统后台对用户信息和电影信息进行管理。

### 2.2 可行性分析

本电影推荐系统的可行性分析如下：

- (1) 技术可行性：基于协同过滤算法的电影推荐系统使用 django 框架，开发软件有 pycharm 和 Mysql 数据库，不需要特殊的硬件。一般的电脑配置即可保证系统的开发和流畅运行。
- (2) 经济可行性：电影推荐系统可以参考国内成熟的视频网站，并且电影的推荐算法的实现难度不是特别的，使用的 pycharm 和 mysql 工具花销不大。系统所包含的数据量不是特别大，所以开发系统的成本可控。
- (3) 操作可行性：电影推荐系统界面设计友好，功能通俗易懂，用户易于上手。系统对运行环境的要求一般，平常的家用电脑就可以对系统进行流畅的操作。
- (4) 法律可行性：本项目不侵犯用户的私人利益，用于系统开发的软件都是免费版，不触犯法律。

### 2.3 系统用例分析

系统用例分析主要分析系统的参与者、用例、以及参与者与用例之间的关系<sup>[6]</sup>。电影推荐系统有两种参与者，分别为用户与管理员。如图 2-1 和图 2-2 所示。

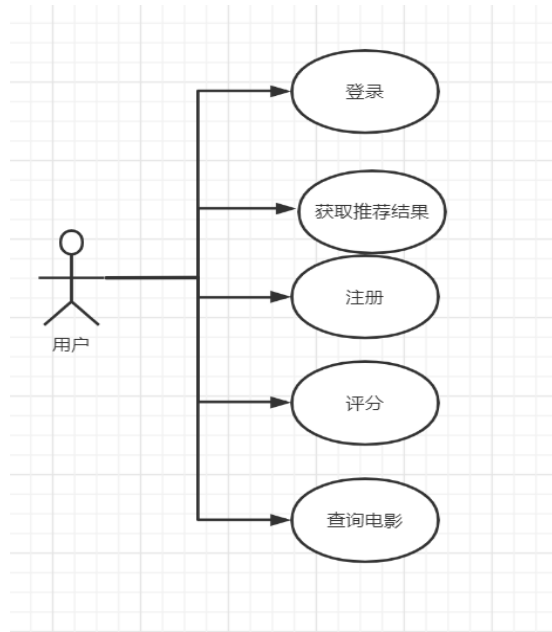


图 2-1 用户用例图

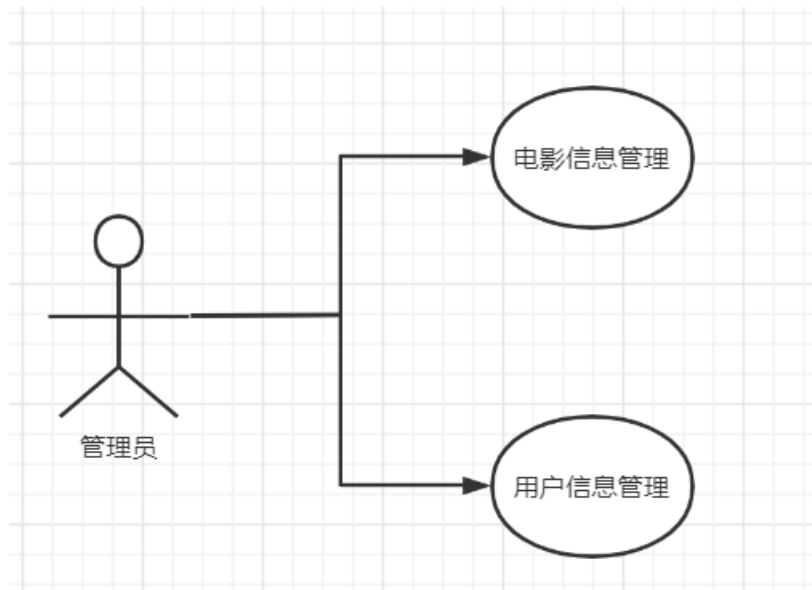


图 2-2 管理员用例图

## 3 项目设计

### 3.1 设计模式

本项目使用 MTV 设计模式，如图 3-1 所示。

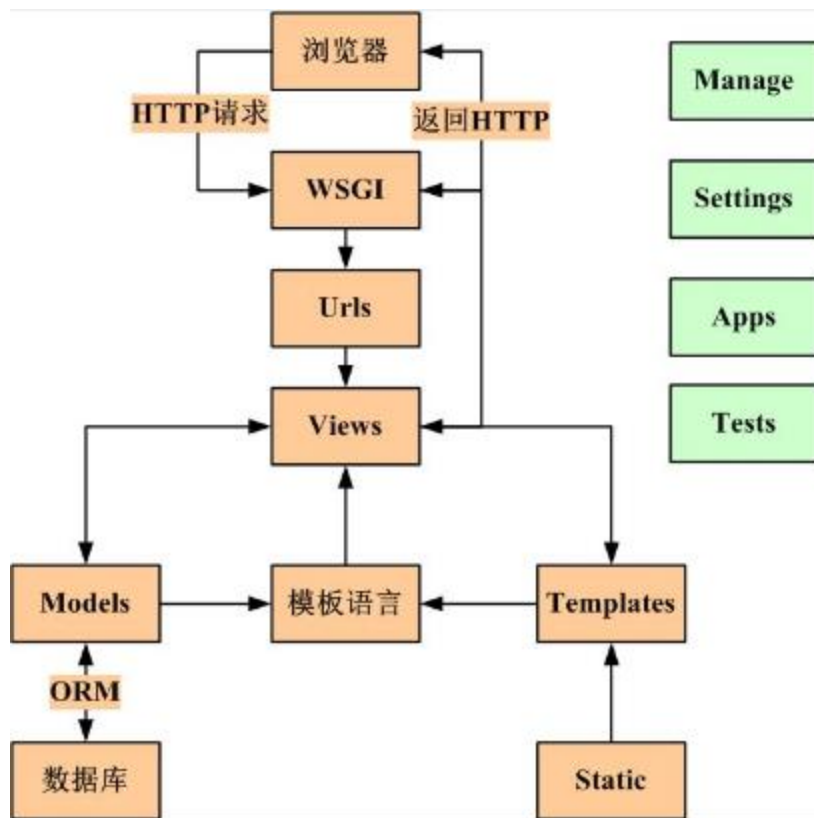


图 3-1 MTV 设计模式示意图

### 3.2 系统结构

电影推荐系统包括前端页面和后端页面两大页面，前端页面为用户提供电影查找和电影推荐服务，管理员可以在系统后端对用户和电影的信息进行管理。具体模块如下：

#### 3.2.2 前台管理模块

- (1) 电影展示模块：显示数据库中包含的所有的电影的海报和名称；
- (2) 电影分类模块：用户可以根据自己的喜好来选择不同类型的电影；
- (3) 高分电影模块：向用户推荐数据库中得分最高的前十部电影；
- (4) 用户信息模块：用户可以在用户信息的页面查看自己点评过的电影以及系统推荐给用户的电影；
- (5) 电影搜索模块：用户可以通过搜索框来搜索自己想查看的电影，如果数据库中存在用户搜索的电影则返回数据库中的内容，如果数据库中不存在用户的搜索内容，则启动爬虫来抓取豆瓣电影里的搜索结果并显示到搜索页面上。

