某某大学

**《工程项目实习》课程报告**

|  |  |  |
| --- | --- | --- |
| **系（院）：** | 计算机与软件工程学院 | |
| **班 级：** |  | |
| **题 目：** | 基于Django的电影推荐系统 | |
| **组 名：** |  | |
| **组 长：** |  |  |
| **组 员：** |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| **任课教师：** |  | |

**学年学期:** 2022 - 2023 学年 第 1 学期

2022 年 12 月 31 日

目 录

[1 团队建设 1](#_Toc91669940)

[1.1 团队的任务和目标 1](#_Toc91669941)

[1.2 团队制度 1](#_Toc91669942)

[1.3 团队任务分配 1](#_Toc91669943)

[1.4 项目进度安排 1](#_Toc91669944)

[2 需求分析 2](#_Toc91669945)

[2.1 国内外研究现状 2](#_Toc91669946)

[2.2 研究背景与意义 3](#_Toc91669947)

[2.3 功能分析和性能分析 5](#_Toc91669948)

[2.4 用例图 8](#_Toc91669949)

[2.5 类图 10](#_Toc91669950)

[2.6 系统E-R图 10](#_Toc91669951)

[3 推荐算法 12](#_Toc91669952)

[3.1 基于用户的协同过滤算法 12](#_Toc91669953)

[3.2 基于物品的协同过滤算法 13](#_Toc91669954)

[3.3 相似度计算 13](#_Toc91669955)

[4 总体设计 14](#_Toc91669956)

[4.1 设计原则 14](#_Toc91669957)

[4.2 系统架构 15](#_Toc91669958)

[4.3 功能模块 18](#_Toc91669959)

[4.4 系统表结构 19](#_Toc91669960)

[4.5 数据字典 21](#_Toc91669961)

[5 使用手册 27](#_Toc91669962)

[5.1 使用环境配置 27](#_Toc91669963)

[5.2 数据爬取及存储 29](#_Toc91669964)

[5.3 结果展示 31](#_Toc91669965)

[6 代码清单 37](#_Toc91669966)

[6.1 数据爬虫代码 37](#_Toc91669967)

[6.2 动态爬虫抓取链接代码 42](#_Toc91669968)

[6.3 抓取电影详细信息代码 43](#_Toc91669969)

[6.4 连接数据库代码 46](#_Toc91669970)

[6.5 将数据写入数据库 49](#_Toc91669971)

[6.6 推荐算法代码 52](#_Toc91669972)

[6.7 电影图文展示代码 59](#_Toc91669973)

[6.8 前端代码 61](#_Toc91669974)

[总 结 91](#_Toc91669975)

[参考文献 92](#_Toc91669976)

# 1 团队建设

## 1.1 团队的任务和目标

总体任务：完成对豆瓣电影推荐网页的爬虫，并在此基础上完成推荐电影的系统。

子任务：

（1）对爬取数据进行预处理使用；

（2）使用对数据进行分析；

（3）使用Django完成对系统的设计；

（4）完成文档编写。

目标：

（1）最大程度的完成预期任务，尽全力完善细节，完美实现既定的功能，满足用户需求；

（2）不仅要巩固基础知识，还要学习较难知识，为完成项目做好充分的准备，也为自身求职就业增加竞争力。

## 1.2 团队制度

（1）实训课准时到教室上课，无特殊情况不可请假；

（2）按时完成老师、组长布置的任务，不拖延；

（3）遇到问题时，相互交流，团结合作。

## 1.3 团队任务分配

（1） ：

团队工作协作，代码督办；

部分代码编写及团队建设、需求分析和总文档的编写。

（2） 、 、 ：

后期项目编写及代码测试。

（3） ：

部分团队文档编写及全程测试。

## 1.4 项目进度安排

小组进度安排表

|  |  |
| --- | --- |
| 时间 | 进度安排 |
| 2022/11/22—2022/11/28 | 完成团队建设文档编写 |
| 2022/11/29—2022/12/05 | 完成需求分析文档的编写 |
| 2022/11/06—2022/12/12 | 完成对豆瓣电影数据的爬取、预处理及分析 |
| 2022/12/13—2022/12/19 | 完成对系统设计的代码编写 |
| 2022/12/20—2022/12/26 | 完成对项目系统的总体测试并加以改进 |
| 2022/12/27—2022/12/19 | 完成对前期文档的汇总及总报告的编写 |

# 2 需求分析

现在电影资源[1]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。所以本系统旨在为每一个用户推荐与其兴趣爱好契合度较高的电影。本系统包含电影前端展示界面、电影评分、电影收藏、推荐算法的实现。其中实现推荐算法是整个电影推荐系统的核心。

## 2.1 国内外研究现状

推荐系统被提出并受到大众的关注已有二十多年了。在国外，Goldbery [2]等人员于 1992 年在 Tapestry 垃圾邮件过滤系统中运用了协同过滤思想。2006 年 Netflix[3]举办了改进推进系统大赛，然而这场比赛也催生出了许多其他优质的推荐算法，例如比较有名的当属隐语义模型和矩阵分解。它们为后续的推荐系统发展奠定了重要的理论依据。

在国内，文献对目前比较流行的框架进行分析，Netflix Prize 竞赛提出了基于矩阵分解的二项矩阵分解模型。文献[4]提出了在相似度计算中融入项目类别喜爱度和类别关注度，再对用户进行聚类，最后做出推荐，不仅时间复杂度变低，而且也使得推荐精度进一步升高。

综合来看，推荐系统主要是从优化推荐算法以及推荐算法与实际工程应用相结合等两个方面进行研究，目前常用的有：

（1）基于协同过滤算法的推荐。它是以用户的历史行为信息以及最近邻居为基础进行推荐，也是较早被广泛关注和发展的算法。在某一特定用户或物品的集合内，求出它们的相似度，并获取各自的邻居集合。然后把用户没有使用过的物品推荐给他，同时也去除用户曾经使用过的物品，以避免重复推荐，从而完成整个推荐过程。一些商业系统中，在用户物品信息量比较大的情形下，用户间直接有共同行为的物品数据非常少，会造成共同行为数据比较稀疏的问题。同时，在新用户或者新物品进入业务系统后，由于没有历史行为也会造成冷启动的问题，对此提出了协同聚类算法。协同过滤发展的历史最悠久，到现在依然普遍适用，充分的体现出它的健壮性，而且还得到了普遍的应用，亚马逊、淘宝、京东、爱奇艺等流行的网站中都包含了基于该思想的推荐算法。

（2）基于内容的推荐。从信息检索领域发展出基于内容的推荐方法。主要是通过预先提取物品自身的属性特征，然后根据它们的属性差异来作为数据源进行推荐的。首先为每个用户建立符合自身的特征向量，然后通过这些特征向量计算出他们各自的相似性，再通过相似性筛选出用户各自的邻居集合。因此，这种推荐方式不需要考虑其他用户行为的信息，也即不存在考虑用户对物品评价的因素。正因如此，它的这一优点克服了在推荐过程中会常常遇到的冷启动问题，而且对于数据稀疏方面上也能得到一定程度的缓和，从而实现推荐系统的个性化。

（3）基于关联规则的推荐。关联规则有个“啤酒和尿布”的小故事案例，但是也特别的经典，常常被拿来用作分析关联规则算法的分析，其思想是从数据集中找到项与项的关系，简而言之的理解就是用户购买某个物品的时候，同时也去购买另一个物品的概率。它是对于最大频繁项的求解，通常有Apriori和FP-Growth使用较多。

