设计总说明

本文设计了一个个性化电影推荐系统. 众所周知，现在电影资源是网络资源的重要组成部分，随着网络上电影资源的数量越来越庞大，设计电影个性化推荐系统迫在眉睫. 所以本文旨在为每一个用户推荐与其兴趣爱好契合度较高的电影.

论文首先阐述推荐系统的研究现状以及意义，随后介绍了相关的推荐算法，重点介绍协同过滤算法，并对系统实现所需技术进行了研究，接着介绍了整个推荐系统的实现，最后对整个项目进行了回顾与总结.

本系统包含电影前端展示界面、电影评分板块、推荐算法的实现以及后端数据库的设计. 其中实现推荐算法是整个电影推荐系统的核心. 系统采用由grouplens项目组从美国著名电影网站movielens整理的ml-latest-small数据集，该数据集包含了671个用户对9000多部电影的10万条评分数据. 首先将该数据集包含的全部文件经过筛选重组之后存储到建好的数据库中，并将数据集按一定比例划分为训练集和测试集，对训练集进行算法分析生成Top-N个性化电影推荐列表，然后在测试集上对算法进行评测，至少包括准确率和召回率两种评测指标.

协同过滤算法是推荐领域最出名也是应用最广泛的推荐算法. 所以系统拟采用两种协同过滤算法给出两种不同的推荐结果，一种是基于用户的协同过滤算法，另一种是基于物品的协同过滤算法，用户可以根据两种推荐结果更加合理的选择合适的电影. 系统采用了改进之后的ItemCF-IUF和UserCF-IIF算法，对计算用户相似度和物品相似度的计算都做出了改进. 最后通过计算两种算法的准确率（Precision）、召回率(Recall)和流行度从而对系统进行评测、并比较了两种算法各自的优势和劣势. 实验证明，改进后的算法比原始的协同过滤算法推荐效果要好，准确率更高.

整个系统涉及到的编程语言包含Python、Html5、JQuery、CSS3以及MySQL数据库编程. 用到的框架是Django重量级web框架，通过该框架连接系统的前、后端. 用户首先需要填写用户名、密码以及邮箱注册系统，然后才能登陆推荐系统. 进入首页后会看到8个电影分类，包括恐怖片、动作片、剧情片等. 用户需要给自己看过的电影进行评分，评分起止为0.5-5.0分，共10个分段. 每评价一部电影就要点击一下提交按钮，将所评分的电影的imdbId号以及对应的评分存入数据库中. 用户点击“推荐结果”按钮，系统就调用推荐算法遍历数据库所存数据，得出推荐列表之后将结果反馈给浏览器，同时调取数据库所存电影海报图片进行展示. 用户点击自己登陆的昵称，会跳转页面显示自己已经评价过的电影.

本文还分析了系统的需求，并对需求进行相关设计，最后用Django框架实现了该系统，并给出了系统所用的主要数据表展示以及各个功能界面的展示.

关键词：电影推荐系统；协同过滤；基于邻域推荐；个性化服务

DESIGN GENERAL DESCRIPTION

This paper designs a personalized movie recommendation system. As we all know, nowdays, film resources are an important part of network resources. The number of film resources on the Internet is increasing. Designing a personalized movie recommendation system is imminent. Therefore, this project aims to implement a personalized movie recommendation system, recommending movies for each user in accordance with their interests.

The paper elaborates the research status and significance of the recommendation system firstly. Then it introduces the related recommendation algorithm, focuses on the collaborative filtering algorithm, and studies the required technology of the system implementation. Then it introduces the implementation of the entire recommendation system, finally reviews and summarizes the whole system.

The system includes the front-end display interface of the movie, the movie scoring board, the implementation of the recommendation algorithm, and the design of the back-end database. The implementation of the recommendation algorithm is the core of the entire movie recommendation system. The system plans to adopt the ml-latest-small dataset organized by the grouplens project team from the famous movie site movielens in the United States. This dataset contains 671 user ratings data for more than 9,000 movies. Firstly, the csv file included in the data set is stored in the database. The data set is divided into training set and test set. Algorithm analysis of the training set generates Top-N personalized movie recommendation list, and then the algorithm is evaluated on the test set, there include at least two indicators of test: accuracy and recall.

Collaborative filtering algorithms are the best known and most widely used recommendation algorithms. Therefore, the system proposes two collaborative filtering algorithms to give two different recommendation results. One is a user-based collaborative filtering algorithm, and the other is an item-based collaborative filtering algorithm. Users can make more reasonable choices based on the two recommended results. The right movie. The improved ItemCF-IUF and UserCF-IIF algorithms are used in the system to improve the calculation of user similarity and item similarity. Finally, the system is evaluated by calculating the precision, recall and popularity of the two algorithms, and the advantages and disadvantages of the two algorithms are compared. Experiments show that the improved algorithm is better than the original collaborative filtering algorithm and the accuracy is higher.

The programming languages involved in the entire system include Python, Html5, JQuery, CSS3 and MySQL database programming. The framework used is django's heavyweight web framework, connecting the system's front and back ends via the Django framework. The user first needs to fill in the username, password, and email registration system before logging in to the recommendation system. After entering the front page, you will see 8 movie categories, including horror films, action films, drama films, etc. Users need to rate the movies they have seen. The score starts from 0.5-5.0 points, a total of 10 segments. Each time you evaluate a movie, you must click the submit button to save the imdbId number of the movie you are rating and the corresponding rating into the database. When the user clicks the “recommendation result” button, the system invokes the recommendation algorithm to traverse the data stored in

the database, and after the recommendation list is obtained, the result is fed back to the browser, and the movie poster picture stored in the database is retrieved for display.When the user clicks on the nickname that he or she login, he will jump to the page to show the movie he has already evaluated.

This article also analyzes the requirements of the system, and related design of the requirements. Finally, the system is implemented using the Django framework, and given the main data table and function interface display .

Keywords: Movie recommendation system; Collaborative Filtering; criterion; Based on neighborhood recommendation; Personalized service

目录

第 1 章 绪论

1.1 研究背景及意义

随着互联网技术的快速发展，现在已经进入了大数据时代，网络上的信息呈现爆炸式增长，每天都会有数以亿计的数据涌现. 人们接触各种信息的途径也越来越丰富，比如微博、Facebook、Twitter、微信公众号等等. 而这些在给用户带来便利的同时也带来了前所未有的问题——“信息过载”[1].

“信息过载”就是指用户很难从庞大的数据中找到自己感兴趣的信息[2],[3]. 为了解决信息过载问题，首先出现的是搜索引擎，但是一旦用户无法准确描述自己所需的关键词，搜索引擎就无能为力了，且搜索是一种被动的检索. 而且，不同用户之间的需求差异很大. 如果只是单纯的靠搜索引擎以及无法满足获取自身需求的信息. 随着科技的发展，后来才有了推荐系统[4].

但是，最初的自动推荐系统，只是会将时下热门的、大众都爱的或者能使公司获得最大收益的产品推荐给用户，并没有针对每一个用户进行分析给出推荐. 这样推荐的效果非常不理想. 因此，人们希望有一种能向用户自动推荐项目的系统和方法，并且这个系统基于用户的偏好且对所推荐的产品进行属性分析. 这就是个性化推荐系统.

个性化推荐系统技术可以应用到很多网站上，比如图书网站、视频网站、音乐网站、交友网站等. 用户使用推荐系统的时间越长，不仅可以提高用户对该网站的忠诚度[5]，还能为网站带来更多收益. 最近几年推荐系统发展迅速，这要归功于Web2.0技术的成熟. 现在用户已经成为了网页的积极参与者而不再是被动的网页浏览者[6]. 所以，为众多平台用户提供个性化推荐迫在眉睫.

截止于2017年底，国内知名视频网站优酷网上的电影和电视剧数量已达16040部. 在如此庞大的视频数量下，怎样快速帮助用户发掘自己感兴趣的电影在网站运营中显得尤为重要. 而且看电影常常被用户当做一种放松娱乐的方式，所以用户在打开电影网站时可能没有明确想看的电影. 这样只有靠推荐系统通过分析用户的历史行为以及现下看的电影去分析潜在的用户可能感兴趣的电影.

1.2 国内外研究现状

“推荐系统”这个概念首次由Resnick在1997年提出[7]，此后就一直被广泛引用. 2007年，ACM推荐系统会议开始举办，这是第一个以“推荐系统”命名的顶级会议，旨在分享研究成果和方法，推动该领域的发展. 现在，AI（人工智能）、DM（数据挖掘）等学科的研究更加推动了推荐系统的发展.

如今，人们生活的方方面面已经离不开推荐系统了. 几乎所有的领域都在应用个性化推荐，比如电子商务、图书网站、社交网站等等.

• 电子商务推荐

Amazon是电子商务推荐系统的代表，当用户登录后进入亚马逊图书网站首页，就会看到“为您推荐”板块，系统根据你的历史行为生成了图书推荐列表，如图1-1所示. 还有一

个“与您浏览过的商品相关的推荐”板块. 正式这样有针对性的推荐，潜在的给Amazon带来了巨大的经济收益.

图1-1 亚马逊网站的推荐板块1

图1-2 亚马逊网站的推荐板块2

国内的淘宝网也是典型代表. 淘宝app首页的“必买清单”其实就是根据用户的浏览和购物行为而自动生成的产品推荐. 当用户购买商品之后，页面下方有“你可能喜欢”板块，这样对用户进行针对性的推荐，极大的促进了每日的交易量.

• 社交网络推荐

扎克伯格开发的Facebook是社交网络推荐领域的代表[16]，它也是利用社交网络数据进行相关推送. 比如里面的好友推荐界面，其实是根据用户现有的好友以及用户的行为记录给用户推荐新的好友. 因为如果社交网站里的用户的好友量很稀少，就不能体验到社会化的优势. 因此，好友推荐模块已经成为社交网络的标配.

• 视频推荐

Netflix是电影和视频推荐领域的代表，它举办了在整个业界都产生广泛影响的Netflix

比赛，极大促进了推荐技术的研究. 国内的爱奇艺视频网站也有相应的电影推荐模块.

• 音乐推荐

国内非常受大众喜爱的音乐播放器有“网易云音乐”. 打开界面就能看到“私人FM”板块，里面全部是根据用户历史听过的歌曲进行的推荐；“个性化推荐”板块如图1-3所示，其中展示了三个歌单，都是根据用户听过的歌曲所生成的推荐列表. 里面还包括了“每日歌曲推荐”，也是根据用户的音乐口味生成. 点进去查看如图1-4所示.

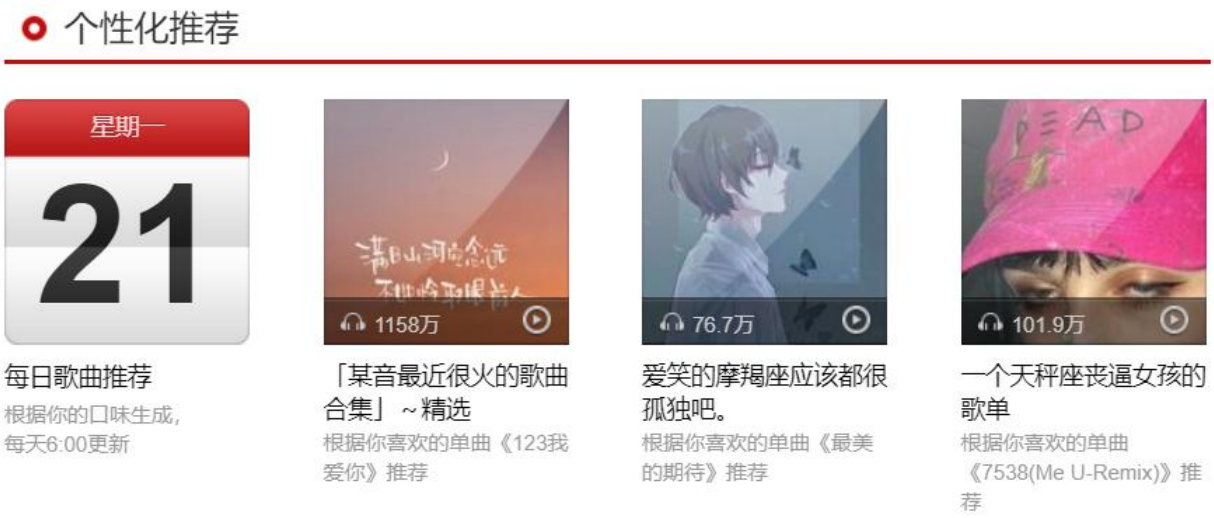

图1-3 网易云“个性化推荐”板块

图1-3 网易云“每日歌曲推荐”板块

国外对音乐个性化推荐系统的研究：认为用户所处的情境，比如用户当时所处的地点、时间以及天气等因素，都会对用户的偏好类型有影响[8]. 如Wang等人[9]用手机作传感器来收

集用户听音乐时的环境信息，然后构建出概率模型进行分析后给用户做推荐.