(6) 电影详情模块：用户点击电影海报后系统搜索该电影在数据库中的信息是否完善，如果完善就将详情返回到电影信息展示页面，如果信息不完善，则启动爬虫抓取豆瓣电影里的具体信息并显示到详情页面上。  
系统前端页面模块如图 3-2 所示。

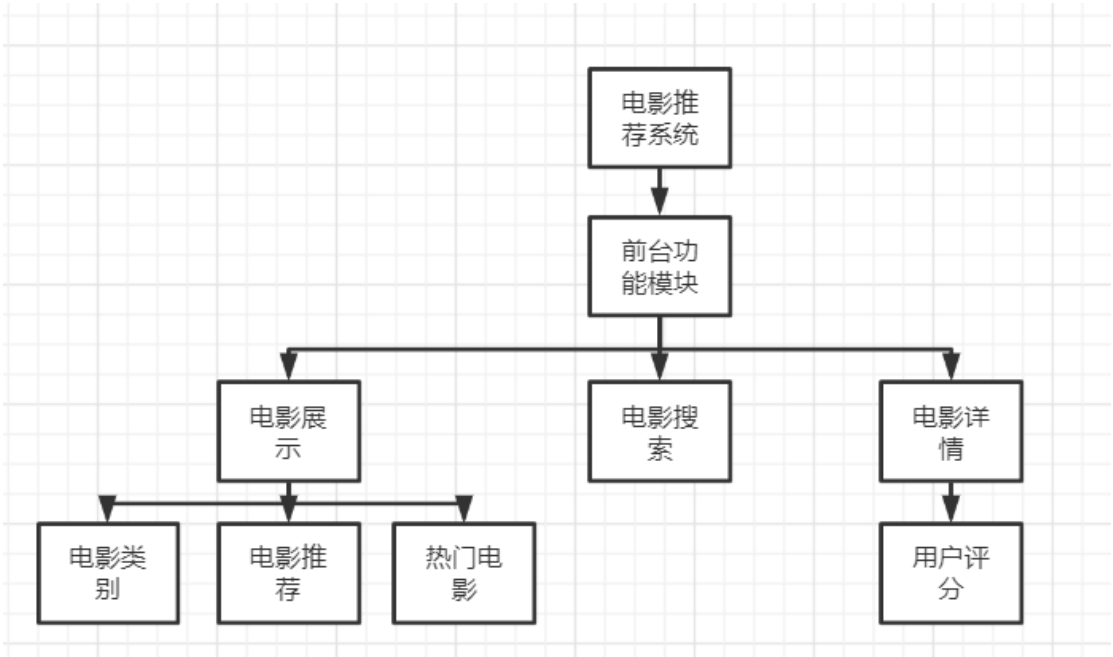


图 3-2 前端页面结构图

3.2.3 后台管理模块

(1) 用户管理模块：管理员能够对使用该推荐系统的所有用户的账号密码和评论记录进行管理。

(2) 电影管理模块：管理员能够对电影的标题、类型、演员等信息进行操作和管理。

管理员后台如图 3-3 所示。

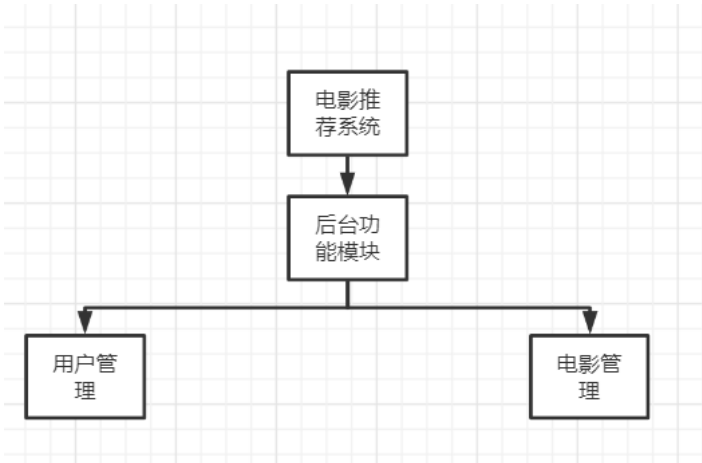


图 3-3 后台管理系统模块



### 3.3 系统流程设计

#### 3.3.1 用户登录

用户需要输入必要的信息进行网站的登录，后台对用户输入的信息进行校验，校验通过后跳转到电影展示页面，否则提示重新输入，用户登陆如图 3-4 所示。

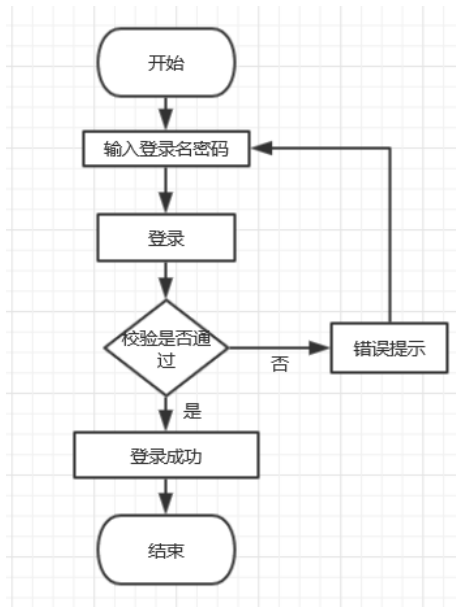


图 3-4 登陆流程

#### 3.3.2 电影搜索

用户在电影展示页面并没有发现自己喜欢的电影，这时可以通过电影的搜索功能来寻找自己想要寻找的电影，如图 3-5 所示。

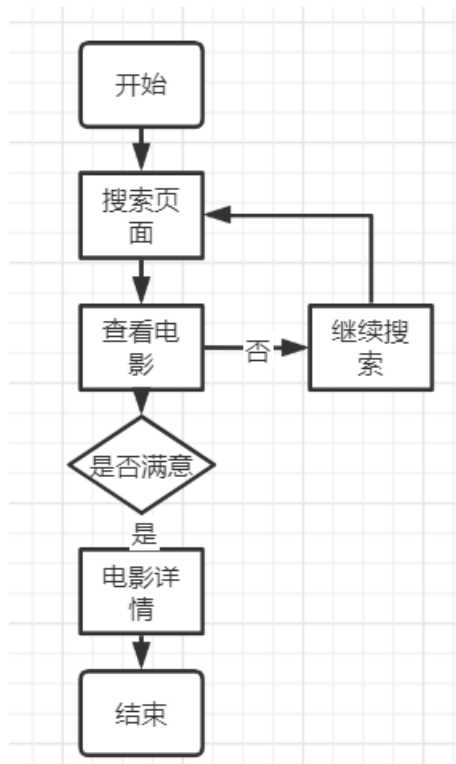


图 3-5 电影搜索流程图

### 3.3.3 获取电影推荐

用户在给部分电影打过分后系统会根据用户的历史纪录来为用户推荐最适合用户的电影，如图 3-6 所示。

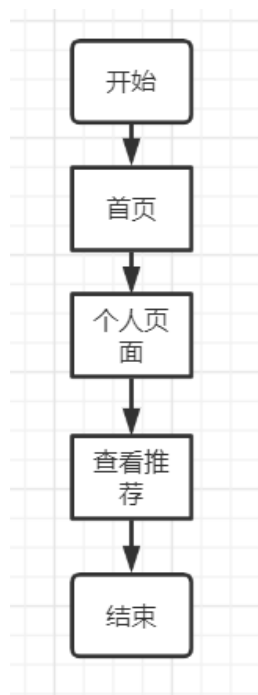


图 3-6 获取电影推荐流程图

### 3.4 数据库设计

如今系统开发常用的数据库有：MYSQL、Oracle、MongoDB 等。电影推荐系统使用 MySQL 数据库，MySQL 可以满足项目开发的种种需要。

开发电影推荐系统的目的是通过对用户的历史行为进行分析，然后寻找与用户相似度最高的电影并推荐给用户，通过分析可以得知此电影推荐系统的数据项应该具有以下要求：

- (1) 用户登录、注册需要建立用户信息表，包含用户账号名称、密码等；
- (2) 电影的信息需要建立电影信息表来存储，包含电影名称、电影类型、演员、编剧等等；
- (3) 系统需要记录用户给电影的打分并以此来为用户推荐电影，所以需要建立评分表，包含用户的个人 id、被用户评论过的电影的 id 以及用户给电影的打分等等。

综上所述，应建立如下数据表和数据项：

- (1) 用户表，包含用户 id、用户年龄、用户的账号、用户的昵称、用户的密码等数据项；
- (2) 电影信息表，包括电影名称、发行时间、上映地区等数据项；
- (3) 评分信息表，包含电影 id、用户 id、用户评分等数据项；

用户表记录用户的登录信息，用户用这个表中的信息来完成登录，推荐算法也根据表中的用户 id 来进行电影推荐，如表 3-1 所示。

表 3-1 用户表 (user)

字段名	说明	类型	长度	可否为空	主键