（4）基于聚类的推荐算法。基于聚类的算法有K最近邻算法，它可以按照用户或物品划分，对于专门来缓解数据稀疏有非常很好的作用。

## 2.2 研究背景与意义

从信息匮乏到信息过载，人们的生产生活遇到了巨大的挑战。社会各行业均以数据作为研究对象，与此同时也累积了大量具有潜在价值的数据，现今已步入大数据时代。我们进入各种搜索引擎网站亦或是APP[5]，在输入关键信息后检索想要得的信息。尽管我们可以通过搜索引擎来获取我们想要的信息，但是对于很多人来说，却并不明确自己需要什么具体的信息。例如当人们进入超市或商店进行购物时，并不太明确自己一定就做好了买什么的准备，而是去了货架、储物柜才知道。通过分析自身的喜好和需求，最终决定买什么。我们身处于一个信息爆炸的时代，让用户在有限的时间和无限的海量信息中去找所需要的信息，就像是大海捞针，这就是大家经常所称的信息过载现象。推荐系统能够为客户提供推荐服务，现在越来越多的科研工作者投入到此领域当中。如何能让这些海量信息自己主动“找上门”，从以前的主动寻找所需物品，变为被动的去接受。因此，这样也就形成了推荐系统的雏形。

现如今大数据越来越火，在生活中的应用涉及到方方面面，相应的分布式存储系统的技术发展也让存储的运行成本控制的很低。分布式计算是将存储在不同服务节点中的海量数据进行运算，达到降低系统对硬件的要求，使得低廉的机器也能满足这种规模方式的计算量，进而发掘出巨大的潜在价值。数据挖掘[6-7]是跨越多门学科的计算机技术，从海量信息中通过预先定义好的方式去寻找所需要的有价值的信息，再通过一定的转换方式格式化为可以理解的信息。机器学习是通过分析大量的用户历史行为数据从中寻找出其背后所隐藏的规律。大数据处理框架比较早的非 Hadoop莫属，其核心技术 HDFS、MapReduce等更是经典。它能将分布于不同机器上的进程协同进行工作，从而形成一个庞大的分布式应用系统，也被许多的软件厂商青睐。为了在海量数据中寻找用户所需的信息，使用传统的寻找方式需要付出巨大的成本，正是在这样的大背景下推荐系统便出现了。它可以让信息更加精准的流向所需要的用户，用户也无需绞尽脑汁在庞大的数据量中毫无头绪的搜集信息。

从刚开始单一化的发展，到现在的多元化发展与应用，在许多行业领域推荐系统都有它的身影，在像淘宝，京东，拼多多等电商网站中显得尤为凸出，为他们的销售业绩作出了巨大的贡献。同样在电影、新闻媒体等领域中，传统的租碟片、看报纸的方式显得越来越难以为继，效率非常低下。比如，在视频领域中国内比较热门的有优酷、爱奇艺等网站都是大家所熟知的。当用户打开网站或者 APP 首页，可以看到各种板块的推荐影片信息，会让用户感到一种很熟悉的感觉，似乎这些 APP 非常智能懂得用户喜好。虽然推荐系统经历了长时间的发展，但其性能仍有一定的提升空间，如何给用户提供高质量的信息推荐，也是目前推荐领域行业大家所一直研究的重难点。在多元化的大信息量的前提下，设计一个满足所有用户需要的高效推荐系统，从而达到具有差异性的个性化，这正是本文所需要研究的。

## 2.3 功能分析和性能分析

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。电影推荐系统能给用户带来便利.本系统要实现的是一个面向用户的个性化电影推荐系统。根据豆瓣 Top250 的电影数据集，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。本系统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）良好的用户交互体验：包括用户注册登录在内的整个 Web 系统，用户可查看电影，收藏电影；

（4）系统要易于扩展和维护。

2.3.1 功能分析

由于国内外电影行业的不断发展，每天都有新加入的观影用户和新上架的影片。如何让用户在规模如此巨大的影片中找到自己喜欢的电影，就必须让系统从被动接收用户请求转化为主动感知用户的信息需求，从而节约用户的时间、增加系统的粘性。本文所设计的电影推荐系统就是以此为目的，让系统主动为用户寻找喜欢的电影。

用户可以分为有目标性的和无目标性的，对于有目标性的用户，他们知道自己想看某部电影，可直接根据用户提供的电影名字进行查询或者模糊匹配寻找电影；对于无目标性的用户，可根据用户静态数据与用户历史行为数据进行数据分析挖掘用户可能喜欢的电影，让系统自动给用户进行推荐预测。

本文设计的电影推荐系统是面向用户的，具有前台业务系统和后台推荐系统。由于推荐系统是依据用户的个性化数据去做推荐，所以用户必须先注册账号，登录成功后才会根据用户的个性化数据自动展示实时推荐、离线推荐、热门推荐以及详情页推荐模块的推荐结果。当用户发生点击事件时，系统还会对该数据进行记录，以此为下次推荐做准备，让系统形成一个闭环。

根据系统的特点，可将功能分为基本功能、个性化推荐功能和数据存储功能三个部分。其中基本功能为用户的基本行为功能，包括用户注册、用户登录、电影搜索、电影评分、历史评分和退出登录功能。个性化推荐功能主要由推荐引擎提供支持，包含离线推荐、实时推荐、热门推荐和详情页推荐四个模块。数据储存[8]功能主要为用户历史行为数据存储和推荐引擎的计算提供支持，使用数据库完成。

（１）基本功能

1）用户注册：本系统只对会员开放，用户登录成功后，方能访问。并且系统

根据用户唯一标示进行电影推荐，所以对新用户提供注册功能。用户进行账号注册时系统会在网页中显示填写账号、登录密码和确认密码的输入框，之后系统会验证用户提交的信息是有效，并返回是否注册成功，注册成功则返回登录页面。

2）用户登录：用户在登录页的相应输入框填写用户名和密码，后台系统接收到用户填写的信息后会进行校验。校验通过，则自动跳转到推荐首页，否则将会提不登录失败。

3）电影搜索：用户在页面顶部的搜索框内输入想要查询电影的关键词信息，系统则会根据用户提供的信息在电影信息里进行查找，将匹配到的电影呈现给用户。

4）电影评分：用户进入电影详情页后，可根据自己的喜好对该电影进行评分。评分值的范围为0-5分，0分为最低分，5分为最高分，以0.5分为一个评分值间隔。

5）历史评分：用户可对自己有过评分记录的电影进行查看，页面会展示相关电影与评分值。

6）退出登录：当用户点击“退出登录”后，则会退出系统，跳转到登录页面。若还想进入系统，就得再次登录。

（２）推荐功能

本电影推荐系统的推荐场景有四种，既有个性化推荐功能，也有非个性化推荐功能。个性化推荐会根据每个用户的不同偏好，为其提供符合各种喜好的影片。非个性化推荐根据所有用户对电影的历史行为，提供评分次数高、评分值高等的相同推荐结果。为了提高用户的交互体验，还提供了实时推荐功能，该推荐模块可在秒级时间内响应推荐需求。

1）离线推荐：离线推荐位于推荐首页，根据用户、电影和上下文等相关信息进行电影推荐，每天只需提供一次推荐结果，推荐的结果具有个性化特点。其推荐的准确率、召回率和覆盖率应有良好的得分值，其是用户对系统粘性的重要保障。

2）实时推荐：实时推荐位于推荐首页，它会根据用户对不同电影的评分，实时的更新此推荐模块的结果。这部分注重推荐效果的实时性，是对当前评分电影的同类型电影进行推荐，必须让系统快速响应用户。

3）热门推荐：热门推荐位于推荐首页，是非个性化的推荐，其对每个用户的推荐结果都一样。这部分的推荐是根据所有用户对电影的历史评分行为进行统计后获得，统计的方式有所有电影中评分次数最多的电影、所有电影中平均得分值最高的电影等。