基本现在所有的推荐系统都是由前端web页面、推荐算法、后台日志系统组成. 其中最核心的就是推荐算法. 目前常用的算法有：协同过滤算法（Breese JS et al,1998）、基于内容的推荐算法（Mooney et al,2000）、 基于关联规则的推荐算法(Agrawal el)以及混合推荐算法（Claypool M et al,1999）.

基于内容的过滤算法是根据信息资源和用户兴趣的相似度来推荐商品. 能推荐新项目或者冷门的项目，使推荐系统不受冷启动和数据稀疏问题的影响. 其实搜索引擎就是基于内容的检索，该算法在国外的主要应用有Pazzani等人设计[19]的Syskill&Webert系统[17].

协同过滤是利用用户之间的相似性来推荐用户该兴趣的物品. 但有两个很难解决的问题，一个是稀疏性：系统初期由于系统资源还未获得足够的评价与反馈[9],[10]，另一个是可扩展性：用户和资源的增多会使系统性能越来越差.

1.3 本文研究目标和研究内容

本论文的研究目标是通过协同过滤算法实现一个个性化电影推荐系统：用户首先通过填写用户名、密码、邮箱地址注册后进入系统，然后对系统主页所展示的8个类别的电影中看过的电影进行评分，0.5分为最低分，满分为5分，所对应的评价分别是：不喜欢、一般、喜欢、推荐. 提交评分后浏览器将评分数据通过表单提交到数据库，推荐系统后台的分析算法通过UserCF（基于用户的协同过滤算法）和ItemCF（基于物品的协同过滤算法）给出两种推荐. 即一个是基于用户之间的相似度，一个是基于电影之间的相似度.

本文主要研究内容包括：

（1）研究原始的协同过滤算法，并调研应用该算法的视频网站.

（2）对原始的协同过滤算法做出改进，使推荐结果更加切符合用户兴趣.

（3）选定Top-N推荐的常用评价标准召回率和准确率与原始的协同过滤算法进行比较.

（4）以传统的协同过滤算法为基础，设计和实现一个个性化电影推荐系统，并从需求分析、系统设计、系统实现三个方面对该系统进行阐述.

1.4 论文结构安排

本文的结构安排如下：

第一章 绪论

主要阐述个性化推荐系统技术的研究背景及意义，说明推荐系统的实用性，讲解推荐算法的国内外研究现状以及推荐系统的国内外应用现状，并简述本文的研究目标以及研究内容.

第二章 推荐算法的研究

分析并介绍现有的几种推荐算法：协同过滤算法、基于内容的推荐算法、基于标签的推荐算法，重点分析协同过滤算法. 简要介绍相似度的计算公式：余弦相似度、欧式距离、皮尔逊(Pearson)相关系数以及曼哈顿距离，本系统所用的相似度计算公式为余弦相似度.还介绍了几种推荐系统常用的评测指标.

第三章 实验设计及系统实现相关技术的研究

介绍了三个实验的设计流程及结果分析，对算法进行了评测. 介绍了实现系统所需技术，重点介绍了Python、Django和MySQL.

第四章 推荐系统的设计与实现

对国内外主流视频网站进行调研.阐述了系统的需求分析，详细描述了系统的各个功能模块设计、数据库设计以及界面展示等方面的内容.

第五章 总结与展望

本章对整篇论文的研究过程进行回顾，并总结出不足之处，对推荐系统的后续研究做出相关探讨.

第 2 章 推荐算法的研究

2.1 推荐算法简介

2.1.1 协同过滤算法

协同过滤算法[12]主要包括基于用户的协同过滤算法（UserCF）和基于物品的协同过滤算法(ItemCF)两种推荐算法.

基于用户的协同过滤算法是推荐系统中最古老的算法，它是推荐系统诞生的标志[17]. 其主要包括两个步骤：

（1）找到和目标用户兴趣相似的用户集合

（2）找到这个集合中大多数用户都喜欢的，且目标用户没有买过或者看过的物品进行推荐[18]

基于物品的协同过滤算法目前使用也很普遍. 很多著名的视频网站如：Hulu、YouTube推荐算法的基础都是该算法. 该推荐算法的原理是向用户推荐和他们之前购买过的物品相似度很高的物品. 比如，该算法会因为你购买过《python编程从入门到实践》而给你推荐《用python写网络爬虫》. 主要分为两个步骤：

（1）计算各个物品间的相似度

（2）对物品的相似度从高到低进行排序，然后挑选一定数量的相似度较高物品进行推荐. 或者根据用户曾经的历史记录进行推荐.

2.1.2 基于内容的推荐算法

从每个物品中抽取一些特征，然后利用一个用户过去喜欢和不喜欢的物品的特征数据来学习此用户的喜好特征. 通过比较用户的特征和物品的特征，给用户推荐相似度较高的物品. 基于内容的推荐算法保护了用户之间的独立性，并且能很好的解决物品的冷启动问题.

基于内容的推荐算法可以充分将项目内容和用户本身的诸多特征联系起来，比如一部电影的演员、导演、类型，用户的性别、年龄、职业等.

其根据用户历史行为构造出用户偏好文档，计算所推荐的项目与用户偏好文档的相似度，选择最相似的项目推荐给用户. 首先为每个物品构建一个物品的属性资料，然后为每个用户建立其一个喜欢的物品集，最后计算其和物品属性资料的相似度. 相似度高意味着用户可能喜欢这个物品，相似度很低则意味着用户会讨厌这个物品[13].

2.1.3 基于标签的推荐算法

标签是一种无层次化结构的、用来描述信息的关键词，可以描述物品的内涵特征. 有很多网站都利用了标签系统. 比如：CiteULike论文书签网站，它允许科研人员收藏自己喜欢的论文并给论文打上标签[21]. 还有国内的豆瓣网站，豆瓣网允许用户对电影以及图书等打标签，系统利用打标签这个途径可以获悉图书和电影的内容以及所属类别.

2.2 系统实现算法介绍

2.2.1 基于用户的协同过滤算法

假设给定用户 u 和用户 v，令 N(u)表示用户 u 曾经评价过的电影集合，令 N(v)表示用户 v 曾经评价过的电影集合.通过余弦相似度计算用户 u 和 v 的兴趣相似度：

$$w_{uv} = \frac{|N(u) \cap N(v)|}{\sqrt{|N(u)||N(v)|}} \tag{2-1}$$

但是上述公式过于粗糙，本系统采用的计算相似度的公式是 John S.Breese 在论文[14]中提出的下列计算兴趣相似度的公式：

$$w_{uv} = \frac{\sum i \in N(u) \cap N(v) \frac{1}{\log(1+|N(i)|)}}{\sqrt{|N(u)||N(v)|}} \tag{2-2}$$

该改进过的基于用户的协同过滤算法记为 UserCF-IIF[14].式（2-2）通过 $\frac{1}{\log(1+|N(i)|)}$ 惩罚了两个用户的兴趣列表中包含热门电影而对他们相似度产生的影响.这样得到用户之间的相似度之后，推荐算法会给用户推荐与他兴趣最相似的 K 个用户喜欢的电影，下列公式计算了用户 u 对电影 j 的兴趣度.

$$p_{ui} = \sum_{v \in I(i) \cap S(u,k)} w_{uv} r_{vi} \tag{2-3}$$

式（2-3）中的 I(i)是对电影 i 打过分的用户集合. $S(u,k)$ 是和用户 u 有着相似兴趣爱好的 k 个用户的集合，w_{uv} 表示用户 u 和用户 v 之间的相似度，r_{vi} 衡量了用户 v 对电影 i 的兴趣度，在本系统里是指用户 v 对电影 i 的评分.最后按对电影的兴趣度从高到低排序，推荐前 N 部电影给用户 u，属于 Top-N 推荐.

2.2.2 基于物品的协同过滤算法

基于物品的协同过滤算法是计算物品之间的相似度，然后根据用户之前喜欢的物品来推荐用户可能喜欢的物品.

$$w_{ij} = \frac{|N(i) \cap N(j)|}{\sqrt{|N(i)||N(j)|}} \tag{2-4}$$

式（2-4）为计算物品相似度的公式，它惩罚了物品 j 的权重，运用到本系统中可以有效避免某一热门电影和很多其他多数电影都产生较大的相似度.

在基于物品的协同过滤算法中，两个物品之间有相似度是因为它们存在于很多用户的兴趣列表中.但是每个用户的贡献都不相同.活跃用户对物品相似度的贡献应该小于不活跃的用户.所以本系统采用下列改进过的计算物品相似度的公式：

$$w_{ij} = \frac{\sum u \in N(i) \cap N(j) \frac{1}{\log(1+|N(u)|)}}{\sqrt{|N(i)||N(j)|}} \tag{2-5}$$

式（2-5）中的$\frac{1}{\log(1+|N(u)|)}$也惩罚了活跃用户对物品相似度的贡献.之后通过下列公式计算来用户 v 对物品 j 的兴趣度.

$$p_{vj}=\sum_{i\in I(v)\cap S(j,k)} w_{ji}r_{vi} \tag{2-6}$$

式（2-6）中的 I(v)表示用户 v 喜欢的或者曾经产生过行为的物品集合，$S(j,k)$表示和物品 j 最相似的 k 个物品的集合，r_{vi}是用户 v 对物品 i 的兴趣度，w_{ji}为物品 j 和物品 i 之间的相似度，在本系统中取用户 v 对电影 i 的评分.改进过的基于物品的协同过滤算法记为 ItemCF-IUF[14].

2.3 相似度计算

在协同过滤算法中，相似度的计算至关重要.只有计算了用户或者物品之间的相似度才能得出推荐列表.常用的相似度计算方法有以下几种：

（1）余弦相似度

在 n 维空间中，任意两个向量之间的夹角的余弦值大小即代表这两个向量的相似程度.余弦值的取值范围为[-1,1].

假设 n 维空间中存在向量 i 和向量 j，式（2-7）为计算其余弦相似度的公式.

$$Sim(i,j)=\cos(\vec{i},\vec{j})=\frac{\vec{i}\bullet\vec{j}}{\|\vec{i}\|*\|\vec{j}\|} \tag{2-7}$$

（2）欧式距离

欧式距离也称为欧几里得度量，是指在 n 维向量空间中，任意两个向量之间的自然长度.同样的，假设存在向量 M 和 N，则欧式距离表达公式如式（2-8）所示.

$$Dist(M,N)=\sqrt{\sum_{i=1}^{n}(m_i-n_i)^2} \tag{2-8}$$

（3）皮尔逊(Pearson)相关系数

如果一个推荐算法使用的是皮尔森相关系数来计算相似度，那么该算法将会比使用余弦相似度公式计算相似度的算法的精确度更高[19].然而这是要建立在两个用户拥有较多共同评分项目的基础上，如果两个用户之间共同评分项目很少[19]，使用皮尔森相关系数进行计算无法反映出真实的相似度.

假设存在两个向量 J 与 K，这两个向量间的皮尔森相关系数计算公式如式（2-9）所示.

$$\rho_{J,K}=\frac{\sum(J-\overline{J})(K-\overline{K})}{\sqrt{\sum(J-\overline{J})^2\sum(K-\overline{K})^2}} \tag{2-9}$$

（4）曼哈顿距离

在平面上，坐标（x_1，y_1）与坐标（y_1，y_2）的曼哈顿距离为：

$$Sim(x,y)=|x_1-x_2|+|y_1-y_2| \tag{2-10}$$

2.4 推荐算法评测指标

2.4.1 评分预测

（1） RMSE，即均方根误差，其定义为：

$$RMSE=\sqrt{\frac{1}{|\mathrm{T}|}\sum_{u,i\in T}(r_{ui}-\hat{r}_{ui})^2} \tag{2-11}$$

式（2-11）中的 T 表示实验数据集的数量，r_{ui} 表示的是用户 u 对物品 i 的真实评分，$\hat{r}_{ui}$ 表示的是用户 u 对物品 i 的预测评分[14].

（2） MAE，即平均绝对误差，其定义为：

$$MAE=\frac{1}{|\mathrm{T}|}\sum_{u,i\in T}|r_{ui}-\hat{r}_{ui}| \tag{2-12}$$

2.4.2 TopN 推荐

通常，网站在提供预测服务时会选择为用户生成推荐列表，这种推荐就叫 Top-N 推荐，这种预测准确率一般是通过准确率或者召回率测量[14].