id	用户编号	Int	11	不可	是
gender	用户性别	Varchar	1	可	否
age	用户年龄	Int	50	可	否
work	用户工作	Varchar	50	可	否
username	用户账号	Varchar	16	不可	否
nickname	用户名称	Varchar	16	可	否
password	用户密码	Varchar	20	不可	否

电影信息表用于记录所有电影的信息，在电影推荐系统的电影展示页面和电影详情页面会用到此表中记录的信息，如表 3-2 所示：

表 3-2 电影信息表(movie)

字段名	说明	类型	长度	可否为空	主键
id	电影编号	Int	11	不可	是
title	电影名称	longtext	0	可	否
genres	电影类别	longtext	0	可	否
imbd	电影 imbd 编号	longtext	0	可	否
url	电影海报链接	longtext	0	可	否
actor	电影演员名称	longtext	0	可	否
aditor	电影编剧名称	longtext	0	可	否
ares	电影上映地区	longtext	0	可	否
douban	电影豆瓣编号	longtext	0	可	否
douban_rating	电影豆瓣评分	double	0	可	否
imbd_rating	电影 imbd 评分	double	0	可	否
rename	电影别名	longtext	0	可	否
runtime	电影时长	longtext	0	可	否
summary	电影梗概	longtext	0	可	否
releasedate	电影上映日期	longtext	0	可	否

评分信息表记录不同用户对不同电影的打分信息,为电影推荐算法提供用户数据,如表 3-3 所示。

表 3-3 鞋子类别表 (rating)

字段名	说明	类型	长度	可否为空	主键
id	评分编号	Int	11	不可	是
rating	电影分数	double	0	可	否
movies_id	电影编号	Int	11	可	否
user_id	用户编号	Int	11	可	否
datetime	评分时间	date	0	可	否

根据电影推荐系统功能和用户需求,得到如图 3-7 所示的系统 E-R 图。

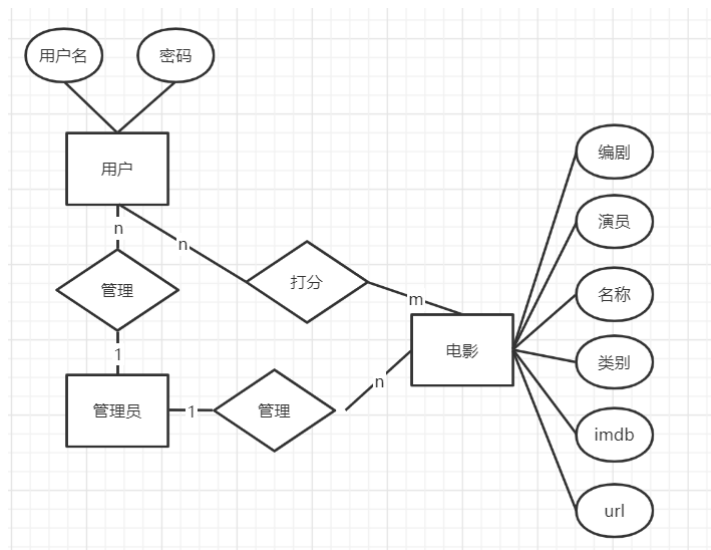


图 3-7 系统 E-R 图

### 3.5 电影推荐算法介绍

#### 3.5.1 基于物品的协同过滤算法

基于物品的协同过滤算法通过收集到的用户历史行为来为用户推荐与用户兴趣最相似的电影。因为每个用户都独立操作，拥有自己的历史行为记录，不需要考虑别的用户的历史行为，所以当系统为一个新的用户推荐电影时会因为用户没有历史记录而造成推荐不准确的问题发生。因此，通常系统会为用户提供用户喜好标签，这样可以在很大程度上缓解冷启动带来的问题。

此算法认为，如果喜欢电影 A 的用户大都喜欢电影 B，那么则说明电影 A 和电影 B 之间具有很大的相似度。基于物品的协同过滤算法需要计算电影间的相似度，然后根据电影的相似度和用户的历史行为给用户推荐电影<sup>[7]</sup>。