4）详情页推荐：详情页推荐位于电影详情页，为了避免推荐同类型电影导致用户审美疲劳，此推荐部分不根据电影相似度进行推荐，而是采用基于模型的协同过滤方式进行推荐。

（３）数据存储功能

电影推荐系统应具备电影相关数据、用户相关数据和计算中间结果存储的能力。对于非实时性需求的数据，应采用可扩展、稳定性高的存储平台功能进行保存，例如对于用户历史行为的备份数据。对于实时性需求高的数据，则采用基于内存的软件，以保证其处理速度，例如实时推荐需要的评分数据、供展示层调用的推荐结果。

2.3.2 性能分析

（1）准确性

准确性是推荐系统的重要性能度量指标，拥有较高的准确性从某种程度上预示着模型能很好地预测用户的兴趣。因此，系统如果有较高的准确性，那么就能更加准确的给用户推荐其真正喜欢的电影，提升用户对推荐功能的体验，增强用户对系统的粘性。

（2）实时性

当用户对一电影进行评分后，会触发推荐引擎的实时推荐计算，系统需马上计算出实时推荐结果集，实时推荐耗时越短越能给用户更加的体验。本课题的实时推荐模块在接收到用户对电影的评分数据后，到计算出实时推荐结果集的时间需要低至秒级。

由于定位的用户群体规模较大，所以本系统对性能以及兼容性要求比较严格，

（1）满足大量用户同时登录系统并能成功地运行；

（2）简单人性化的操作页面，易于用户操作，也便于后期维护；

（3）因为系统对数据库依赖性较强，所以数据库需要具备相当高的稳定性和高效性。

## 2.4 用例图

用例的核心概念中首先它是一个业务过程，经过逻辑整理抽象出来的一个业务过程，这是用例的实质。业务过程是指在待开发软件所处的业务领域内完成特定业务任务的一系列活动。

我们小组所设计的推荐系统主要服务于用户和管理员。

2.4.1 游客和普通用户用例图

从用户角度来看，分为游客和普通用户：游客只能浏览电影信息，而且需要用户登录才可以对电影进行评分，评论等操作，所以游客无法获得个性化定制的推荐电影列表。

对于普通用户，首先需要新用户在系统当中注册user\_id，便于系统能够准确识别用户信息，并根据该用户的观影兴趣为其提供独特的推荐列表。

用例图可视化[9]地表达了系统的需求，具有直观、规范等优点，克服了纯文字性说明的不足。借助于用例图，系统用户、系统分析人员、系统设计人员、领域专家能够以可视化的方式对问题进行探讨，减少了大量交流上的障碍，便于对问题达成共识。现通过用例图展现用户与系统之间的交互，游客和普通用户的用例图如下：

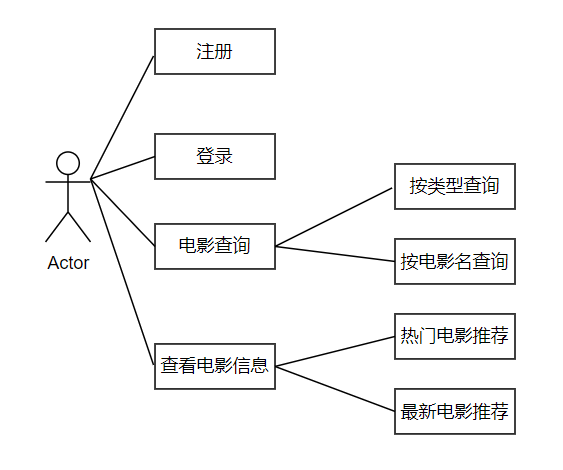


图2.1 游客用例图

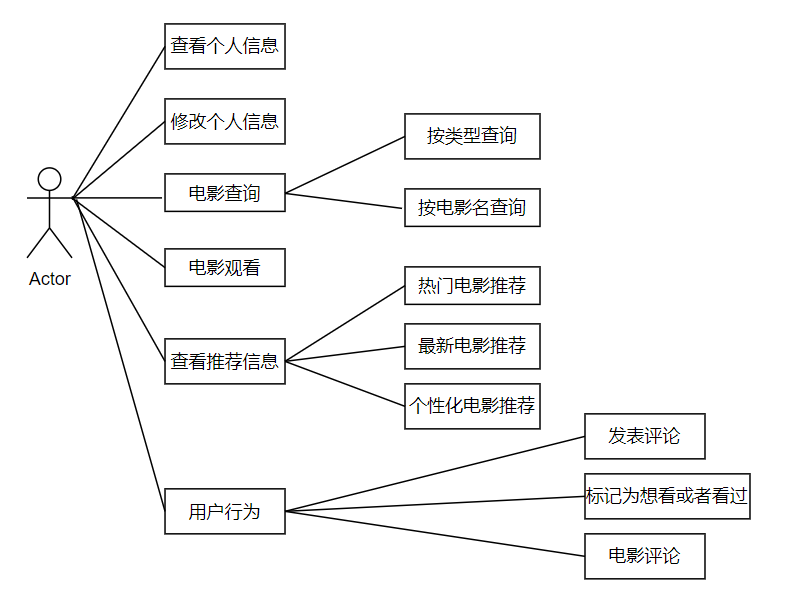


图2.2 普通用户用例图

2.4.2 管理员用例图

从管理员的角度来看，管理员在系统中具有最高权限，其主要是对用户信息以及电影信息进行维护与管理操作。管理员可以随时添加新电影，以及对电影信息进行删除，更新修改以及检索等，管理员还可以添加、删除、查找用户信息，还可以对用户行为进行管理。



图2.3 管理员用例图

## 2.5 类图

所设计的电影推荐系统中的类可以大致分为以下几种：Person，User，Administrator, Movies, Rating, Interest, Recommendation。

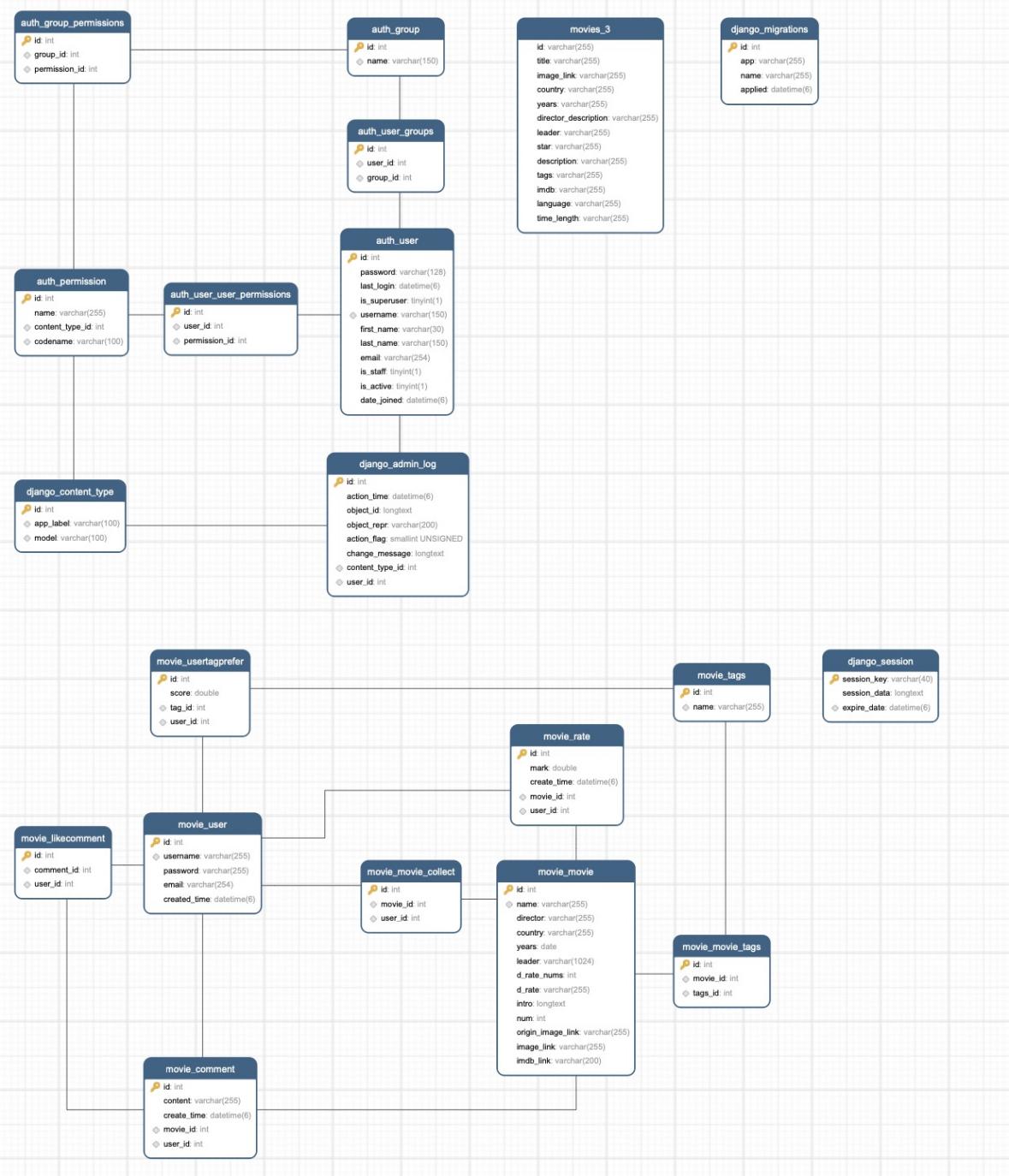
Person是Administrator和User的父类；Administrator负责管理用户和电影信息；User可以设置自己的观影兴趣并且给电影评分；Movies主要是进行电影信息的管理和更新电影操作；Recommendation系统则是根据用户的兴趣和它对电影的评分，为用户生成个性化的推荐。总体类图如下：