推荐结果的准确率定义为：

$$Precision=\frac{\sum_{u\in U}|R(u)\cap T(u)|}{\sum_{u\in U}|R(u)|} \tag{2-13}$$

推荐结果的召回率定义为：

$$Recall=\frac{\sum_{u\in U}|R(u)\cap T(u)|}{\sum_{u\in U}|T(u)|} \tag{2-14}$$

式（2-14）中的 R(u)为算法根据用户的训练集给出的推荐列表，T(u)是用户测试集列表.

2.5 本章小结

本章节主要阐述了推荐系统实现所需的算法基础、技术平台以及一些开发工具，主要介绍了协同过滤算法及常用的评测指标，同时简述了相似度的计算公式.

第 3 章 实验设计及系统实现相关技术的研究

3.1 实验设计及结果分析

本节将设计三个实验对第 2 章第 2 节介绍的改进后的两种算法与原始的协同过滤算法进行性能对比，并分析得出结论.

3.1.1 实验环境

对比实验使用的环境如表 3-1 所示：

表 3-1 实验环境

处理器	Intel(R) Core(TM) i5-7200U 2.5GHz
内存	4.00GB
运行环境	Windows 10
软件	Sublime Text3, MYSQL，Pycharm
编程语言	Python

3.1.2 实验设计

将数据集随机分为 10 份，其中的 7 份为训练集，另外的 3 份为测试集.在训练集上建立用户行为数据模型，在测试集上进行实验测评，并统计测评结果.由于本系统采用的是 Top-N 推荐，故使用准确率、召回率以及流行度来对算法进行评测.

• 实验一：对 UserCF 和 UserCF-IIF 算法分别取不同的推荐电影数目（N）计算准确率和召回率的值，得出 N 取多少时推荐算法的性能相对最好.

分别取推荐电影资源个数为 5，10，15，20 对 UserCF 算法进行对比实验，得到的结果数据如表 3-2 所示.

表 3-2 UserCF 算法的准确率和召回率

推荐电影数目 N	准确率	召回率
5	0.3768	0.0421
10	0.3244	0.0724
15	0.2881	0.0965
20	0.2629	0.1174

分别取推荐电影资源个数为 5，10，15，20 对 UserCF-IIF 算法进行对比实验，得到的结果数据如表 3-3 所示.

表 3-3　UserCF-IIF 算法的准确率和召回率

推荐电影数目 N	准确率	召回率
5	0.3782	0.0422
10	0.3264	0.0729
15	0.2925	0.0980
20	0.2660	0.1188

将上述两张表的数据画成折线图分别对比两个算法的准确率和召回率，如图 3-1 和图 3-2 所示.

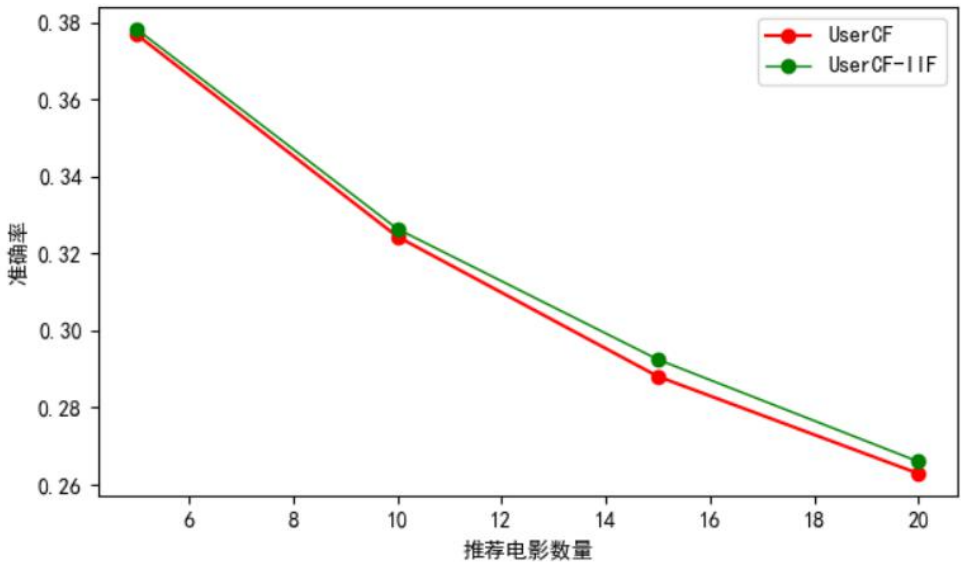

图 3-1　UserCF 和 UserCF-IIF 准确率对比

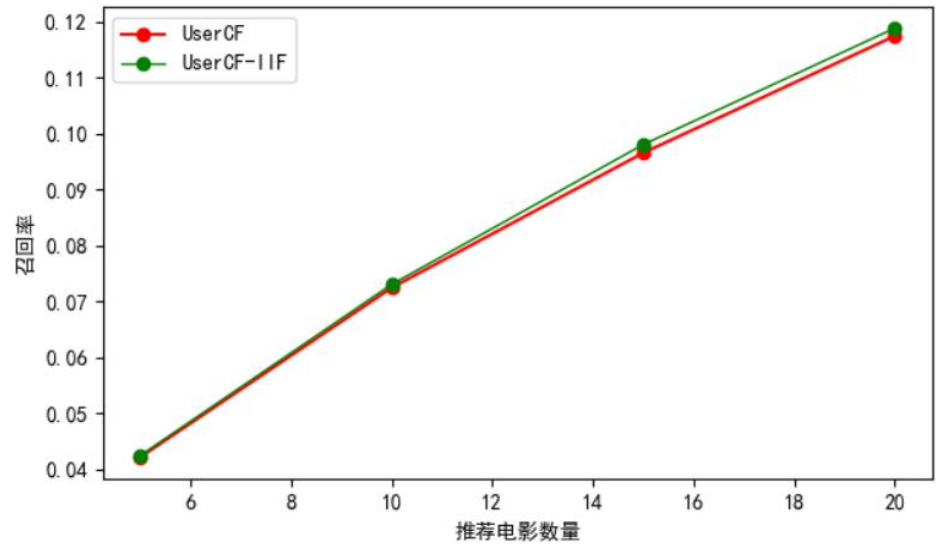

图 3-2　UserCF 和 UserCF-IIF 召回率对比

由图知，准确率随着推荐数目的增加呈递减趋势，召回率随着推荐数目的增加呈递增趋势.结合实际的推荐情况，选择 N=10 推荐效果最好.

并且改进之后的基于用户的协同过滤算法的准确率和召回率都比传统的算法略高.

• 实验二：对 ItemCF 和 ItemCF-IUF 算法也进行准确率和召回率的计算和对比.

通过图 3-3 和图 3-4 可以得出推论，ItemCF-IUF 算法的准确率和召回率都明显高于传统的 ItemCF 算法.

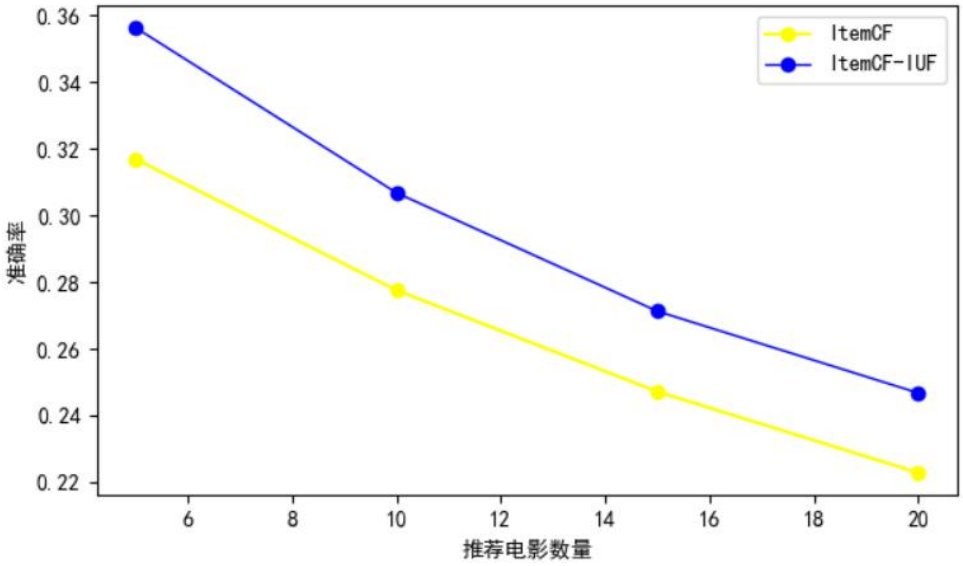

图 3-3 ItemCF 和 ItemCF-IUF 准确率对比

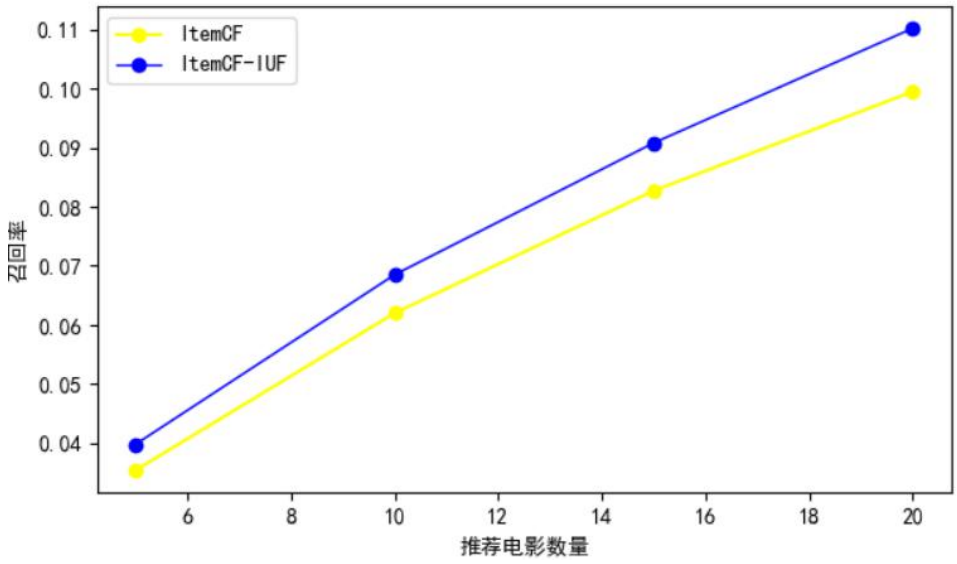

图 3-4　ItemCF 和 ItemCF-IUF 召回率对比

• 实验三：综合准确率、召回率、流行度的平均值对比 UserCF-IIF 和 UserCF 以及 ItemCF-IUF 和 ItemCF 算法性能.

分别计算出四个算法在 N 取 5、10、15、20 的准确率、召回率、流行度的值，并取平均数.

如表 3-4 所示，UserCF-IIF 在准确率和召回率上和 UserCF 算法相近，降低了推荐结果的流行度.如表 3-5 所示，ItemCF-IUF 在准确率和召回率上明显优于 ItemCF 算法，同样降低了推荐结果的流行度.这说明在计算物品相似度时考虑用户的活跃程度可以提升推荐结果的质量.

且对于本数据集，基于用户的协同过滤算法性能优于基于物品的协同过滤算法.

表 3-4 基于用户的协同过滤算法对比

算法	准确率	召回率	流行度
UserCF	0.3130	0.0821	4.8831
UserCF-IIF	0.3158	0.0830	4.8637

表 3-5 基于物品的协同过滤算法对比

算法	准确率	召回率	流行度
ItemCF	0.2660	0.0829	4.0427
ItemCF-IUF	0.2952	0.0831	4.0316

3.2 系统实现相关技术的研究

论文中推荐系统的开发与实现主要包括 web 前端技术、数据处理技术、推荐算法实现，系统开发所需技术如表 3-6 所示.

表 3-6 系统所需技术

技术类型	技术名称
推荐算法开发语言	Python3
前端开发语言	Html5、CSS3、Jquery
web 框架	Django
数据库	MySQL

3.2.1 Python 语言研究

Python 是由 Guido van Rossum 在八十年代末和九十年代初，在荷兰国家数学和计算机科学研究所设计出来的[16],[21].是一个高层次的结合了解释性、编译性、互动性和面向对象的脚本语言[21].相比于 C 或者 Java 等其他编程语言，Python 能让开发者很轻松的用几行代码就能实现一个功能完善的程序.