#### 3.5.2 基于用户的协同过滤算法

基于用户的协同过滤算法推荐那些和目标用户有相似兴趣爱好的用户喜欢的物品。假设用户 A 喜欢物品 A、物品 C，用户 B 喜欢物品 B，用户 C 喜欢物品 A、物品 C 和物品 D；从这些用户的历史喜好信息中，我们可以发现用户 A 和用户 C 的口味和偏好是比较类似的，同时用户 C 还喜欢物品 D，那么我们可以推断用户 A 可能也喜欢物品 D，因此可以将物品 D 推荐给用户 A。

#### 3.5.3 算法流程

由于两个算法原理相似，所以这里只介绍基于物品的协同过滤算法：先构建用户-电影的倒排，然后构建电影与电影的同现矩阵，再计算电影间的相似度，最后根据用户的历史行为，给用户推荐电影。

(1) 构建用户-电影倒排表

行表示电影，列表示用户，1 表示用户喜欢该电影，如表 3-4 所示。

表 3-4 用户-电影倒排表

用户\电影	a	b	c	d	e
A	1	1		1	
B		1	1		1
C			1	1	
D		1	1	1	
E	1			1	

(2) 构建电影与电影的同现矩阵

矩阵中的数字表示同时喜欢这两个电影的人数，如表 3-5 所示

表 3-5 同现矩阵

电影\电影	a	b	c	d	e
a		1		2	
b	1		2	2	1
c		2		2	1
d	2	2	2		
e		1	1		

(3) 计算电影间的相似度

一开始用来计算相似度的公式如式 (3-1) 所示。

$$w_{ij} = \frac{|N(i) \cap N(j)|}{|N(i)|} \quad (3-1)$$

式 (3-1) 虽然可以计算电影间的相似度，但是存在一个问题，当电影 j 有许多人观看过并且对其进行了评分的时候，有很大的概率会出现观看过电影 i 的人全都看过电影 j，这时计算出的相似度会接近与 1。这样的算法会让很多的电影都与当前的热门电影有接近 1 的相似度，极大的影响了推荐的准确性，所以改进后的公式如式 (3-2) 所示。

$$w_{ij} = \frac{|N(i) \cap N(j)|}{\sqrt{|N(i)| \cdot |N(j)|}} \quad (3-2)$$

式 (3-2) 中的  $|N(i)|$  表示点评过电影 i 的用户数， $|N(i) \cap N(j)|$  表示同时点评了电影 i，j 的用户数，矩阵 N 表示点评过某电影的用户总数，式 (3-2) 避免了热门电影与很多电影都有较大相似度的情况发生。

矩阵 N 和电影间的相似矩阵如表 3-6，3-7 所示。

表 3-6 矩阵 N

电影	a	b	c	d	e
用户数	2	3	3	4	1

表 3-7 相似矩阵

电影\电影	a	b	c	d	e
a		0.41		0.71	
b	0.41		0.67	0.58	0.58
c		0.67		0.58	0.58
d	0.71	0.58	0.58		
e		0.58	0.58		

(4) 兴趣度计算

根据用户历史行为，给用户推荐物品，如图 3-8 所示。

**物品j预测兴趣度=用户喜欢的物品i的兴趣度×物品i和物品j的相似度**

图 3-8 预测兴趣度公式

### 3.5.4 推荐算法评测

推荐系统为用户生成推荐列表，系统推荐的预测准确率是通过现实准确率和召回率来评测的。

计算推荐结果的准确率的公式如式(3-3)所示：

$$\text{Precision} = \frac{\sum_{u \in U} |R(u) \cap T(u)|}{\sum_{u \in U} |R(u)|} \quad (3-3)$$

计算推荐结果的召回率的公式如式(3-4)：

$$\text{Recall} = \frac{\sum_{u \in U} |R(u) \cap T(u)|}{\sum_{u \in U} |T(u)|} \quad (3-4)$$

上述式子中的  $R(u)$  是推荐算法根据训练集给出推荐结果， $T(u)$  是用户测试集中用户的评分情况。

将数据集分为训练集和测试集，训练集和测试集分别占数据集总量的 75% 和 25%。分别取推荐 5，10 个电影来计算 ItemCF 算法和 UserCF 算法的准确率和召回率。得到的结果如表 3-8 所示。

表 3-8 ItemCF 算法和 UserCF 算法的比较

推荐电影数	ItemCF 算法		UserCF 算法	
	准确率	召回率	准确率	召回率
5	0.3387	0.0407	0.3888	0.0469
10	0.2905	0.0702	0.3434	0.0828

由于该系统使用的数据集中电影的数量远多于用户的数量，所以看起来基于物品的协同过滤算法准确率低一些。



# 4 项目实现

## 4.1 系统前台模块

电影推荐系统的前端页面有四个组成部分，头部是一个导航栏可以登录和查找电影，左侧是电影的所有类别，用户可以根据这个列表选择自己希望看到的电影类型，中间是电影的展示区，用户可以查看电影的海报和名称，点击自己想要查看的电影的海报或名称后可以进入电影详情页面，右侧是评分最高的电影列表，列表中包含 imdb 中评分前十的电影，为新用户提供了选择。首页页面如图 4-1 所示。

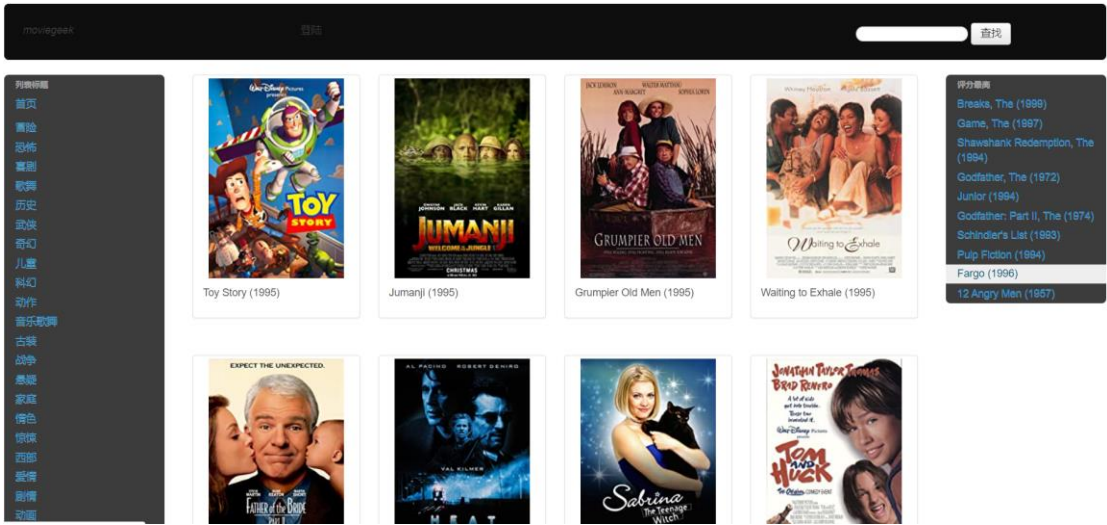


图 4-1 首页页面

### 4.1.1 用户登录

用户可点击页面上方的导航栏中的登录按钮进行登录操作，如图 4-2 所示。



图 4-2 用户登录界面

### 4.1.2 用户注册

用户点击注册后，需要输入用户名和密码，用户名是唯一的，系统会判断用户输入的用户名是否已经存在，如果已经存在需要输入新的用户名。如图 4-3 所示。



The image shows a user registration form titled "用户注册" (User Registration). It contains two input fields: "请输入用户名" (Please enter username) and "请输入密码" (Please enter password). Below these fields is a blue button labeled "立即注册" (Register Now).