图2.4 总体类图

## 2.6 系统E-R图

按照数据组织模型[10-11]来组织显示信息的过程需要花费大量的资源，另外设计的工作也是很烦乱，这样造成了工作效率的低下。而E-R方法就很容易来解决这种问题，实体-关系方法简称为E-R方法，此方法所使用的工具为E-R图，实体、实体属性以及实体间关系就组成了E-R图。

本系统E-R图如图2.5所示。共有4个实体，分别是会员、管理员、电影以及电影分类。一个会员可以查询多个电影和电影分类，一个管理员管理多个电影和电影分类。

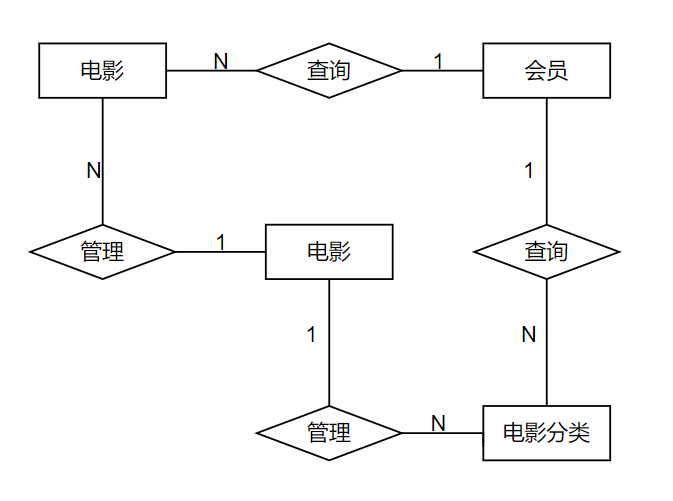


图2.5 系统E-R图

其中每个实体的属性图如下所示。

管理员实体属性图如图2.6所示。管理员具有编号、账号、密码属性。

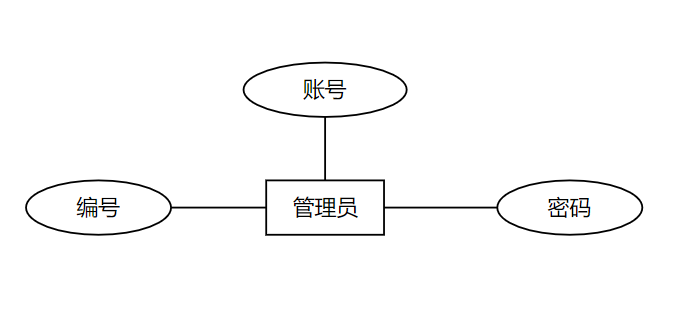


图 2.6 管理员实体属性图

电影实体属性图如图2.7所示。电影具有标题、图片、时间、内容属性。

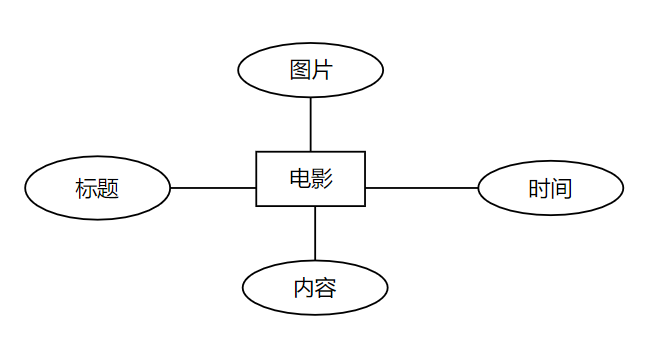


图 2.7 电影实体属性图

电影分类实体属性图如图4.6所示。电影分类具有编号和分类名属性。

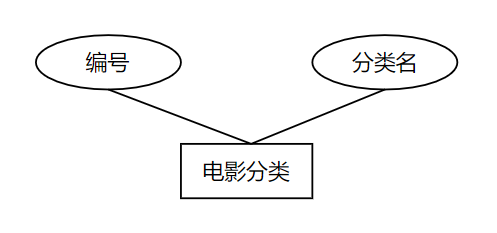


图 2.8 电影分类实体属性图

# 3 推荐算法

## 3.1 基于用户的协同过滤算法

该算法主要用相似统计的方法得到具有相似爱好或者兴趣的相邻用户。基于用户（User-Based）的协同过滤算法[12-13]首先要根据用户历史行为信息，寻找与新用户相似的其他用户；同时，根据这些相似用户对其他项的评价信息预测当前新用户可能喜欢的项。给定用户评分数据矩阵 R，基于用户的协同过滤算法需要定义相似度函数 s：U×U→R，以计算用户之间的相似度[14]，然后根据评分数据和相似矩阵计算推荐结果。

具体步骤如下：第一步，收集用户信息。收集可以代表用户兴趣的信息。一般的网站系统使用评分的方式或是给予评价，这种方式被称为“主动评分”。另外一种是“被动评分”，是根据用户的行为模式由系统代替用户完成评价，不需要用户直接打分或输入评价数据。电子商务网站在被动评分的数据获取上有其优势，用户购买的商品记录是相当有用的数据。第二步，最近邻搜索(Nearest neighbor search, NNS)。以用户为基础（User-based）的协同过滤的出发点是与用户兴趣爱好相同的另一组用户，就是计算两个用户的相似度。例如：查找 n 个和 A 有相似兴趣用户，把他们对 M 的评分作为 A 对 M 的评分预测。一般会根据数据的不同选择不同的算法， 目前较多使用的相似度算法有 Pearson Correlation Coefficient（皮尔逊相关系数）、Cosine-based Similarity（余弦相似度）、Adjusted Cosine Similarity（调整后的余弦相似度）。

## 3.2 基于物品的协同过滤算法

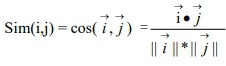
该以物品为基础的协同过滤方法有一个基本的假设“能够引起用户兴趣的项目，必定与其之前评分高的项目相似”，通过计算物品之间的相似性来代替用户之间的相似性[15-18]。