详细说来，Python 语言的优势有以下几点：

（1）易于学习：Python 相对其他语言比如 C++来说关键字较少，并且语法简单，这种特性可以让开发者轻而易举的上手.

（2）易于阅读：Python 严格控制代码缩进，使读者能够更加清晰的阅读代码.

（3）面向对象：Python 和 Java 类似，存在一个介于源代码文件和二进制代码文件之间的字节码文件，更有利于项目在不同的平台间移植.

（4）丰富的库：Python 拥有大量且丰富的标准库，可以减少很多不必要的代码，也可以更高效的对数据、图像进行处理.

（5）跨平台且开源.Python 可以跨平台运行，并且开放源码超过 20 年，能让开发者更加深入的了解 Python 的机制.

3.2.2 Django 框架研究

Django 是一个基于 Python 的 Web 框架，可以用来开发交互式网站.Django 采用的是 MTV 框架模式，即模型（Model），模板(Template)和视图(Views).它们各自的职责如表 3-7 所示.

由于 Django 是免费的，且公布了其源码，又拥有丰富的文档，所以使用的人员很多.其具体有以下几个优点：

（1）通用性：Django 可以和任何客户端框架一起工作，并且可以提供几乎任何格式的内容.

（2）安全性：Django 提供一种安全的方式管理用户的账户和密码，其 cookies 只包括一个秘钥，而实际数据存储在数据库中.并且 Django 可以防范许多漏洞，大大提高了网站的安全性.

（3）灵活性：由于 Django 是用 Python 语言所编写的，所以它不受服务器平台的限制.比如在 Linux 系统、Windows 系统或者 Mac 系统上都能运行程序.

表 3-7 MTV 各自职责

层次	职责
模型，即数据存取层	处理与数据相关的事物，比如存取数据、验证数据合法性
模板，即业务逻辑层	一般包含前端所有代码以及调用的图片
视图，即表现层	是模型与模板之间的桥梁

Django 的 MTV 各个组织之间的协作如图 3-5 所示.

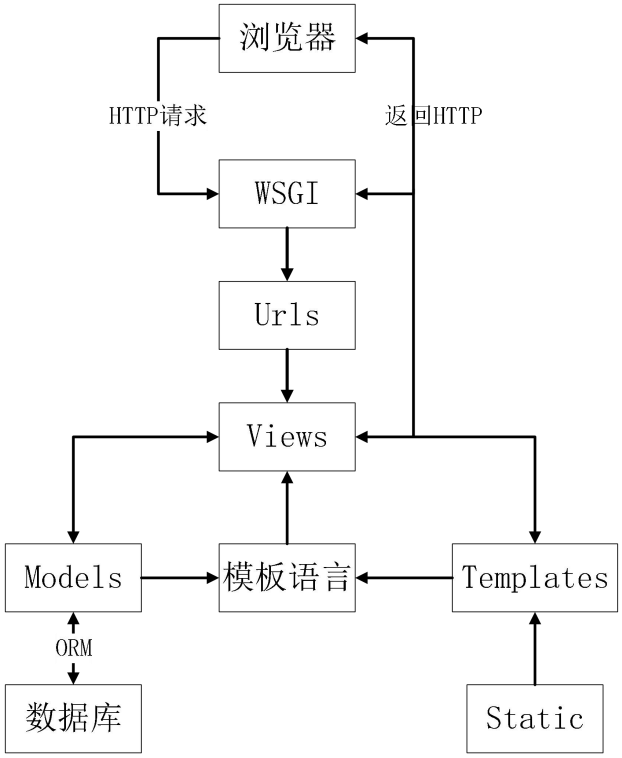

图 3-5 MTV 协作流程图

3.2.3 MySQL 数据库研究

数据库是一种用于存储数据集合的独立应用程序.每种数据库都会有一个或多个独特的 API，用来创建、访问、管理、搜索或复制数据库中保存的数据.关系数据库是建立在关系模型基础上的数据库，其所有数据都存储在不同的表中，表之间的关系由主键或者外键等连接.

MySQL 数据库就是一种快速易用的关系型数据库管理系统（RDBMS），是瑞典的 MySQL AB 公司开发的一个可用于各种流行操作系统平台的关系数据库系统，并且是一种免费的数据库管理系统.

MySQL 语言由 DDL（数据定义语言）、DML（数据操纵语言）、DCL（数据控制语言）组成.相比于其他的关系型数据库，MySQL 具有许多吸引人之处：

（1）比较容易使用. MySQL 是一个高性能并且相对很多其他数据库来说是比较简单的数据库系统.

（2）全面支持查询语言. MySQL 可以利用结构化查询语言，并且支持聚合函数，程序员可以在同一个查询中混合来自不同数据库的表.

（3）连接性和安全性. 完全支持网络化，其数据库可以在 Internet 上的任何地方访问，不会有限制.

（4）MySQL 是客户端/服务器架构的服务器，为客户端提供了不同的程序接口和连接库，且是多线程的[24].

（5）支持大型的数据库，可以方便的支持上千万条记录的数据库.

3.3 本章小结

本章简要介绍了系统所需实验环境，设计了三个实验对算法进行比较，第一个实验比较 UserCF-IIF 和 UserCF 算法的性能，第二个实验比较 ItemCF-IUF 和 ItemCF 算法的性能，第三个实验从准确率、召回率、流行度三个方面对基于用户的协同过滤算法和基于物品的协同过滤算法进行了比较，得出在本数据集的基础上，基于用户的协同过滤算法性能更优一些的结论.最后对系统所用的技术进行了研究，详细介绍了 Python 语言的优势、Django 框架和 MySQL 数据库的优点.

第 4 章 推荐系统的设计与实现

本章将改进后的基于用户的协同过滤算法和基于物品的协同过滤算法相结合应用到实际中，做出了个性化的电影推荐系统.本章首先对国内外的主流视频网站进行了调研，然后讲解系统的需求分析，介绍系统架构以及数据库，最后展示系统界面.

4.1 国内外主流视频网站推荐效果调研

4.1.1 国内视频网站调研

调研了作为国内第一大视频网站的优酷网.图 4-1 为优酷视频网站首页的“优酷懂你”板块为某用户给出的推荐列表[17]，图 4-2 为该用户的观看记录.

对比观察用户的观看记录和优酷给出的推荐列表，发现该用户比较偏向看综艺、日剧以及恐怖片.但是推荐列表中的视频基本都与用户的喜好不同.其实优酷网的这个版块只是推荐了一些热门视频，缺乏个性化和针对性，推荐效果很差，既浪费用户时间也降低了用户的体验.

图 4-1 “优酷懂你”给出的推荐

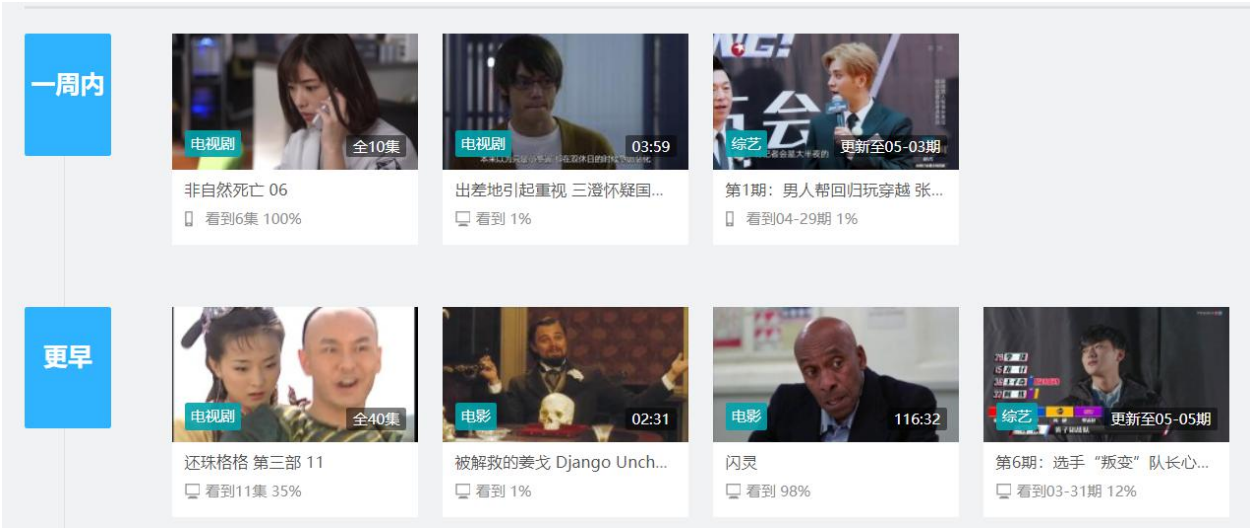

图 4-2 用户观看记录

4.1.2 国外视频网站调研

调研了在全世界都著名的 Youtube 视频网站. 用户登录后进入首页就有“推荐视频”板块，如图 4-3 所示. 其中推荐的视频都是根据用户的历史看过的视频分析所得, 而且每个视频下方还有上传网站的时间. 该用户的历史记录所看的视频节选如图 4-4 和图 4-5 所示.

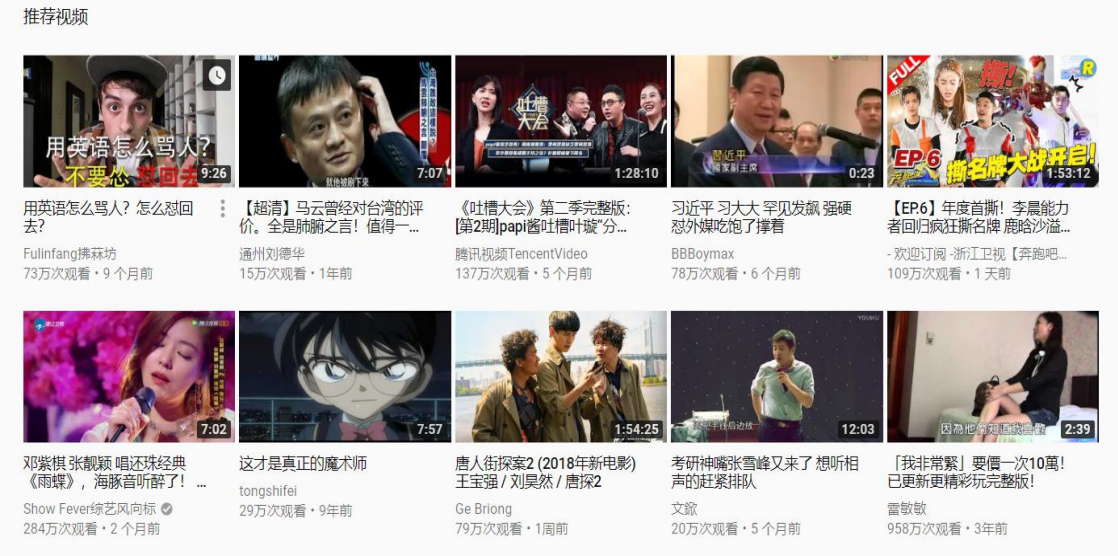

图 4-3 Youtube 首页的推荐板块

图 4-4 用户历史记录节选 1

图 4-5 用户历史记录节选 2

4.2 需求分析

随着电影市场的迅速发展，每天都有大量电影上映.人们都希望可以高效的在海量电影库中找到自己可能会喜欢的电影，以节省寻找电影的时间[22].电影推荐系统能给用户带来便利.本文要实现的是一个面向用户的个性化电影推荐系统，根据 movielens 数据集里面大量用户对电影的评分数据，通过计算用户相似性、电影相似性，实现为用户推荐符合其兴趣的电影.

本文实现的个性化电影推荐系统有以下几点基本需求：

（1）数据集：每个用户所评电影数量要多，尽量广泛涉及大量电影

（2）推荐算法：推荐效果要良好

（3）包括用户注册登录在内的整个 web 系统

（4）系统要易于扩展和维护

4.3 用户功能需求

如图 4-6 是系统中用户的用例图，有 5 个用例，分别是注册、登录、注销、评分、查看推荐结果.