图 4-3 用户注册

### 4.1.3 电影类型

用户选择首页页面左侧的列表中的电影类型后，可以筛选出选定类型的电影集合并返回到页面中供用户完成选择。

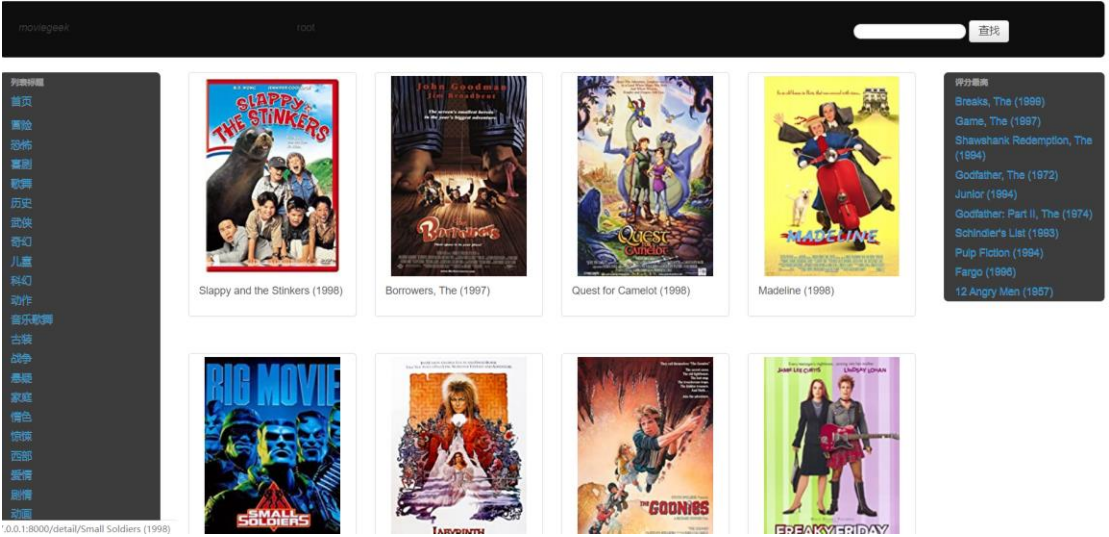


图 4-4 电影类型

4.1.4 电影搜索

用户可以自行搜索自己想要找的电影，可以输入电影的 imdb 号、豆瓣号或电影名称关键字来进行搜索<sup>[8]</sup>。如图 4-5，4-6 所示。



图 4-5 电影搜索

	蜘蛛侠：英雄远征 Spider-Man: Far from Home (2019) 7.7 美国 / 动作 / 科幻 / 冒险 / 蜘蛛侠：决战千里(港) / 蜘蛛人：离家日(台) / 127分钟 乔·沃茨 / 汤姆·赫兰德 / 赞达亚 / 杰克·吉伦哈尔 / 寇碧·史莫德斯 / 塞缪尔·杰克逊 / 乔恩·费儒 / 玛丽莎·托梅 / 雅各布·巴特朗
	蜘蛛侠：平行宇宙 Spider-Man: Into the Spider-Verse (2018) 8.6 美国 / 动作 / 科幻 / 动画 / 冒险 / 蜘蛛侠：新纪元 / 蜘蛛人：新宇宙(台) / 116分钟 鲍勃·佩尔西凯蒂 / 彼得·拉姆齐 / 罗德尼·罗斯曼 / 沙梅克·摩尔 / 杰克·约翰逊 / 海莉·斯坦菲尔德 / 马赫沙拉·阿里 / 布莱恩·泰里·亨利 / 莉莉·汤姆林 / 劳伦·维勒斯 / 佐伊·克罗维兹
	蜘蛛侠：英雄归来 Spider-Man: Homecoming (2017) 7.4 美国 / 动作 / 科幻 / 冒险 / 蜘蛛侠：强势回归(港) / 蜘蛛人：返校日(台) / 133分钟 乔·沃茨 / 汤姆·赫兰德 / 小罗伯特·唐尼 / 玛丽莎·托梅 / 迈克尔·基顿 / 雅各布·巴特朗 / 托比·凯利 / 赞达亚 / 乔恩·费儒
	蜘蛛侠 Spider-Man (2002) 7.8 美国 / 动作 / 科幻 / 冒险 / 蜘蛛侠 / 蜘蛛人 / 121分钟 山姆·雷米 / 托比·马奎尔 / 威廉·达福 / 克斯汀·邓斯特 / 詹姆斯·弗兰科 / 克里夫·罗伯逊 / 罗斯玛丽·哈里斯 / J·K·西蒙斯 / 乔·曼根尼罗
	蜘蛛侠3 Spider-Man 3 (2007) 7.4 美国 / 动作 / 科幻 / 冒险 / 蜘蛛侠3 / 139分钟 山姆·雷米 / 托比·马奎尔 / 克斯汀·邓斯特 / 詹姆斯·弗兰科 / 托马斯·哈登 / 托弗·戈瑞斯 / 布莱丝·达拉斯·霍华德 / 罗斯玛丽·哈里斯 / J·K·西蒙斯
	蜘蛛侠2 Spider-Man 2 (2004) 7.6 美国 / 动作 / 科幻 / 冒险 / 蜘蛛侠2 / 蜘蛛侠2 / 127分钟 山姆·雷米 / 托比·马奎尔 / 克斯汀·邓斯特 / 詹姆斯·弗兰科 / 阿尔弗雷德·莫里纳 / 罗斯玛丽·哈里斯 / J·K·西蒙斯

图 4-6 电影搜索列表

4.1.5 电影详情

用户在选择好自己想查看的电影后点击海报进入电影详情页面，如果数据库中保存的有该电影的完整信息，那么直接在详情页面显示电影的基本信息，如果数据库中该电影的信息不完整，那么会启动爬虫爬取豆瓣电影中该电影的详细信息并存储到数据库中，然后显示到页面上。如果数据库中同时有豆瓣电影的评分和 imdb 的评分，那么在详情页面上两个评分都显示，否则数据库中存在哪一个评分就显示哪一个评分。如图 4-7 所示。