具体步骤如下：第一步，收集用户信息。同以用户为基础（User-based）的协同过滤。第二步，针对项目的最近邻搜索。先计算已评价项目和待预测项目的相似度，并以相似度作为权重，加权各已评价项目的分数，得到待预测项目的预测值。例如：要对项目 A 和项目 B 进行相似性计算，要先找出同时对 A 和 B 打过分的组合，对这些组合进行相似度计算。第三步，产生推荐结果。以物品为基础的协同过滤不用考虑用户间的差别，所以精度比较差。但是却不需要用户的历史数据，或是进行用户识别。对于项目来讲，它们之间的相似性要稳定很多，因此可以离线完成工作量最大的相似性计算步骤，从而降低了在线计算量，提高推荐效率，尤其是在用户多于项目的情形下尤为显著。

## 3.3 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（3-1）为计算其余弦相似度的公式。

(3-1)

2.欧氏距离

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（3-2）所示。

(3-2)

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[19]，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。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，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（3-3）所示，



(3-3)

4.曼哈顿距离

在平面上，坐标(1x,1y)与坐标(1y,2y)的曼哈顿距离为：

(3-4)

# 4 总体设计

## 4.1 设计原则

（1）实用性原则

任何项目的开发都应该是按照实际实用性来执行的，既要考虑到开发的成本，也要保证项目最后的成果。过多的开发成本，虽然完成了项目需求，但是浪费了大量人力物力以及其他资源。所以按照具体的业务水平、管理能力、以及数据的处理方面来保证项目的实用性是十分重要的。

（2）可扩展性与可维护性原则

一个系统开发出来之后，并不是一层不变的，系统需要随着时间而不断地需要扩展和维护来满足增长的需求。所以在设计软件的时候要最大程度上模块化设计，灵活运用组件，这样在未来需要扩展系统的时候不至于将系统全盘翻新，而是一个模块地进行扩展，这样既节约了资源也更容易调整。

（3）安全可靠性原则

系统的开发都需要涉及到安全性问题，系统对不正当访问的限制以及系统开发过程中对数据库数据的及时备份都是十分重要的。

（4）用户界面设计原则

主要还是考虑到用户的使用体验，系统的界面应该简洁、易于操作，这样既能让用户乐于使用系统，也节省了开发人员在界面设计上花费的资源。

（5）数据库设计原则

数据库设计体现需要体现以下四个原则：设计的一致性、数据库的完整性、数据库的安全保障性、数据库的可伸缩性和设计的规范化。

## 4.2 系统架构

4.2.1 Bootstrap3 CSS框架

前端界面采用 Bootstrap3 css 框架。Bootstrap 是基于 HTML、CSS、JavaScript 的，它简洁灵活，使得 Web 开发更加快捷。Bootstrap 提供了一个带有网格系统、链接样式、背景的基本结构。自带以下特性：全局的 CSS 设置、定义基本的 HTML 元素样式、可扩展的 class，以及一个先进的网格系统。

Bootstrap 还包含了十几个可重用的组件，用于创建图像、下拉菜单、导航、警告框、弹出框等等。其中包含了十几个自定义的 JQuery 插件，便于使用者直接包含所有的插件，也可以逐个包含这些插件。还支持定制 Bootstrap 的组件、Less 变量和 JQuery 插件来得到自己的版本。

4.2.2 MVC框架

（1）MVC模式

MVC英文全称是Model View Control，是模式—视图—控制器的缩写，是一种框架模式，数据、逻辑、界面的代码在MVC模式中获得了很好的分离，三个部分之间有条不紊的协调工作。MVC开始是存在于桌面程序中的，M是指业务模型，V是指用户界面，C则是控制器，使用MVC的目的是将M和V的实现代码分离，从而使同一个程序可以使用不同的表现形式。比如一批统计数据可以分别用柱状图、饼图来表示。C存在的目的则是确保M和V的同步，一旦M改变，V应该同步更新。

MVC指MVC模式的某种框架[20]，它强制性地使应用程序的输入、处理和输出分开。使用MVC应用程序被分成三个核心部件：模型、视图、控制器。它们各自处理自己的任务。最典型的MVC就是JSP + Servlet + JavaBean的模式。

MVC框架的核心思想是解耦，让不同的代码块之间降低耦合，增强代码的可扩展性和可移植性，实现向后兼容。

（2）MVC架构优点

1）耦合性低

视图层和业务层分离，这样就允许更改视图层代码而不用重新编译模型和控制器代码，同样，一个应用的业务流程或者业务规则的改变只需要改动MVC的模型层即可。因为模型与控制器和视图相分离，所以很容易改变应用程序的数据层和业务规则。

2）重用性高

MVC模式允许使用各种不同样式的视图来访问同一个服务器端的代码，因为多个视图能共享一个模型，它包括任何WEB（Http）浏览器或者无线浏览器（Wap），比如，用户可以通过电脑也可通过手机来订购某样产品，虽然订购的方式不一样，但处理订购产品的方式是一样的。由于模型返回的数据没有进行格式化，所以同样的构件能被不同的界面使用。

3）部署快，生命周期成本低

MVC使开发和维护用户接口的技术含量降低。使用MVC模式使开发时间得到相当大的缩减，它使程序员（Java开发人员）集中精力于业务逻辑，界面程序员（HTML和JSP开发人员）集中精力于表现形式上。

4）可维护性高

分离视图层和业务逻辑层也使得Web应用更易于维护和修改。

（3）MVC架构缺点：

1）完全理解MVC比较复杂。

由于MVC模式提出的时间不长，加上同学们的实践经验不足，所以完全理解并掌握MVC不是一个很容易的过程。

2）调试困难。

因为模型和视图要严格的分离，这样也给调试应用程序带来了一定的困难，每个构件在使用之前都需要经过彻底的测试。

3）不适合小型，中等规模的应用程序

在一个中小型的应用程序中，强制性的使用MVC进行开发，往往会花费大量时间，并且不能体现MVC的优势，同时会使开发变得繁琐。

4）增加系统结构和实现的复杂性

对于简单的界面，严格遵循MVC，使模型、视图与控制器分离，会增加结构的复杂性，并可能产生过多的更新操作，降低运行效率。

5）视图与控制器间的过于紧密的连接并且降低了视图对模型数据的访问

视图与控制器是相互分离，但却是联系紧密的部件，视图没有控制器的存在，其应用是很有限的，反之亦然，这样就妨碍了他们的独立重用。依据模型操作接口的不同，视图可能需要多次调用才能获得足够的显示数据。对未变化数据的不必要的频繁访问，也将损害操作性能。

（4）MVC三部分关系

首先等待用户输入，待用户输入后通过控制器进行人机交互，从而将用户数据及指令传递给模型。接着通过模型分别进行业务逻辑判断及数据库存取和根据业务逻辑选择不同视图。接着通过视图与用户人机交互，将结果返回给用户。最后用户获得反馈并进行下一步同样操作。以此反复。

程序的输入操作、程序的输出操作和处理操作被MVC很好的分开来。MVC三部分关系如图4.1所示。

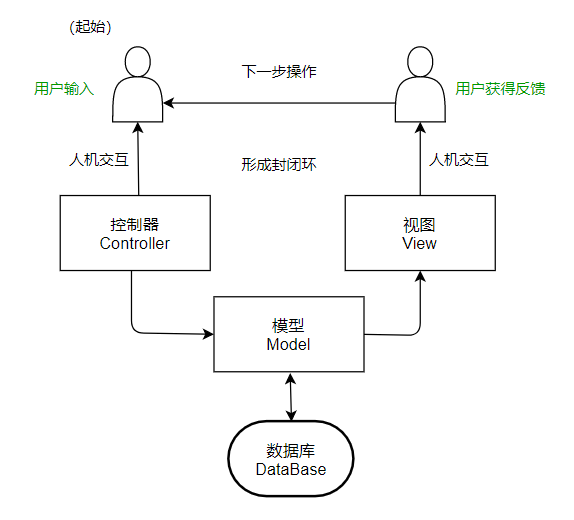


图4.1 MVC关系图

4.2.2 Django框架

本系统后端使用 Django 框架结合 Sqlite3 数据库设计[21]。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)。它们各自的职责所示。Django框架（内部的URLconf)作为控制器的角色，负责了接收用户请求和转发请求的工作。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。其优点如下：