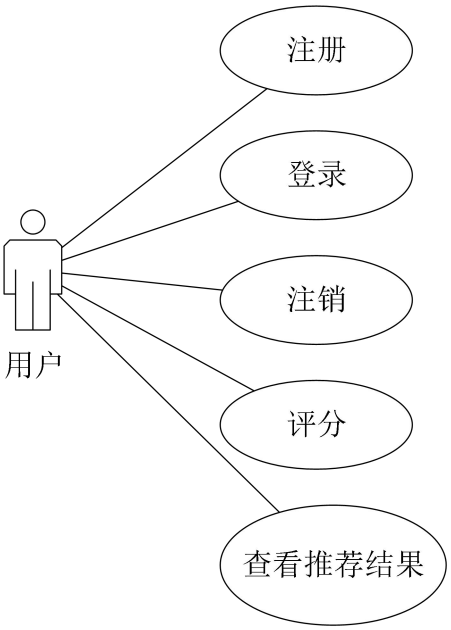

图 4-6 用户的用例图

4.4 系统设计

4.4.1 系统总体架构

本文从互联网上下载 movielens 数据集，经过数据重组和筛选，基于两种推荐算法得出推荐结果保存至 MySQL 数据库中，并通过 Django 框架进行前端展示.本系统采用 B/S（浏览器/服务器）体系结构，用户通过浏览器就能和网站上的内容交互.

实现本系统主要需要以下几种编程语言：

（1）Python：进行后台开发，写推荐算法，和 MySQL 数据库交互，将用户的数据存储到数据库中，又将生成的推荐列表展示到前端页面.

（2）Html5：进行前端页面的开发.

（3）Css3：美化前端页面，特别是对电影分类板块做处理.

（4）Jquery：实现提交表单和首页中的星星评分效果.

系统的总体架构设计图如图 4-7 所示.

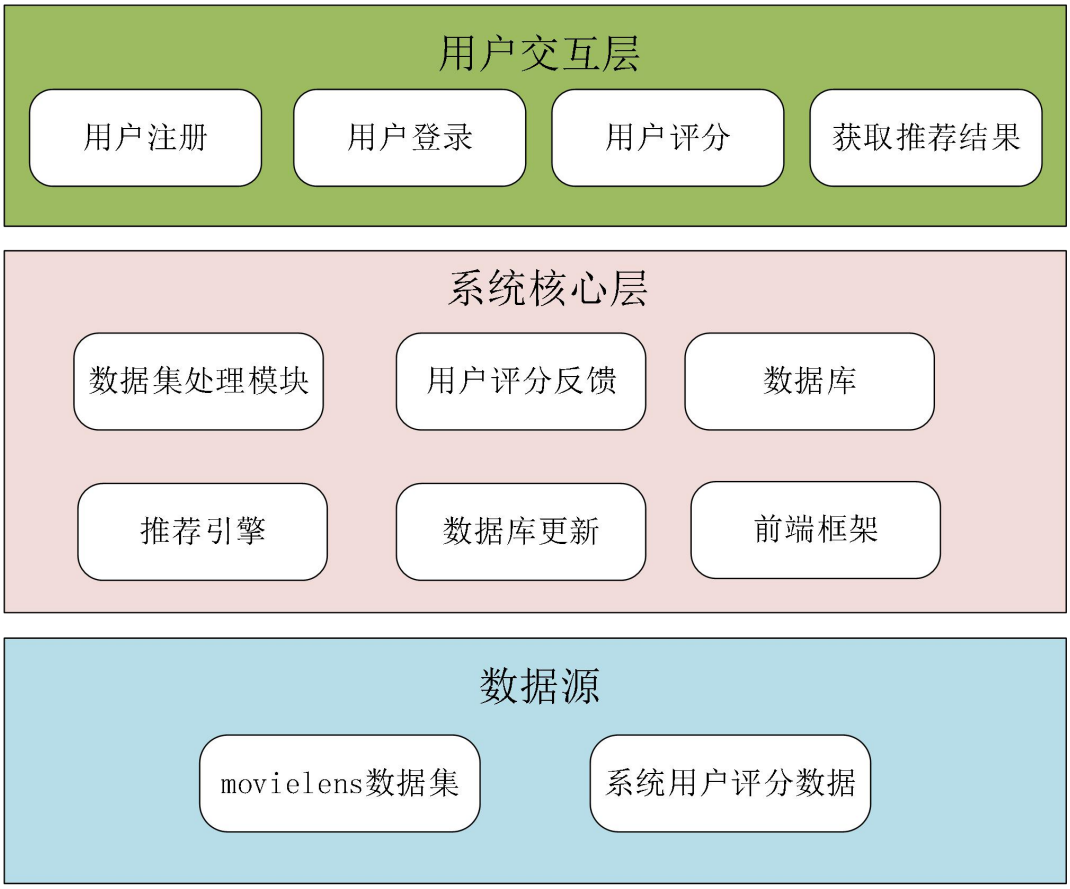

图 4-7 系统架构图

一个良好的推荐系统，必须要有一个良好的用户交互界面和用户行为数据[16]. 其之间的关系如图 4-8 所示.

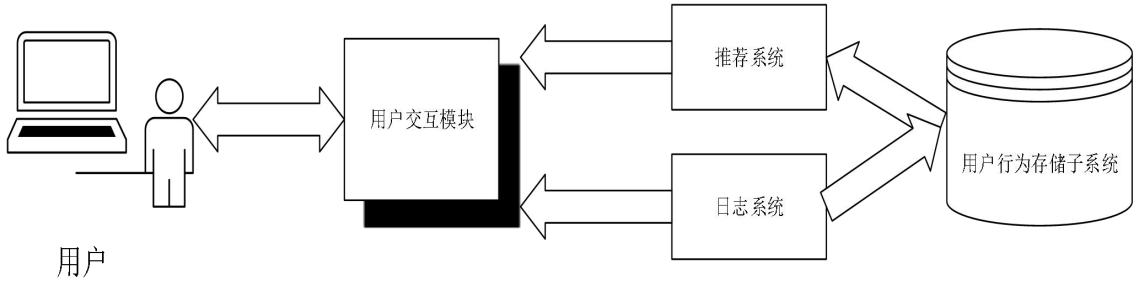

图 4-8 推荐系统和日志系统之间的联系

4.4.2 系统功能模块简述

本系统以改进后的用户协同过滤和物品协同过滤算法为依据，采用 django 重量级 Web 框架设计并实现.包括了数据集处理模块、注册登录模块、电影分类模块、用户评分反馈模块、推荐算法模块和推荐电影展示模块.

① 数据集处理模块

本文设计的推荐系统的源数据集来源于 movielens 的 ml-latestl-small，其中包含 671 个用户的 10 万条评分数据.对数据集里面的 ratings.csv 和 links.csv 文件进行连接处理，只保留 userId、imdbId、rating 三个字段存入数据库中新建好的数据表 users_resulttable 中.

② 注册登录模块

用户只有先注册并且登录系统之后才能提交对电影的评分. 注册界面含用户名、电子邮箱地址以及密码. 注册和登录界面如图 4-9 所示.

图 4-9 注册和登录界面

用户注册界面的核心代码如下：

```
@register.html
<div class="unit-1-2 unit-1-on-mobile">
    <h3>注册</h3>
    <form class="form" action="{% url 'users:register' %}" method="post">
            {% csrf_token %}
            {% for field in form %}
            {{ field.label_tag }}
            {{ field }}
            {{ field.errors }}
            {% if field.help_text %}
            <p class="help text-small text-muted">{{ field.help_text|safe }}</p>
            {% endif %}
            {% endfor %}
            <button type="submit" class="btn btn-primary btn-block">注册</button>
    </form>
     <div class="flex-center top-gap text-small">
            <a href="login.html">已有账号登录</a>
     </div>
</div>
```

用户登录界面的核心代码如下：

@login.html

```
<div class="unit-1-2 unit-1-on-mobile">
    <h3>登录</h3>
    <form class="form" action="{% url 'login' %}" method="post">
        {% csrf_token %}
        {{ form.non_field_errors }}
        {% for field in form %}
            {{ field.label_tag }}
            {{ field }}
            {{ field.errors }}
            {% if field.help_text %}
             <p class="help text-small text-muted">{{ field.help_text|safe }}</p>
            {% endif %}
        {% endfor %}
        <button type="submit" class="btn btn-primary btn-block">登录</button>
        <input type="hidden" name="next" value="{{ next }}"/>
    </form>
    <div class="flex-left top-gap text-small">
    <div class="unit-2-3"><span>没有账号？ <a href="{% url 'users:register' %}">立即注册</a></span></div>
    </div>
</div>
```

③ 电影分类模块

系统首页一共有 8 种类型的电影，如 4-10 所示.

从左到右依次是动作片、恐怖片、喜剧片、动画片、科幻片、犯罪片、爱情片以及剧情片.

图 4-10 系统首页

图 4-11 是爱情片的展示界面.

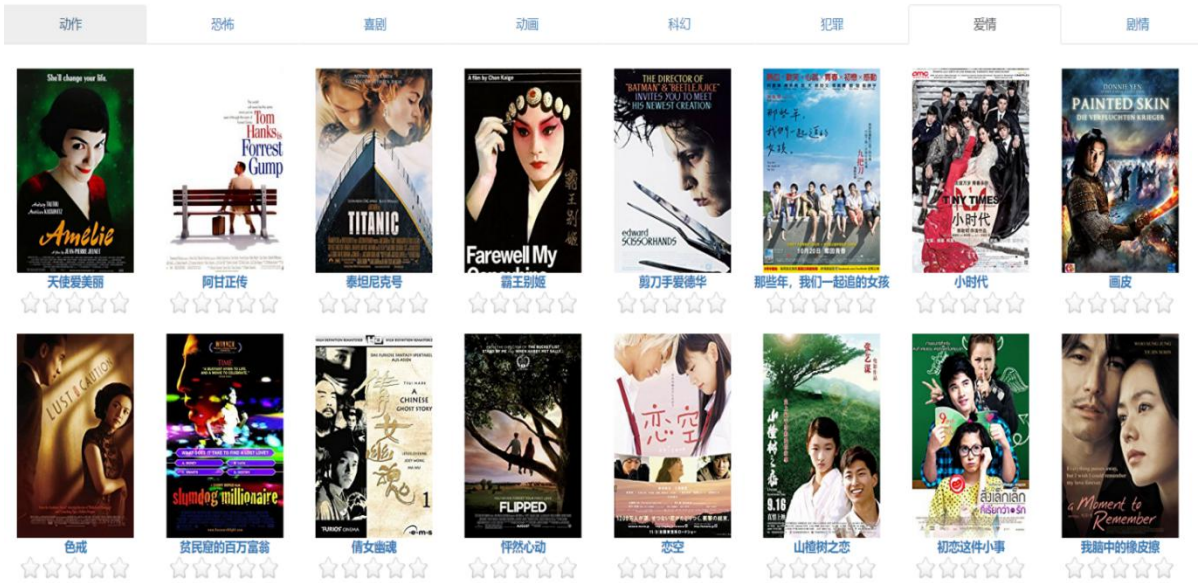

图 4-11 爱情片展示界面

实现分类效果的部分代码如下：

@index.html

```
<div>
    <hr />
    <ul class="nav nav-tabs nav-justified" id="index">
        <li class="active"><a href="#tab0" data-toggle="tab">动作</a></li>
        <li><a href="#tab1" data-toggle="tab">恐怖</a></li>
        <li><a href="#tab2" data-toggle="tab">喜剧</a></li>
        <li><a href="#tab3" data-toggle="tab">动画</a></li>
        <li><a href="#tab4" data-toggle="tab">科幻</a></li>
        <li><a href="#tab5" data-toggle="tab">犯罪</a></li>
        <li><a href="#tab6" data-toggle="tab">爱情</a></li>
        <li><a href="#tab7" data-toggle="tab">剧情</a></li>
    </ul>
</div>
```

每一部电影包括电影海报、电影名字、imdb 网站的超链接以及星星评分. 代码如下：

@index.html

```
<li class="list_item" data-trigger-class="list_item_hover">
    <a _boss="film"    target="_blank" class="figure" tabindex="-1">
            <img src="{% static 'img/action/103776.jpg' %}" alt="蝙蝠侠归来">
    </a>
    <strong class="figure_title">
            <a _boss="film"      href="http://www.imdb.com/title/tt0103776/"  target="_blank"
title="蝙蝠侠归来">蝙蝠侠归来(8.1 分)</a>
            <div class="evaluate">
                <div class="starts" id = "103776"></div>
            </div>
    </strong>
</li>
```

点击电影名字会链接到 imdb 网站，用户可以浏览电影的详情页. 图 4-12 是电影《千与千寻》的 imdb 网站的展示页面.