图 4-7 电影详情

#### 4.1.6 电影评分

用户根据自己对电影的喜好程度对电影打分，如图 4-8 所示。



图 4-8 电影打分

点击重置按钮后可以对电影重新打分，如图 4-9 所示。




图 4-9 重新打分

#### 4.1.7 电影推荐页面


用户在个人页面可以看到自己所有的打分记录，并获得系统推荐给用户的电影，系统根据用户过往的电影评分来为用户推荐电影，用户对自己看过的电影进行评分后即可进入电影推荐页面查看推荐结果。系统使用 ItemCF 算法进行推荐，

在电影推荐系统中长尾物品丰富，用户的个性化需求强烈，而 ItemCF 算法正好契合这一特点。当用户对新的电影进行评价过后会导致推荐结果的实时变化，并且因此推荐算法是根据用户的历史行为来为用户做推荐，所以其推荐结果更容易使用户信服。操作流程如图 4-10，4-11，4-12 所示。


编号	标题	imdb号	地区
1	Toy Story (1995)	0114709	美国
2	Jumanji (1995)	2283362	美国
3885	蜘蛛侠：英雄远征 Spider-Man: Far from Home (2019)	6320628	美国
530	影 (2018)	6864046	中国大陆 / 中国香港
73	Misérables, Les (1995)	1707386	英国 / 美国 / 法国
661	All Things Fair (1996)	0113720	瑞典 / 丹麦
1287	Until the End of the World (Bis ans Ende der Welt) (1991)	0101458	德国 / 法国 / 澳大利亚



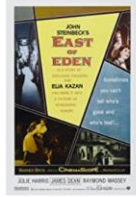
Golden Earrings (1947)



Excalibur (1981)



Pretty Woman (1990)



East of Eden (1955)

图 4-10 个人页面

[主页](#) / [剧情](#) / [黑暗时代](#) [Excalibur](#)

[黑暗时代](#) [Excalibur](#)



编剧: Rospo Pallenberg 豆瓣评分:6.7

主演: 尼古尔·特瑞 / 海伦·米伦 / 尼古拉斯·克莱 / 切瑞·朗吉 / 连姆·尼森 / 加布里埃尔·伯恩 / 帕特里克·斯图尔特 / 厄科尔·威廉森更多...

类型: 剧情 | 奇幻 | 冒险

制片国家/地区: 英国 / 美国

上映日期: 1981-04-10

片长: 140 分钟

又名: 神剑 / 亚瑟王神剑 / 石中剑


IMDb链接: tt0082348


评分:

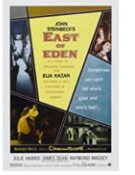
★★★★★

图 4-11 评分页面

编号	标题	imdb号	地区
1	Toy Story (1995)	0114709	美国
2	Jumanji (1995)	2283362	美国
3885	蜘蛛侠：英雄远征 Spider-Man: Far from Home (2019)	6320628	美国
530	影 (2018)	6864046	中国大陆 / 中国香港
73	Misérables, Les (1995)	1707386	英国 / 美国 / 法国
661	All Things Fair (1996)	0113720	瑞典 / 丹麦
1287	Until the End of the World (Bis ans Ende der Welt) (1991)	0101458	德国 / 法国 / 澳大利亚
10	GoldenEye (1995)	0113189	英国 / 美国
595	Window to Paris (1994)	0110719	俄罗斯 / 法国
2804	Excalibur (1981)	0082348	英国 / 美国










图 4-12 推荐电影更新后页面

## 4.2 系统后台模块

为了让管理员可以对用户信息和电影信息进行管理,开发了电影推荐系统的后台管理系统,管理员可以在后台查看所有用户的信息和电影的信息,也可以建立新的管理员,或者对用户信息和电影信息进行各种操作。后台登陆如图 4-13 所示。