（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。

## 4.3 功能模块

本系统用户分为两种，分别是后台管理员和前台用户。前台用户需要注册账号并成功登陆进入该电影推荐系统，可以通过搜索电影类别查询各类电影，并查看电影详细信息。最重要的是可以通过个人喜好和收藏等习惯推荐其可能喜欢的电影。管理员需要登陆后台，进行电影及分类、上新等，管理用户信息以及点击量管理等。本系统主要功能模块如图4.2所示。

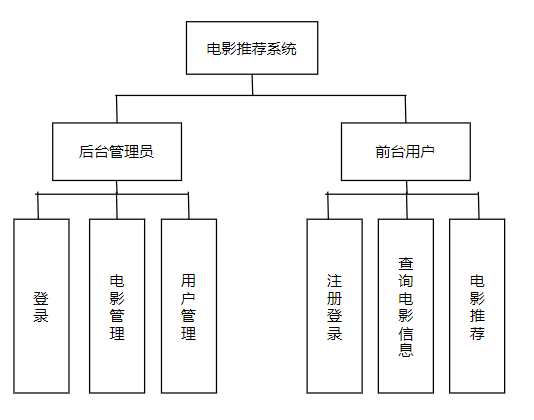


图4.2 系统功能模块图

## 4.4 系统表结构

本系统的运行十分依赖于与数据库的交互，所以数据库的建立必须是整洁有条理而且规范的。而系统所运用到的协同过滤也是以相当规模的数据集为基础，包括后期的推荐结果测试也同样是需要反复与数据库进行交互[22]。

系统后台管理员表如表4-1所示。

表4-1管理员信息表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 默认 | 是否允许为空 | 说明 |
| Id(PRI）  Username  Password | Int  Varchar  Varchar | Null  Null  Null | No  Yes  Yes | 管理员编号为主键  管理员名  密码 |

电影类型表如表4-2所示。

该表用于记录电影类型，而用户注册时会相应地添加自己喜欢的类型，用于后期给用户推荐其他相同类型的电影。

表4-2电影类型表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 默认 | 是否允许为空 | 说明 |
| Id(PRI）  Types | Int  Varchar | Null  Null | No  Yes | 电影类型的编号为主键  类型名 |

电影信息表如表4-3所示。

该表主要用于记录电影的基本信息，并将信息提交给前端显示。

用户信息表如表4-4所示。

用于记录用户基本信息，用于登录与注册。

用户喜好表如表4-5所示

该表记录用户注册系统时添加的所喜好电影的类型，从而作为后期给用户推荐相同类型电影以及跟用户有相同喜好的其他用户喜爱的电影的依据。

表4-3电影信息表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 默认 | 是否允许为空 | 说明 |
| Id(PRI）  Moviename  Actor  Director  Types  Area  Years  Abstract  Clicknum | Int  Varchar  Varchar  Varchar  Varchar  Varchar  Varchar  Varchar  Int | Null  Null  Null  Null  Null  Null  Null  Null  Null | No  Yes  Yes  Yes  Yes  Yes  Yes  Yes  Yes | 管理员编号为主键  电影名  主演  导演  类型  地区  年代  简介  点击数 |

表4-4用户信息表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 默认 | 是否允许为空 | 说明 |
| Id(PRI）  Username | Int | Null | No | 编号为主键 |
| Varchar | Null | Yes | 用户昵称 |
| Password | Varchar | Null | Yes | 密码 |
| Mobile | Varchar | Null | Yes | 手机 |
| Name | Varchar | Null | Yes | 用户姓名 |

表4-5用户喜好表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 默认 | 是否允许为空 | 说明 |
| Id(PRI） | Int | Null | No | 编号为主键 |
| Typename | Varchar | Null | Yes | 类型名 |
| cid | Int | Null | Yes | 点击量表id |

电影点击量表如图4-6所示

电影点击量表会记录下每一部电影的点击次数，以及点击者的id，从而作为后期推荐给该点击者其他可能感兴趣的电影的依据。

表4-6电影点击量表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 数据类型 | 默认 | 是否允许为空 | 说明 |
| Id(PRI） | Int | Null | No | 编号为主键 |
| Fid | Int | Null | Yes | 电影编号 |
| Cid | Int | Null | Yes | 点击量编号 |
| insertdate | Datetime | Null | Yes | 点击时间 |

## 4.5 数据字典

数据库中各表字段以及关联信息如下：

表4-7 auth\_group\_permissions

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 注释 |
| id | int |  | 主键 |
| group\_id | int |  | 外键 |
| permission\_id | int |  | 外键 |

表4-8 auth\_permission

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 注释 |
| id | int |  | 主键 |
| name | varchar | 255 |  |
| content\_type\_id | int |  | 外键 |
| codename | varchar | 100 | 外键 |

表4-9 auth\_user\_user\_permissions

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 注释 |
| id | int |  | 主键 |
| user\_id | int | 100 | 外键 |
| permission | Int | 100 | 外键 |

表4-10 django\_content\_type

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 注释 |
| id | int |  | 主键 |
| app\_label | varchar |  | 外键 |
| model | varchar |  | 外键 |

表4-11 auth\_group

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 注释 |
| id | int |  | 主键 |
| name | varchar |  | 外键 |

表4-12 auth\_user\_groups

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 注释 |
| id | int |  | 主键 |
| user\_id | int |  | 外键 |
| group\_id | int |  | 外键 |

表4-13 auth\_user

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 注释 |
| id | int |  | 主键 |
| password | varchar | 128 |  |
| last\_login | datetime | 6 |  |
| is\_superuser | tinyint | 1 |  |
| username | varchar | 150 | 外键 |
| first\_name | varchar | 30 |  |
| last\_name | varchar | 150 |  |
| email | varchar | 254 |  |
| is\_staff | tinyint | 1 |  |
| is\_active | tinyint | 1 |  |
| date\_joined | datetime | 6 |  |

表4-14 django\_admin\_log

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 注释 |
| id | int |  | 主键 |
| action\_time | datetime | 6 |  |
| object\_id | longtext |  |  |
| object\_repr | varchar | 200 |  |
| action\_flag | smalint UNSIGNED |  |  |
| change\_message | longtext |  |  |
| content\_type\_id | int |  | 外键 |
| user\_id | int |  | 外键 |

表4-15 movies\_3

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 注释 |
| id | varchar | 255 |  |
| title | varchar | 255 |  |
| image\_link | varchar | 255 |  |
| country | varchar | 255 |  |
| years | varchar | 255 |  |
| director\_description | varchar | 255 |  |
| leader | varchar | 255 |  |
| star | varchar | 255 |  |
| description | varchar | 255 |  |
| tags | varchar | 255 |  |
| imdb | varchar | 255 |  |
| language | varchar | 255 |  |
| time\_length | varchar | 255 |  |

表4-16 django\_migrations

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 注释 |
| id | int |  | 主键 |
| app | varchar | 255 |  |
| name | varchar | 255 |  |
| applied | datetime | 6 |  |

表4-17 movie\_likecomment

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 注释 |
| id | int |  | 主键 |
| comment\_id | int |  | 外键 |
| user\_id | int |  | 外键 |