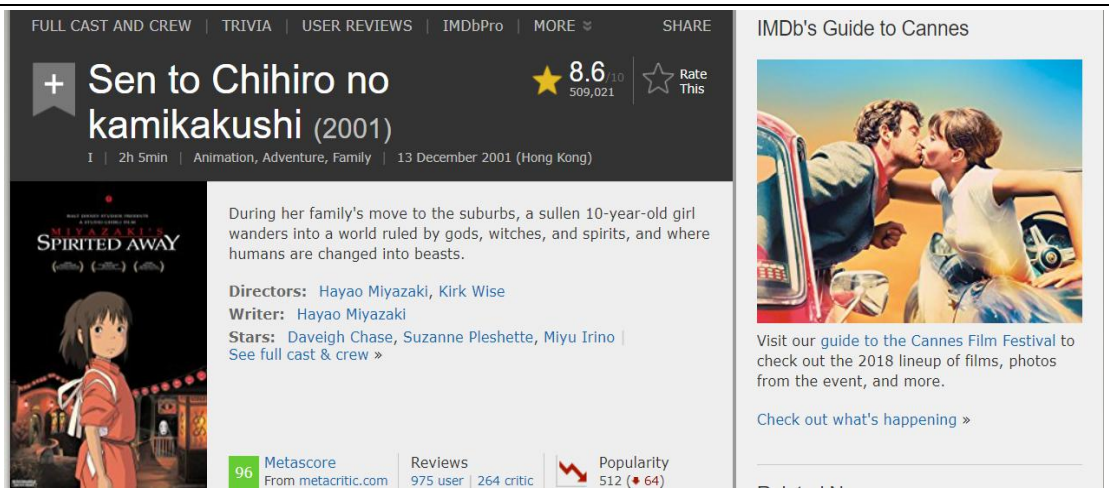

图 4-12 《千与千寻》的 imdb 网站详情页

④ 用户评分反馈模块

系统利用了 jquery-raty 评分插件，收集用户对每一部电影的评分. 用户通过点击‘提交评分’按钮将评分数据提交到 MySQL 中的 users_resulttable 表中，插入到源数据集末尾.

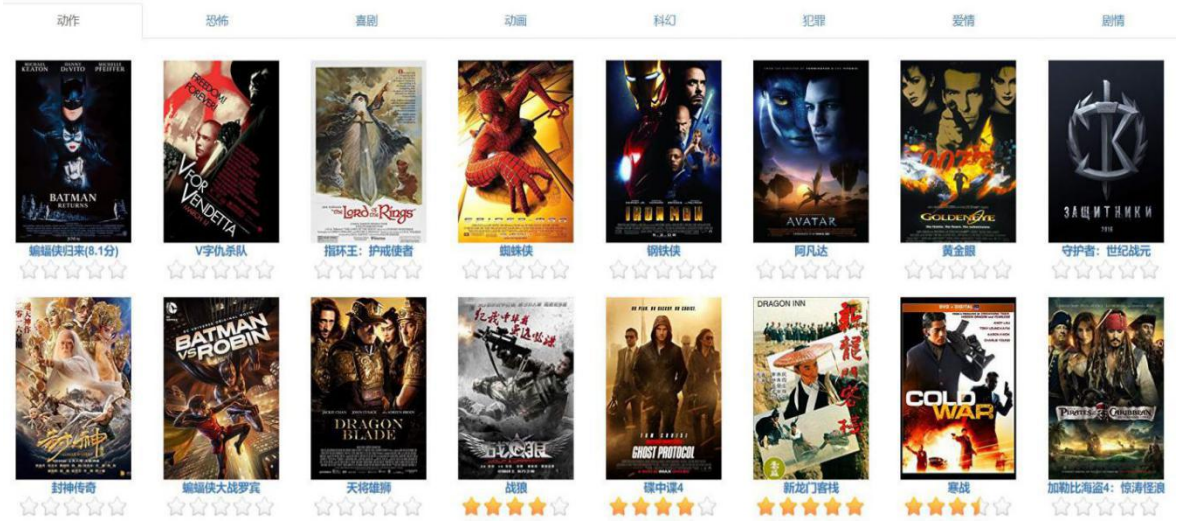

图 4-13 用户对动作片评分

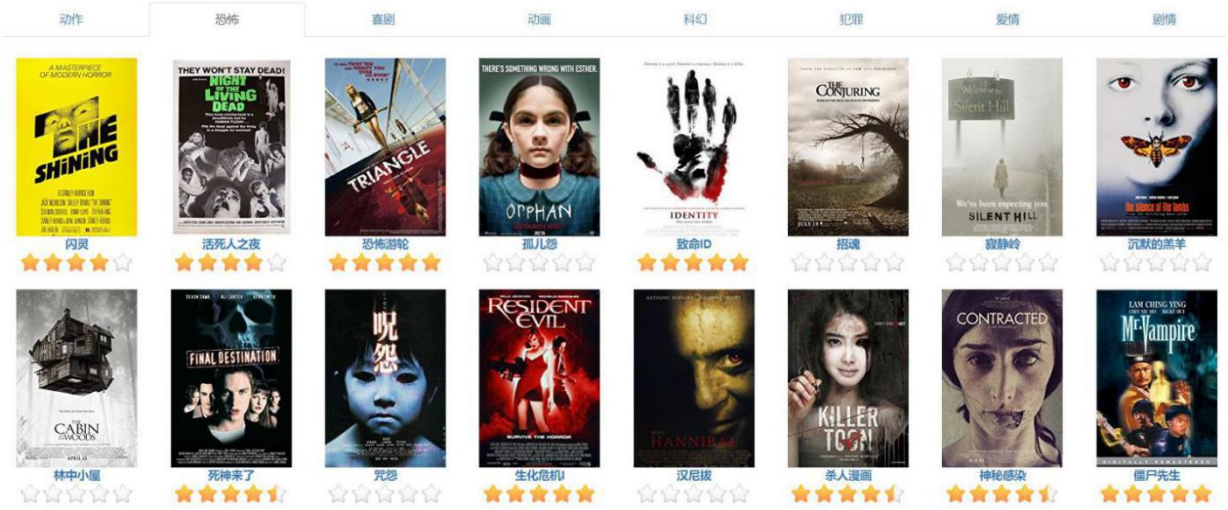

图 4-14 用户对恐怖片评分

图 4-13 和图 4-14 分别是某用户对电影分类模块中的动作片和恐怖片的评分图.

实现星星评分效果的核心代码如下：

```
@index.html
$(".starts").raty({
        number : 5,//星星个数
        {#score :3,#}
        path : '{% static 'img' %}',//图片路径
        {#target : '#grade',//#}
        {#hints : ['0.5','1','1.5','2','2.5','3','3.5','4','4.5','5'],#}
        hints : ['不喜欢','不喜欢','一般','喜欢','推荐'],
        starHalf:'star-half-big.png',
        half:true,
        round:{down:.26,full:.7,up:.9},
        click : function(score, evt) {
            if((score<0.5)) result = 0.5;
            else if((score>0.51)&&(score<1.0)) result = 1.0;
            else if((score>1.1)&&(score<1.5)) result = 1.5;
            else if((score>1.5)&&(score<2)) result = 2.0;
            else if((score>2.0)&&(score<2.5)) result = 2.5;
            else if((score>2.5)&&(score<3.0)) result = 3.0;
            else if((score>3.0)&&(score<3.5)) result = 3.5;
            else if((score>3.5)&&(score<4.0)) result = 4.0;
            else if((score>4.0)&&(score<4.5)) result = 4.5;
            else    result = 5.0;
            $("#rating").val(result.toFixed(1));
            {#alert(result.toFixed(1));#}
        }
    });
```

提交用户对电影的评分相关代码如下：

```
@index.html
<form action="/insert" method="get" target="nm_iframe">
   <input type="hidden" name="userId" id = "userId"> <br>
   评分: <input type="text" name="rating"    id = "rating"> <br>
   电影 ImdbId: <input type="text" name="imdbId" id = "imdbId"> <br>
   <input type="submit" value="提交评分">
</form>
<iframe id="id_iframe" name="nm_iframe" style="display:none;"></iframe>

<script type="text/javascript">
$(document).ready(function(){
   $(".starts").click(function(){
      $("#imdbId").val(this.id);
      $("#userId").val({{ user.id}});
   });
</script>
```

```
def insert(request):
    global USERID
    USERID = int(request.GET["userId"])+1000
    RATING = float(request.GET["rating"])
    IMDBID = int(request.GET["imdbId"])
    Resulttable.objects.create(userId=USERID, rating=RATING,imdbId=IMDBID)
    return HttpResponseRedirect('/')
```

⑤ 用户评分记录模块

用户登录系统后点击自己的昵称，将跳转页面显示自己曾经评价过的电影列表. 如图 4-15 所示.

图 4-15 某用户评价过的电影列表

相关代码如下：

@views.py

```
def showmessage(request):
    usermovieid = []
    usermovietitle = []
    data=Resulttable.objects.filter(userId=1001)
    for row in data:
        usermovieid.append(row.imdbId)
    try:
        conn = get_conn()
        cur = conn.cursor()
        #Insertposter.objects.filter(userId=USERID).delete()
        for i in usermovieid:
            cur.execute('select * from moviegenre3 where imdbId = %s',i)
            rr = cur.fetchall()
            for imdbId,title,poster in rr:
                usermovietitle.append(title)
                print(title)
    finally:
        conn.close()
    return render(request, 'users/message.html', locals())
```

```
@message.html
<html lang="en">
  <head>
  <meta charset="UTF-8">
  <title>用户评过的电影</title>
  <style>
    h2{
        text-align: center;
        font-family: Georgia, serif;
        font-weight: bold
    }
    li{
        list-style: none;
        text-align: center;
        {#display: inline-block;#}
    }
  </style>
  </head>
  <body>
    <h2>您评价过下列电影：</h2>
    <br/>
    {% for dd in usermovietitle %}
    <li>
        <p>{{ dd }}</p>
    </li>
    {%   endfor %}
  </body>
</html>
```

⑥ 推荐算法模块

本模块是整个系统的核心组成部分，当用户登录进入系统并对电影进行评分之后，系统就记录下了该用户的 id 号，当用户再次登入系统时，系统会将源数据集和该用户之前的评分数据整合成新的数据表作为推荐的依据.

整个推荐过程流程图如图 4-16 所示.

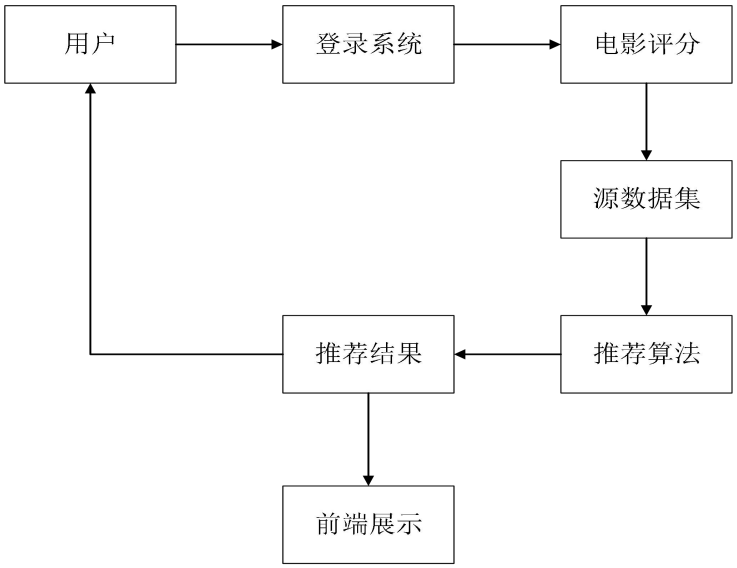

图 4-16 系统推荐流程图

算法生成推荐电影的 imdbId 号之后存入 matrix 全局变量列表中，并在 moviegenre 数据表中查询到相关的电影名称以及电影海报的链接，将 userId、title、poster 数据插入到 users_insertposter 数据表中，此表即为所有登录用户的电影推荐列表集. 得出用户的电影推荐结果并插入数据表中的核心代码如下：

@views.py

```
def recommend(self, user):
    matrix.clear()
    K = self.n_sim_user
    N = self.n_rec_movie
    rank = dict()
    watched_movies = self.trainset[user]
    for similar_user, similarity_factor in sorted(user_sim_mat[user].items(),
                                                  key=itemgetter(1),
    reverse=True)[ 0:K]:

        for imdbid in self.trainset[similar_user]:
            if imdbid in watched_movies:
                continue
            rank.setdefault(imdbid, 0)
            rank[imdbid] += similarity_factor
            rank_ = sorted(rank.items(), key=itemgetter(1), reverse=True)[0:N]
            for key,value in rank_:
          matrix.append(key)       #matrix 为存储推荐的 imdbId 号的数组
    return matrix

def recommend1(request):
    Insertposter.objects.filter(userId=USERID).delete()
    read_mysql_to_csv('users/static/users_resulttable.csv',USERID)   #追加数据，提高速率
    ratingfile2 = os.path.join('users/static', 'users_resulttable.csv')
    usercf = UserBasedCF()
    userid = str(USERID)
    print(userid)
    usercf.generate_dataset(ratingfile2)
    usercf.calc_user_sim()
    usercf.recommend(userid)
    try:
        conn = get_conn()
        cur = conn.cursor()
        for i in matrix:
            cur.execute('select * from moviegenre3 where imdbId = %s',i)
            rr = cur.fetchall()
            for imdbId,title,poster in rr:
                if(Insertposter.objects.filter(title=title)):
                    continue
                else:
                    Insertposter.objects.create(userId=USERID, title=title, poster=poster)

finally:
```

```
        Conn.close()
    results = Insertposter.objects.filter(userId=USERID)
        return render(request, 'users/movieRecommend.html',locals())
```

⑦ 显示推荐模块

根据登录用户对电影的评分反馈，通过基于用户的协同过滤和基于物品的协同过滤算法给出图 4-17 和图 4-18 两种推荐结果.