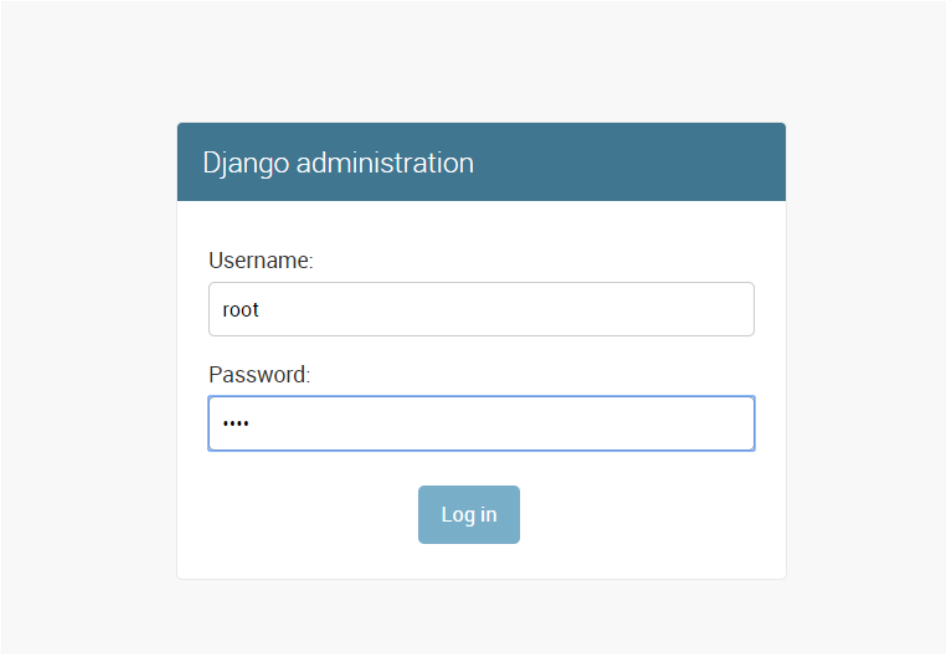


图 4-13 系统后台登录页面

### 4.2.1 密码修改

管理员密码修改页面供管理员修改密码,为网站的安全提供保障,如图 4-14 所示。

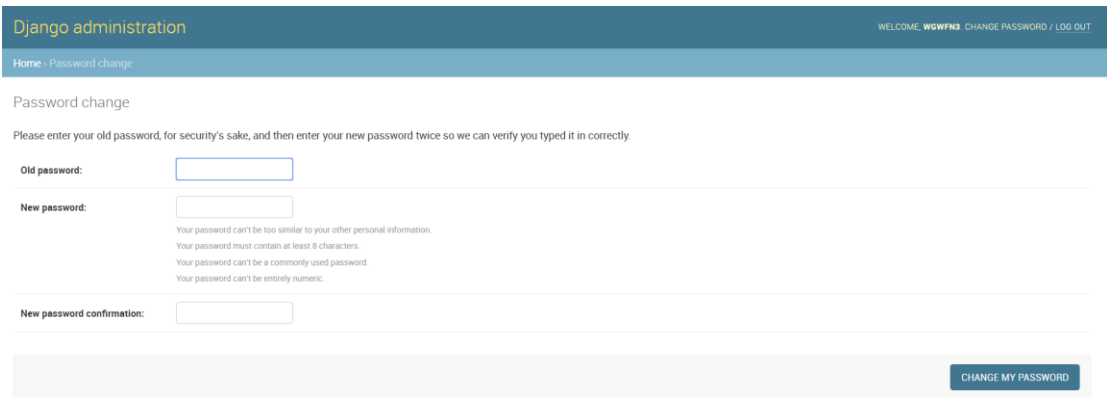


图 4-14 管理员密码修改

### 4.2.2 历史记录

管理员可以查看自己最近在后台的操作信息,如图 4-15 所示。

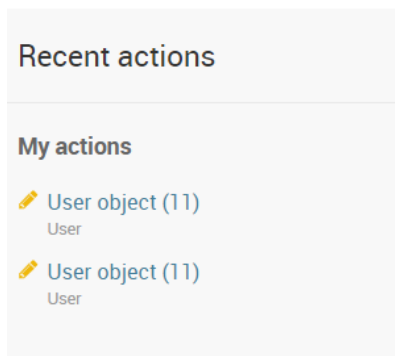


图 4-15 历史纪录

### 4.2.3 管理员添加

当网站需要多个不同的人来管理时可以通过添加管理员页面来增加管理员的数量，可以让多个人参与到后台的管理中，如图 4-16 所示。

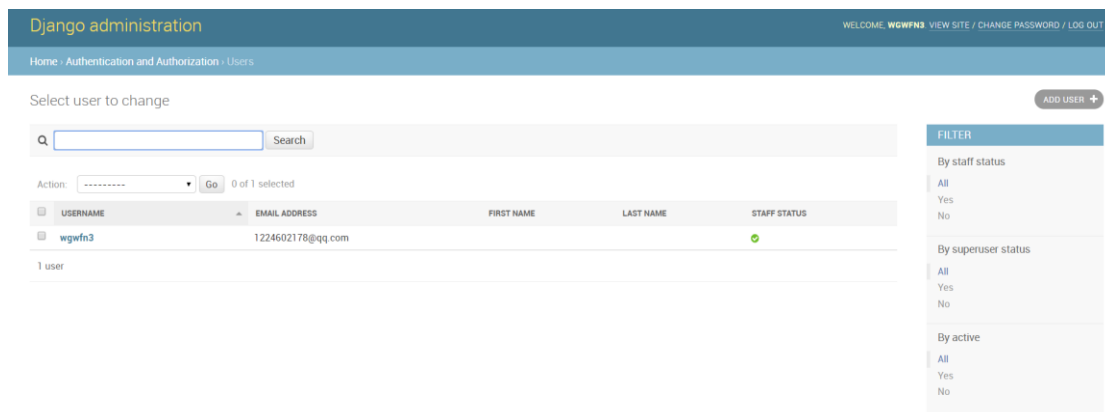


图 4-16 管理员添加

### 4.2.4 电影信息管理

管理员可以在后台界面查看所有的电影信息，也可以对电影信息进行更新，也可以添加或删除电影信息，如图 4-17 所示。

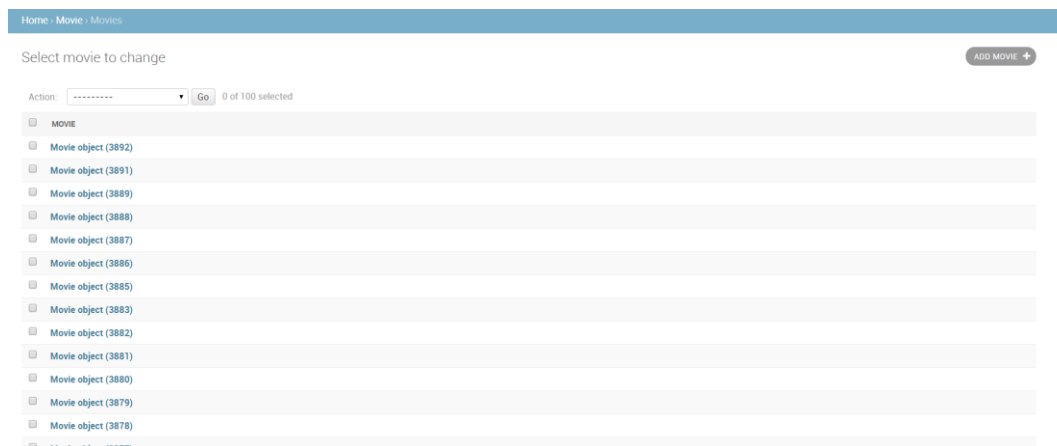


图 4-17 电影信息管理

### 4.2.5 评分信息管理

管理员可以在后台界面查看所有的用户评分信息，也可以对用户评分信息进行更新，也可以添加或删除用户评分信息，如图 4-18 所示。

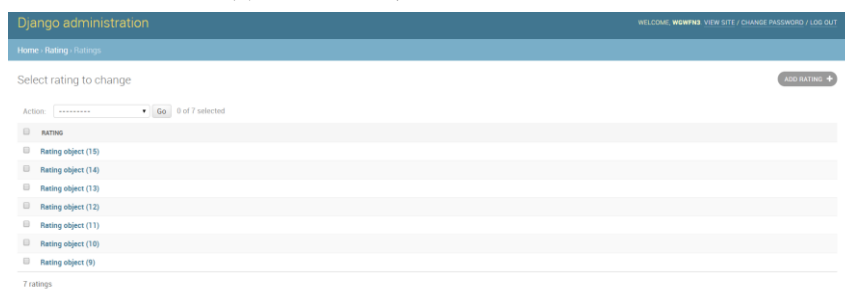


图 4-18 评分信息管理

### 4.2.6 用户信息管理

管理员可以在后台界面查看所有的用户信息，也可以对用户信息进行更新，也可以添加或删除用户信息，如图 4-19 所示。

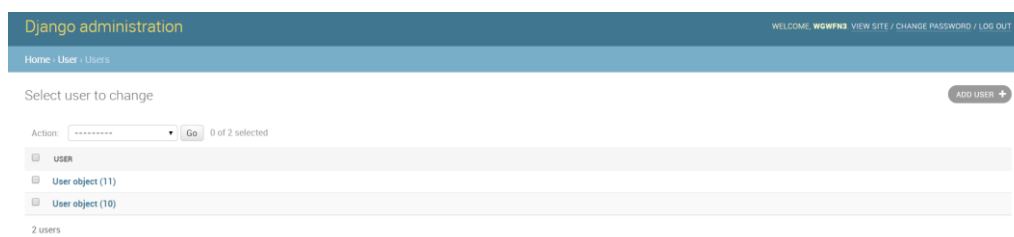


图 4-19 用户信息管理



## 5 总结

### 5.1 项目总结

基于协同过滤算法的电影推荐系统使用 MTV 的设计模式<sup>[9]</sup>，把模型层分为前端模块和后端模块两个部分，分别提供给用户和管理员使用。电影推荐系统以为用户提供电影信息和为用户推荐符合用户口味的电影为主要任务。此系统是在 pycharm 上完成的，基于 django 框架，并使用协同过滤算法为用户推荐电影，使用了 mysql 数据库，在前端使用了 bootstrap 框架并用 ajax 使系统可以在不刷新页面的情况下完成对用户评分的存储或更新。