表4-18 movie\_usertagprefer

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 注释 |
| id | int |  | 主键 |
| score | double |  |  |
| tag\_id | int |  | 外键 |
| user\_id | int |  | 外键 |

表4-19 movie\_user

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 注释 |
| id | int |  | 主键 |
| username | varchar | 255 | 外键 |
| password | varchar | 255 |  |
| email | varchar | 254 |  |
| created\_time | datetime | 6 |  |

表4-20 movie\_comment

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 注释 |
| id | int |  | 主键 |
| content | int | 255 |  |
| create\_time | datetime | 6 |  |
| movie\_id | int |  | 外键 |
| user\_id | int |  | 外键 |

表4-21 movie\_movie\_collect

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 注释 |
| id | int |  | 主键 |
| movie\_id | int |  | 外键 |
| user\_id | int |  | 外键 |

表4-22 movie\_rate

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 注释 |
| id  mark  create\_time | int  double  datetime |  | 主键 |
| movie\_id | int |  | 外键 |
| user\_id | int |  | 外键 |

表4-23 movie\_movie

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 注释 |
| id | int |  | 主键 |
| name | varchar | 255 | 外键 |
| director | varchar | 255 |  |
| country | varchar | 255 |  |
| years | date |  |  |
| leader | varchar | 1024 |  |
| d\_rate\_nums | int |  |  |
| d\_rate | varchar | 255 |  |
| intro | longtext |  |  |
| num | int |  |  |
| origin\_image\_link | varchar | 255 |  |
| image\_link | varchar | 255 |  |
| imdb\_link | varchar | 255 |  |

表4-24 movie\_tages

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 注释 |
| id | int |  | 主键 |
| name | varchar | 255 | 外键 |

表4-25 movie\_movie\_tags

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 注释 |
| id | int |  | 主键 |
| movie\_id | int |  | 外键 |
| tags\_id | int |  | 外键 |

表4-26 django\_session

|  |  |  |  |
| --- | --- | --- | --- |
| 列名 | 数据类型 | 长度 | 注释 |
| session\_key | varchar | 40 | 主键 |
| session\_data | longtext |  |  |
| expire\_date | datetime | 6 | 外键 |

# 5 使用手册

## 5.1 使用环境配置

（1）在Pycharm编译器中打开名为“2021movie”文件夹为项目文件夹。

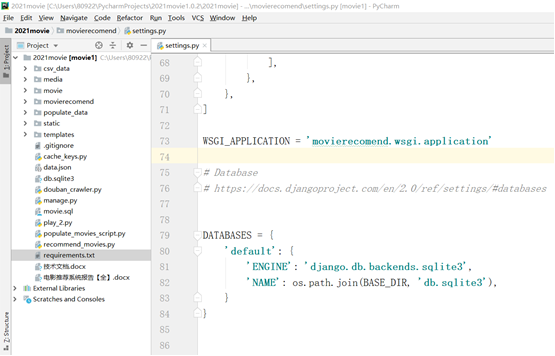


图5.1 打开项目文件夹

（2）打开配置Python解释器，选择Settings

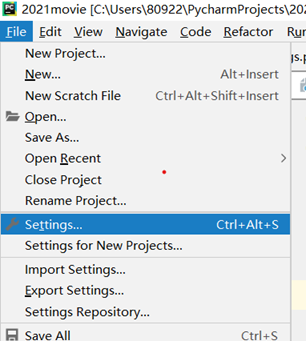


图5.2 选择Settings查看环境配置

（3）进入文件夹依次打开Project:2021movie->Project Interpreter，此时项目文件夹未设置解释器，点击显示‘<No interpreter>’的下拉菜单选择现有的Python解释器，点击ok按钮使用选中的解释器。

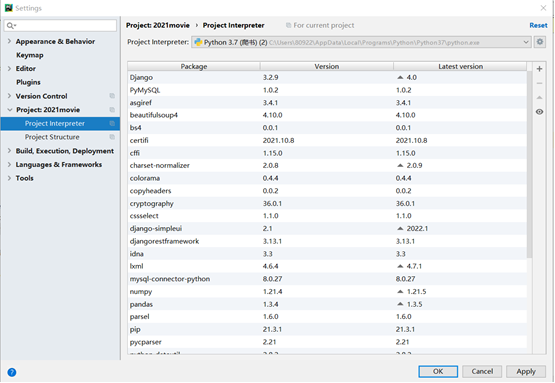


图5.3 所拥有的环境

（4）若缺失需要的库，可选择使用图形化界面点击安装或命令行代码安装。

图形化界面安装：

在上图界面点击环境表右侧”+”号进入搜索界面，搜索到需要的依赖库之后，会弹出如下界面，需要从中选择合适的库的版本（例如图，本程序需要安装Django 2.2.10及以上版本），点击左下方install Package进行安装。