图 4-17 基于用户的推荐结果

图 4-18 基于物品的推荐结果

算法从 users_insertposter 数据表中获取相应登录用户的推荐电影列表相关内容展示到前端界面. 上述过程的核心代码如下：

```
@movieRecommend.html

{% for result in results %}
  <li>
      <p>{{ result.title }}</p>
      <img src="{{ result.poster }}" alt="">
  </li>
{%   endfor %}
```

4.5 数据库介绍与设计

4.5.1 实验数据集介绍

本实验采用的数据集是由 GroupLens 科研小组从电影网站 MovieLens 上统计的 ml-latest-small 数据集，该数据集包含了从 1995 年到 2016 年 10 月内 671 名用户对 9125 部电影的评分记录，一共 100004 条电影评分.里面的每一名用户评价的电影数量都不少于 20 部.该 ml-latest-small 数据集由四个 csv 文件组成，分别是 ratings.csv、movies.csv、links.csv 以及 tags.csv[25].下面将一一介绍.

ratings.csv 是用户评分信息表，第一行是 4 个 title：userId(用户 id)、电影 id(movieId)、用户评分(rating)以及 timestamp(时间戳)，其中用户评分区间为 0.5 分-5 分.表 4-1 即为 ratings.csv 文件的节选.movies.csv 是电影属性表，包含三个字段，分别是 movieId(电影的 id 号)、title(电影名称)、genre（电影的题材），表 4-2 即为 movies.csv 文件的节选.

表 4-1 ratings.csv 内容节选

userId	movieId	rating	timestamp
1	31	2.5	1260759144
1	1029	3	1260759179
1	1061	3	1260759182
2	10	4	835355493
2	17	5	835355681
2	39	5	835355604

表 4-2 movies.csv 内容节选

movieId	title	genres
1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2	Jumanji (1995)	Adventure|Children|Fantasy
3	Grumpier Old Men (1995)	Comedy|Romance
4	Waiting to Exhale (1995)	Comedy|Drama|Romance
5	Father of the Bride Part II (1995)	Comedy

links.csv 包含 movieId、imdbId、tmdbId 三个字段.其中 imdbId 是该电影在电影网站 imdb 的编号，tmdbId 是该电影在电影网站 The Movie Database 的编号，都可以用来唯一识别某部电影.tags.csv 主要包括电影的标签，适合做基于标签的推荐系统，本系统没有用到该文件中的内容.表 4-3 和表 4-4 分别是 links.csv 文件和 tags.csv 文件的内容节选.

表 4-3 links.csv 内容节选

movieId	imdbId	tmdbId
1	0114709	862
2	0113497	8844
3	0113228	15602
4	0114885	31357
5	0113041	11862
6	0113277	949

表 4-4 tags.csv 内容节选

userId	movieId	tag	timestamp
15	339	sandra 'boring' bullock	1138537770
15	1955	dentist	1193435061
15	7478	Cambodia	1170560997
15	32892	Russian	1170626366
15	34162	forgettable	1141391765
15	35957	short	1141391873

另外数据库中还有 moviegenre 数据表，里面包含 imdbId、title(电影名字)、imdb score(imdb 网站的电影评分)、poster(电影海报链接)，通过查询电影 imdbId 可以得到该电影的名字和海报图展示到前端.

4.5.2 数据库逻辑结构设计

本节归纳了用户、电影、推荐列表、用户评分数据表之间的关系.

用户实体属性如图 4-19 所示. 每个用户注册系统的时候都必须填写用户名、邮箱地址，设置用户密码，并且系统会自动为其分配一个 id 号.

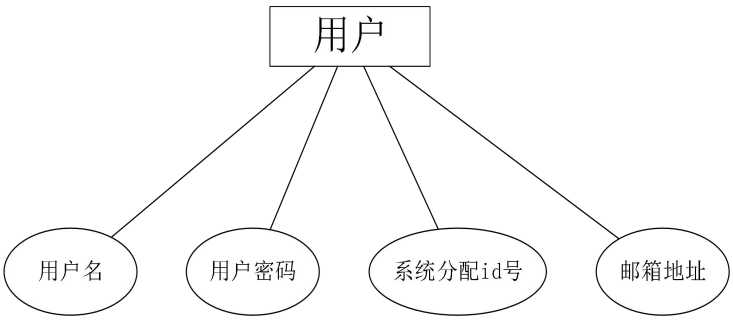

图 4-19 用户 E-R 图

每部电影信息的实体属性如图 4-20 所示.

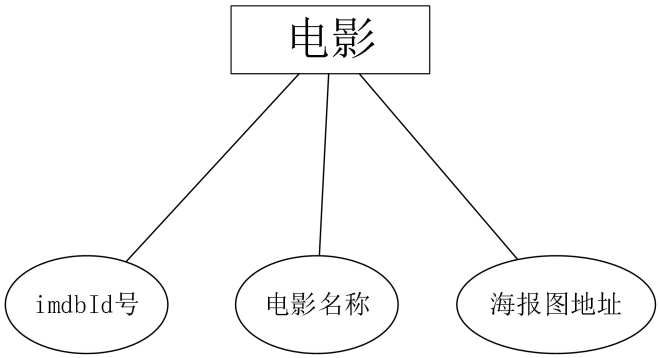

图 4-20 电影 E-R 图

推荐结果列表信息的实体属性如图 4-21 所示. 推荐结果列表中包含电影的海报图地址，这样推荐页面就会展示出电影海报，增加系统美观的同时也提高了用户的可阅读性.

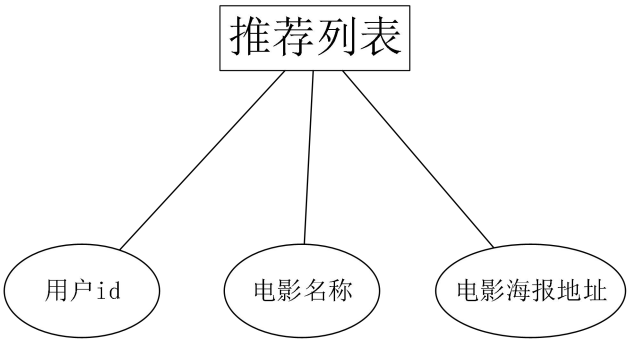

图 4-21 推荐结果 E-R 图

用户对电影的评分实体属性如图 4-22 所示.

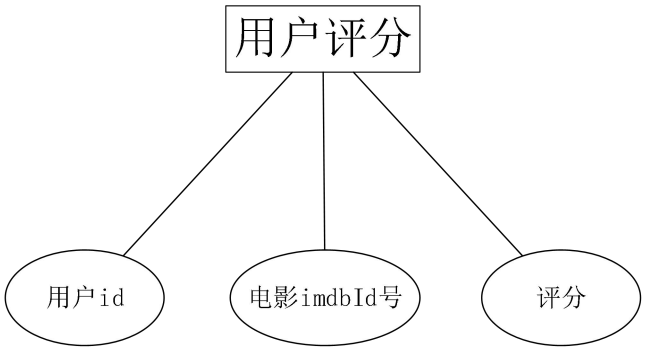

图 4-22 用户评分 E-R 图

4.5.3 系统 E-R 图

数据库模型通常包括层次模型、网状模型、关系模型以及 ER 图（实体-联系图）. 系统的 E-R 图如图 4-23 所示.

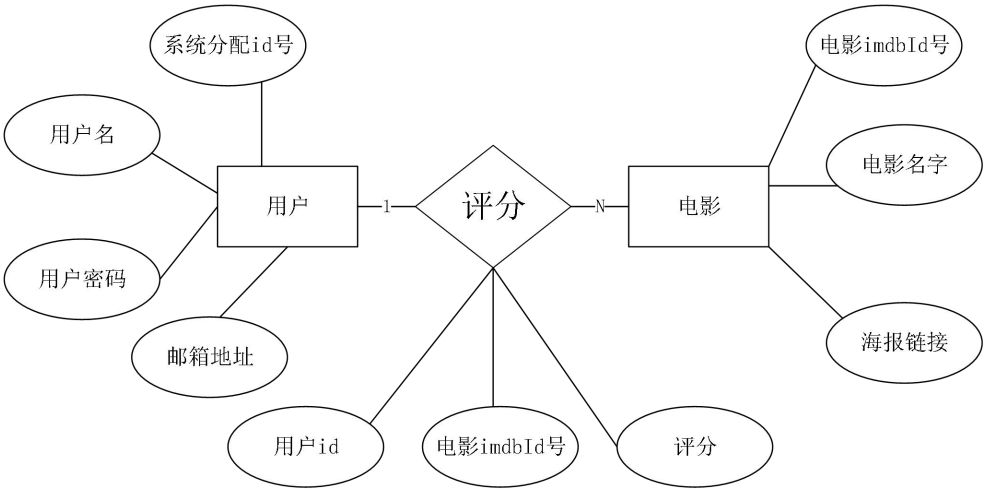

图 4-23 系统 E-R 图

4.5.4 系统数据表设计

本系统采用 MySQL 数据库，数据库中包括用户信息表、电影属性表、电影评分表、电影推荐列表. 下面将介绍主要的存储表的结构.

（1）电影属性表

电影属性表用来保存系统中每部电影的一些属性，包括电影的 imdbId 号、电影名字、电影的海报链接三个字段，数据表的名字为 moviegenre. 表 4-5 即为 moviegenre 表.

表 4-5 电影属性表

字段名称	字段类型	含义	备注
imdbId	int(11)	imdb 网站上电影的编码	主键
title	varchar(300)	电影名字	
poster	varchar(300)	电影海报图的链接地址	

（2）电影评分表

该表所存内容来自 movielens 的 ratings. csv 和 links. csv 文件，这两个文件所含内容已经在 4.5.1 节说明，这里略过. 用 mysql 语句将 ratings.csv 和 links.csv 连接，只保留 userId、imdbId、rating 并存入已经建好的空数据表 users_resulttable 中.这个数据表里的数据就是推荐算法得出推荐结果的依据.算法只有遍历这个数据表，才能计算出登录用户和该表中的哪些用户相似度较高，从而给出推荐结果.电影评分表如表 4-6 所示.

表 4-6 电影评分表

字段名称	字段类型	含义	备注
userId	int(11)	用户的 id 号	
imdbId	int(11)	imdb 网站上电影的编码	
rating	decimal(3,1)	用户的电影评分	
id	int(11)		主键

（3）电影推荐列表

推荐算法遍历 users_resulttable 表之后，会生成目标用户的推荐的电影的 imdbID 号，然后在电影属性表 moviegenre 中查询出电影的名字和海报链接并插入电影推荐列表中. 存海报链接是为了展示页面既有电影名也有电影的海报图，这样就生成了属于登录用户专属的电影推荐列表，表名为 users_insertposter.

系统给出的电影推荐列表包含字段如表 4-7 所示.

表 4-7 电影推荐列表

字段名称	字段类型	含义	备注
id	int(11)		主键
userId	int(11)	用户的 id 号	
title	varchar(60)	电影名字	
poster	varchar(400)	电影海报地址	

表 4-8 为 users_insertposter 数据表部分内容节选.