技术应用如下：

(1) 协同过滤算法是一种较为著名和常用的推荐算法，它基于对用户历史行为数据的挖掘发现用户的喜好偏向，并预测用户可能喜好的产品进行推荐。也就是常见的“猜你喜欢”，和“购买了该商品的人也喜欢”等功能。

(2) Django 对传统的 MVC 设计模式进行了修改，将视图分成 View 模块和 Template 模块两部分，将动态的逻辑处理与静态的页面展现分离开<sup>[10]</sup>。而 Model 采用了 ORM 技术，将关系型数据库表抽象成面向对象的 Python 类，将表操作转换成类操作，避免了复杂的 SQL 语句编写<sup>[11]</sup>。

(3) 电影推荐系统使用 Django 的主要目的是因为 Django 在开发网站时十分的方便快捷，在前台完成后只需少量代码即可完成后台的搭建，方便管理员对数据库进行操作和管理<sup>[12]</sup>。它强调代码复用，多个组件可以很方便的以“插件”形式服务于整个框架，Django 有很多功能强大的第三方插件，也可以开发出自己的工具包，使得 Django 有很强的可扩展性，它还强调快速开发和 DRY 原则<sup>[13]</sup>。

电影推荐系统的特点如下：

(1) 电影推荐系统为用户提供丰富的电影信息并为用户完成电影推荐，节约用户的时间。

(2) 电影推荐系统为用户推荐影片时使用了两种推荐方式，一种是基于物品的协同过滤算法，一种是把 imdb 网站给出的电影评分进行排序并且选出评分最高的十部电影推荐给用户。为用户提供了多种选择，提高用户的满意度<sup>[14]</sup>。

(3) 电影推荐系统在为用户提供服务时可以根据用户的搜索内容来丰富自己的数据库。当用户访问或者搜索的电影并不在数据库中时，系统会启动爬虫，从豆瓣电影中获取信息并保存到数据库中。这样可以在保证用户使用的同时兼顾系统的数据更新。

电影推荐系统的缺点：随着用户的增多，用户的评分记录也越来越多，会导致协同过滤算法的计算时间变长；此系统使用的爬虫运行效率比较低。

## 5.2 个人收获与体会

随着互联网的飞速发展，人们对推荐系统的研究也在蓬勃发展，许多视频网站都有采用一些视频推荐算法来服务用户，比如：抖音，bilibili 等网站。所以我产生了做一个视频推荐网站来筛选自己喜欢的电影。在我研究了推荐系统的技术、推荐算法和国内外推荐系统的现状后决定使用协同过滤算法作为自己网站的推荐算法，协同推荐算法又包括基于用户的协同过滤算法和基于对象的协同过滤算法，在马老师的帮助下最终决定使用基于对象的协同过滤算法。看过网络上的资料后我觉得将系统分为电影显示模块、用户评分模块和推荐模块。在决定好推荐算法后需要使用电影评分的数据集，但是数据集里只有电影名称和评分，没有电影的其他信息<sup>[15]</sup>，无法完全满足系统的需求，在求助过室友和同学后，我决定使用爬虫从 imdb 上爬取电影的海报，因为 imdb 上没有电影的中文信息，所以电影的中文信息要从豆瓣电影上爬取。于是我开始在网络上找有关爬虫的学习资源，在一个星期的学习以后，我完成了爬虫的编写，只是爬虫的效率比较低。网页设计和数据库设计也是参考了网络上已有的资源。通过完成这个项目，我的感受是做事情不能三天打鱼两天晒网，在做项目时如果写了一部分后休息一段时间再重新开始的时候会忘记一些代码逻辑，于是需要重新花一些时间来熟悉自己写的代码，十分的浪费时间和精力。所以以后自己在做项目时不仅要注意代码的整洁也要重视注释的编写，当在自己的代码前加上注释时可以明显的增强代码的可读性，节省自己的时间。还有就是遇到无法解决的问题时要积极的向同学和老师寻找解决方法，通常自己遇到的问题同学和老师也都遇到过。此外还要重视网络上的学习资源，在我编写爬虫的时候网络上的课程给了自己莫大的帮助。完成这个项目我最大的感受就是要动手实践，很多时候只有自己动手实践才会发现问题，才能够真正的解决问题，才能更好的理解平时学到的知识。所以在以后的学习生活中我要不断的通过实践来学习新的知识并巩固旧的知识。电影推荐系统的实现是我人生中不可或缺的一段极其宝贵的经历。

## 参考文献

- [1] 范永全, 刘艳, 陆园. 社会化推荐系统的研究进展综述[J]. 现代计算机, 2014:30~31
- [2] CSDN. 推荐系统二---召回算法和业界最佳实践(一) [DB/OL]. 2018-12-16. [https://blog.csdn.net/weixin\\_40924580/article/details/85023267](https://blog.csdn.net/weixin_40924580/article/details/85023267)
- [3] 许海玲, 吴潇, 李晓东, 等 互联网推荐系统比较研究[J]. 软件学报. 2009, 20(2): 350~362
- [4] Hofmann T. Latent semantic models for collaborative filtering[J]. ACM Transactions on Information Systems, 2004, 22(1):89~115
- [5] 王璐. 基于协作过滤的 Web 服务推荐方法[D]. 东北大学, 2010
- [6] 马云飞. 成品油企业发票管理系统的设计与实现[D]. 西安电子科技大学, 2016
- [7] 李卓远, 曾丹, 张之江. 基于协同过滤和音乐情绪的音乐推荐系统研究[J]. 工业控制计算机, 2018, 31(07):127-128+131
- [8] 钟海涵. 基于聚类算法的推荐系统的设计与实现[D]. 湖南大学, 2017
- [9] 葛宇航. 基于 Django 的留学生信息管理系统设计与实现[J]. 通讯世界, 2019, 26(08):35~36
- [10] 赵海丹. 基于 LNMP 的智慧农业服务器平台的研究[D]. 浙江大学, 2015
- [11] 李炳. 面向农业工厂化生产过程数据平台优化研究[D]. 浙江理工大学, 2016
- [12] 纪占龙. 并行仿真运行配置管理技术的研究与实现[D]. 国防科学技术大学, 2011
- [13] 王密泉. 基于 Docker 的容器资源管理系统的设计与实现[D]. 大连理工大学, 2018
- [14] 钟德强, 黄英. 基于 KANO 模型的冰箱个性化需求分析[J]. 湖南工业大学学报(社会科学版), 2019, 24(02):48~53
- [15] 张琼林. 针对冷启动的分布式协同过滤推荐系统的研究[D]. 湖南工业大学. 2015