表 4-8 users_insertposter 数据表内容节选

id	userId	title	poster
1	1002	Pitch Black (2000)	https://images-na.ssl-images-amazon.com/images/M/MV5BNTNmYzE1OWYtZDdjNC00OTdhLTg1YjUtYWJlZTVkMzkzNmVkXkEyXkFqcGdeQXVyMTQxNzMzNDI@._V1_UX182_CR0,0,182,268_AL_.jpg
2	1002	The Transporter (2002)	https://images-na.ssl-images-amazon.com/images/M/MV5BMTk2NDc2MDAxN15BMl5BanBnXkFtZTYwNDc1NDY2._V1_UY268_CR2,0,182,268_AL_.jpg
3	1002	Big Hero 6 (2014)	https://images-na.ssl-images-amazon.com/images/M/MV5BMDliOTIzNmUtOTllOC00NDU3LWFiNjYtMGM0NDc1YTMxNjYxXkEyXkFqcGdeQXVyNTM3NzExMDQ@._V1_UY268_CR3,0,182,268_AL_.jpg
4	1002	Donnie Darko (2001)	https://images-na.ssl-images-amazon.com/images/M/MV5BZWQyN2ZkODktMTBkNS00OTBjLWJhOGYtNGU4YWVkNTY0ZDZkXkEyXkFqcGdeQXVyNjU0OTQ0OTY@._V1_UY268_CR0,0,182,268_AL_.jpg

4.6 本章小结

本章首先深刻分析了系统的需求分析，接着介绍了整个推荐系统的架构和各个功能模块的设计与实现，不仅展示了系统的注册登录界面、电影首页、电影评分板块以及电影推

荐界面，还给出了实现每个功能的核心代码.

同时，本章节也详细介绍了系统数据库的设计，数据库总共包含了用户信息表、电影属性表、电影评分表、电影推荐列表 4 张数据表，展示了各个表所含内容. 并且以 ER 图的形式展示了各个数据表的属性以及其内在联系.

第 5 章 总结与展望

5.1 总结

随着互联网的飞速发展，关于推荐系统的研究工作也蒸蒸日上.本文主要研究了实现推荐系统所需的技术，研究了推荐算法以及推荐系统的一些国内外现状.并以电影推荐系统为实例，对整个推荐系统的架构以及所有功能做了详尽的阐述与分析.从系统的需求分析出发，剖析了用户对功能的需求，然后介绍了系统的各个功能模块，最后充分展示了系统数据库的设计以及数据表的内容.

设计的系统包含用户信息模块、电影展示模块、用户评分模块、推荐结果模块.每个模块的介绍都有流程图或者系统截图.数据库中的核心数据表也充分对所存内容、每个字段的类型进行了阐述.

本系统采用改进过的两种协同过滤算法：基于用户的协同过滤和基于物品的协同过滤，使用户能从两种不同角度获得推荐，提高用户的可选择性.并且利用准确率、召回率以及流行度三个指标对算法进行评测.实验表明，基于本系统所用数据集，UserCF-IIF 算法的性能要优于 ItemCF-IUF 算法.

核心的推荐算法工作完成后，本文结合现有的较热门的 Web 重量级框架 Django，利用 Django 内置的用户注册表单实现用户注册，编写视图代码进行电影推荐结果的 UI 前端展示.利用 CSS3 语言对网页进行排版、美化，使系统看上去更加简洁和美观，增加用户的体验.

5.2 不足之处及未来展望

本文工作完成了一个简单的个性化电影推荐系统，仍然还有很多地方不足需要进一步完善，主要有以下几个方面：

1. 采用的算法效率不够高.当前使用的两种算法依旧是上个世纪提出的算法，所以系统运行效率不高，速度很慢，推荐的物品准确率不高.后期可以考虑抛弃古老的推荐算法，采用更新、效率更高的推荐算法，并结合基于标签和基于内容的推荐算法进行推荐.可以在用户注册时就填写年龄、性别、职业等信息先进行初步的分类给出推荐，再根据评分精准推荐，效果会更好.

2. 本系统使用的是用户对电影的历史评分数据，这些数据都存在一定的稀疏性.登录用户看过的电影可能很少，这样就导致他和数据库中所存用户评过分的电影有较少的交叉项目，使得计算出的相似度很不准确.如何减少数据稀疏带来的困扰是提高推荐系统准确度的又一个出发点.系统采用的是 movielens 网站的较小的数据集，整个数据集压缩之后只有 897kb，所以包含的电影数目也不是很多，推荐的效果大打折扣.之后可以用更大的数据集，考虑遍历百万条评分数据来进行推荐.

3. 本系统电影首页的分类模块，只针对数据集中含有的电影的 8 个类别分别挑选了 16 部电影，导致用户能进行评分的电影数量有限.且没有设计电影搜索功能，用户不能针对性的对自己看过的电影进行评分.

参考文献

[1] Pilli L E, Mazzon J A. Information overload, choice deferral, and moderating role of need for cognition: Empirical evidence[J]. Revista De Administracao Publica, 2016, 51(1):36-55.

[2] 朱杰. 基于标签和协同过滤的图片推荐系统[D]. 天津师范大学，2014.

[3] 王国霞，刘贺平. 个性化推荐系统综述[J]. 计算机工程与应用，2012:66-76.

[4] 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.

[5] Hofmann T. Latent semantic models for collaborative filtering[J].ACM Transactions on Information Systems, 2004, 22(1):89-115.

[6] 杨杰. 个性化推荐系统应用及研究[D].中国科学技术大学，2009.

[7] Ungar L H, Foster D P. Clustering methods for collaborative filtering[C].Proceedings of the AAAI Workshop on Recommendation Systems,Madison, USA, Jul 26-27, 1998. Menlo Park, CA, USA: AAAI, 1997: 114-129 .

[8] Tan Xueqing, He Shan. Research Review on Music Personalized Recommendation System[J]. New Technology of Library and Information Service, 2014, 30(9): 22-32.

[9] Wang X, Rosenblum D, Wang Y. Context-Aware Mobile Music Recommendation for Daily Activities [C]. In: Proceedings of the 20th ACM International Conference on Multimedia. ACM, 2012: 99-108.

[10]徐晨. 基于二部网络分析的推荐算法研究及其应用[D]. 扬州大学，2017.

[11]李小浩．协同过滤推荐算法稀疏性与可扩展性问题研究[D]．重庆大学，2015.

[12]X. Su, T.M. Khoshgoftaar, A survey of collaborative filtering techniques, Adv. Artif. Intell. 2009 (2009) .

[13]喜晶. 个性化推荐技术的分析和比较[J]. 软件研发与应用,2016:39.

[14]Breese J S, Heckerman D, Kadie C. Empirical analysis of predictive algorithms for collaborative filtering[J]. Uncertainty in Artificial Intelligence, 2013, 98(7):43-52.

[15]项亮.推荐系统实践[M].北京：人民邮电出版社,2012 :26.

[16]冯阿敏. 基于用户协同过滤算法的推荐系统的设计与实现[D]. 西安电子科技大学，2017.

[17]张明珺. 基于用户的个性化影视推荐系统的研究与实现[D]. 电子科技大学，2017.

[18]陈诺言. 基于个性化推荐引擎组合的推荐系统的设计与实现[D]. 华南理工大学，2012.

[19]Pazzani M J,J Muramatsu,D Billsus. Syskill & Webert:Identifying interesting web sites[C]. Proceedings of the national conference on artificial intelligence,1996:54-61.

[20]尤方圆. 电影推荐系统的设计与实现[D]. 华中科技大学,2013.

[21]张兵. 空气质量模型的研究与应用[D]. 中国地质大学(北京),2017.

[22]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.

[23]王玉业. 基于协同过滤的个性化推荐研究[D]. 江苏大学，2017.

[24]余文丽. 基于 Android 的教学信息管理系统的设计与实现[D]. 华中师范大学，2015.

[25]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.

致　谢

从 17 岁的雨季到 21 岁的花季，我在江南大学度过了我人生中最青春的四年.转眼间，这四年的本科生生活就要结束了，我希望能给它画上一个圆满的句号.

在这四年的时光里，我得到许多老师以及同学的帮助.在论文完稿之际，我想借此机会向所有给予我支持和鼓励的人表达我的谢意.

首先，我要感谢我的导师詹千熠老师对我的指导.从开题报告、中期答辩到现在的定稿，詹老师都认真负责，全心全意的辅导我，提出改进意见.詹老师的敬业态度以及对学术的钻研精神，是值得我学习的榜样.

其次，我要感谢我的父母，是他们一直在背后默默支持我，鼓励我，才能让我更加安心的投入到学习中，才能顺利完成本科生学业.

同时，我还要感谢同窗四年的大学同学所给予我生活和学习上的帮助，正是有你们的帮助，才使我在陌生的环境中有了家的归属感.

最后，我要感谢数字媒体学院的全体老师，感谢他们的辛勤付出，使我学习到很多知识，夯实了专业技能.

感谢各位评审老师，感谢你们在百忙之中对我的论文进行评审.

附录 A： 作者在校期间发表的论文

[1] 伍静，刘德丰，张松等. 智能摔倒检测监控系统设计[J]. 计算机技术与发展，2018：6-10.

附录 B： 代码

@views.py

```
def get_conn():
    conn = pymysql.connect(host='127.0.0.1', port=3307, user='root', passwd='aptx4869.', db='haha', charset='utf8')
    return conn

def read_mysql_to_csv(filename,user):
    with codecs.open(filename=filename, mode='w', encoding='utf-8') as f:
        write = csv.writer(f, dialect='excel')
        conn = get_conn()
        cur = conn.cursor()
        cur.execute('select * from users_resulttable')
        rr = cur.fetchall()
        for result in rr:
            write.writerow(result[:-1])

def register(request):
    # 只有当请求为 POST 时，才表示用户提交了注册信息
    if request.method == 'POST':
        form = RegisterForm(request.POST)
        # 验证数据的合法性
        if form.is_valid():
            # 如果提交数据合法，调用表单的 save 方法将用户数据保存到数据库
            form.save()
            # 注册成功，跳转回首页
            return redirect('/')
    else:
        # 请求不是 POST，表明用户正在访问注册页面，展示一个空的注册表单给用户
        form = RegisterForm()
    return render(request, 'users/register.html', context={'form': form})

def index(request):
    return render(request, 'users/..//index.html')

def recommend1(request):
    Insertposter.objects.filter(userId=USERID).delete()
    #selectMysql()
    read_mysql_to_csv('users/static/users_resulttable.csv',USERID)  #追加数据，提高速率
    ratingfile2 = os.path.join('users/static', 'users_resulttable.csv')
    usercf = UserBasedCF()
    #userid = '1001'
    userid = str(USERID)#得到了当前用户的 id
    print(userid)
    usercf.generate_dataset(ratingfile2)
    usercf.calc_user_sim()
```

```
usercf.recommend(userid)       #得到 imdbId 号
try:
      conn = get_conn()
      cur = conn.cursor()
      #Insertposter.objects.filter(userId=USERID).delete()
      for i in matrix:
            cur.execute('select * from moviegenre3 where imdbId = %s',i)
            rr = cur.fetchall()
            for imdbId,title,poster in rr:
                  #print(value)              #value 才是真正的海报链接
                  if(Insertposter.objects.filter(title=title)):
                        continue
                  else:
                        Insertposter.objects.create(userId=USERID, title=title, poster=poster)
     finally:
      conn.close()
  results = Insertposter.objects.filter(userId=USERID)
return render(request, 'users/movieRecommend.html',locals())

def recommend2(request):
      Insertposter.objects.filter(userId=USERID).delete()
      read_mysql_to_csv('users/static/users_resulttable2.csv',USERID)   #追加数据，提高速率
      ratingfile2 = os.path.join('users/static', 'users_resulttable2.csv')
      itemcf = ItemBasedCF()
      userid = str(USERID)#得到了当前用户的 id
      print(userid)
      itemcf.generate_dataset(ratingfile2)
      itemcf.calc_movie_sim()
      itemcf.recommend(userid)       #得到 imdbId 号
      try:
            conn = get_conn()
            cur = conn.cursor()
            #Insertposter.objects.filter(userId=USERID).delete()
            for i in matrix:
                  cur.execute('select * from moviegenre3 where imdbId = %s',i)
                  rr = cur.fetchall()
                  for imdbId,title,poster in rr:
                        #print(value)              #value 才是真正的海报链接
                        if(Insertposter.objects.filter(title=title)):
                              continue
                        else:
                              Insertposter.objects.create(userId=USERID, title=title, poster=poster)
     finally:
            conn.close()
            results = Insertposter.objects.filter(userId=USERID)
            Insertposter.objects.all()
   return render(request, 'users/movieRecommend2.html',locals())
```