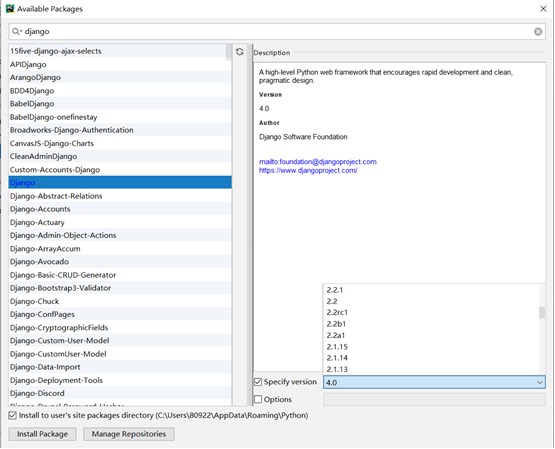


图5.4 图形化界面安装环境

命令行安装：

进入主界面，点击左下方terminal进入终端界面。使用pip指令安装，输入pip install <依赖包名>==<特定版本>；；

运行该指令即可安装（已安装会显示依赖包已满足需要）。

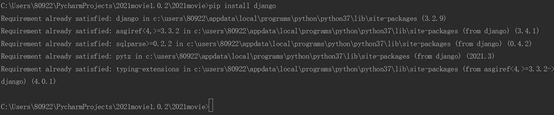


图5.5 命令行安装

## 5.2 数据爬取及存储

（1）打开Pycharm，新建movie2 project

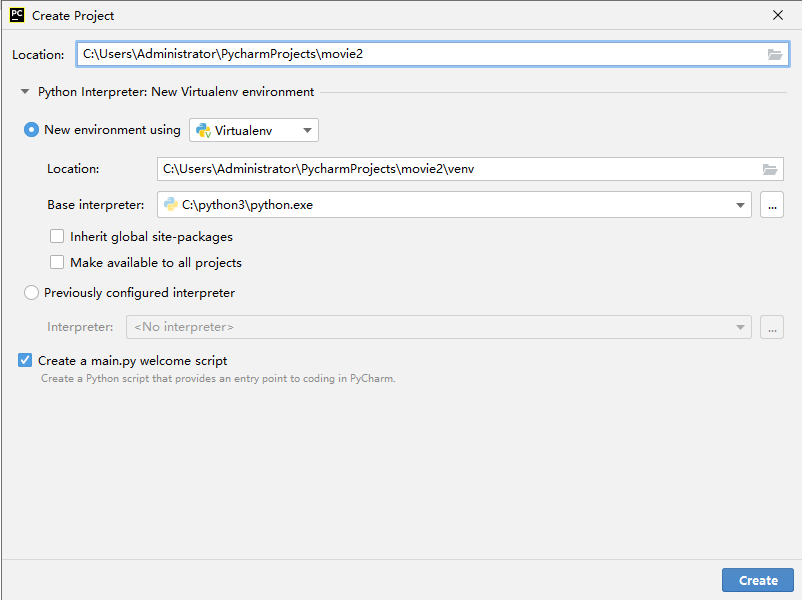


图5.6 新建Project

（2）Crawler用于存豆瓣爬虫文件，新建selecuim\_douban\_crawler.py，用于抓取动态爬虫链接。

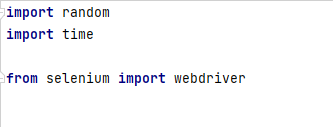


图5.7 新建selecuim\_douban\_crawler.py

（3）在新建的py文件中输入如下爬取代码；在Crawler中新建get\_details.py，用于电影详细信息。

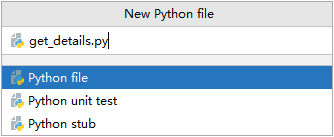


图5.8 新建get\_details.py

（4）调用包



图5.9 所调用的包

（5）最后爬取的电影及相关信息如下



图5.10 爬取的电影的信息

## 5.3 结果展示

（1）注册登录模块

描述：主要实现用户登陆注册功能。

对于新用户可以在推荐系统中注册自己的账号，并登录；对于老用户，可以直接登录系统进行电影搜索及查看。

界面：



图5.11 用户注册



图5.12 用户登录

（2）用户浏览电影

描述：浏览页面首页，其他页面引用此界面样式再进行其他信息展示。get\_user\_recommend() 方 法 用 于 调 用 后 端 基 于 用 户 推 荐 电 影 数 据 ， get\_latest\_movie()方法用于调用最新的电影数据。另外可根据热度/收藏/评分/时间，浏览不同排序的电影，在标签页，可根据标签浏览不同标签下的电影。

界面：

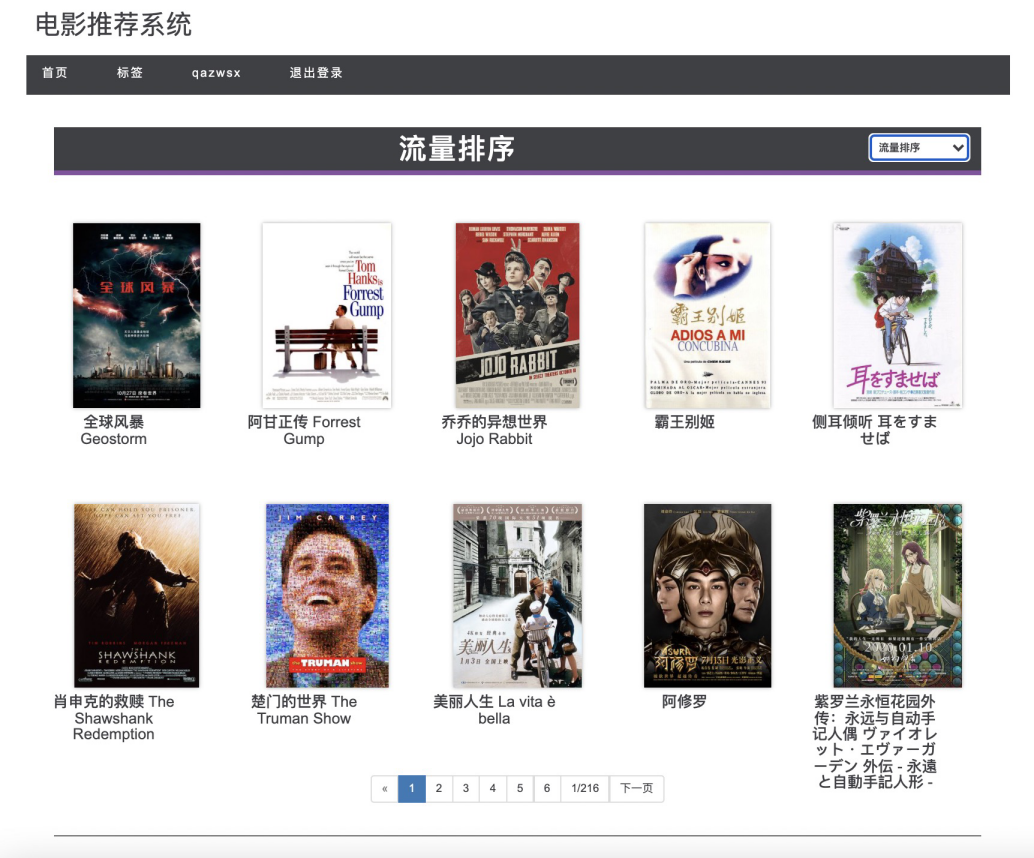


图5.13 按流量排序

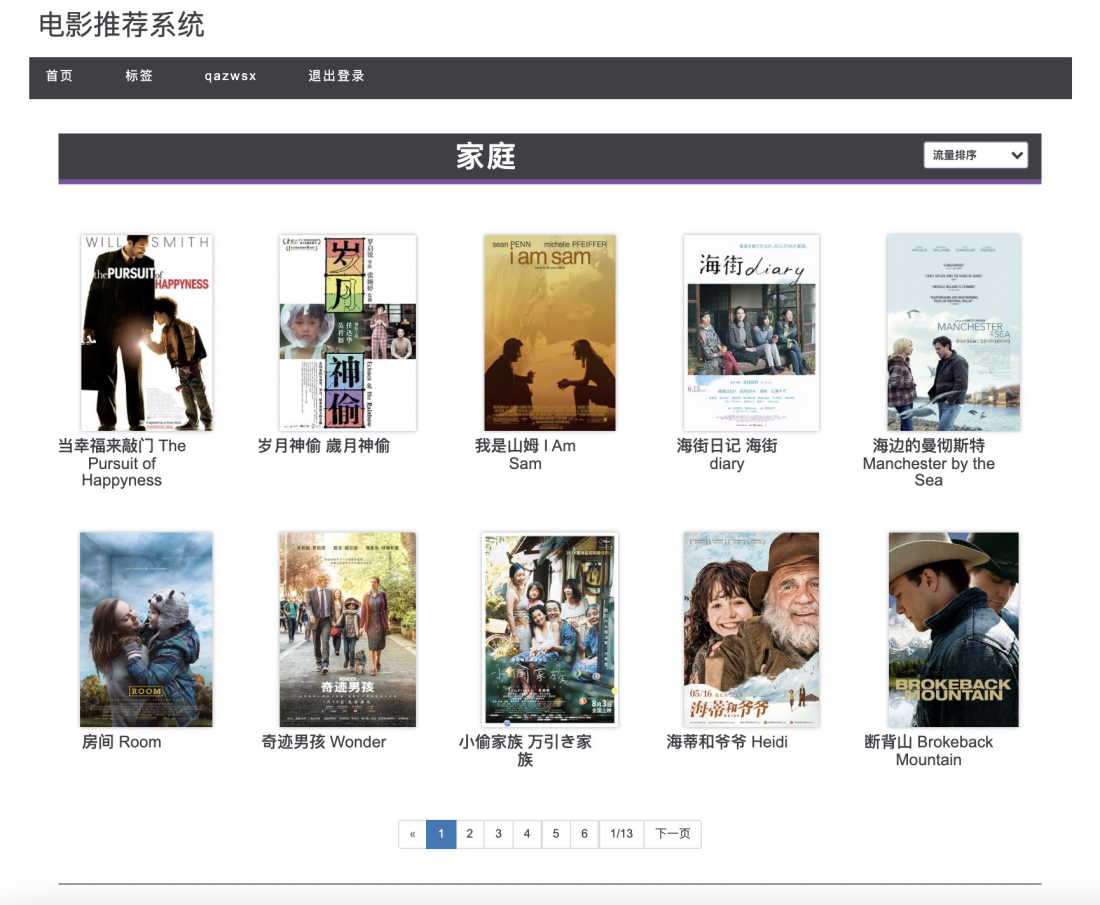


图5.14 按家庭标签查询

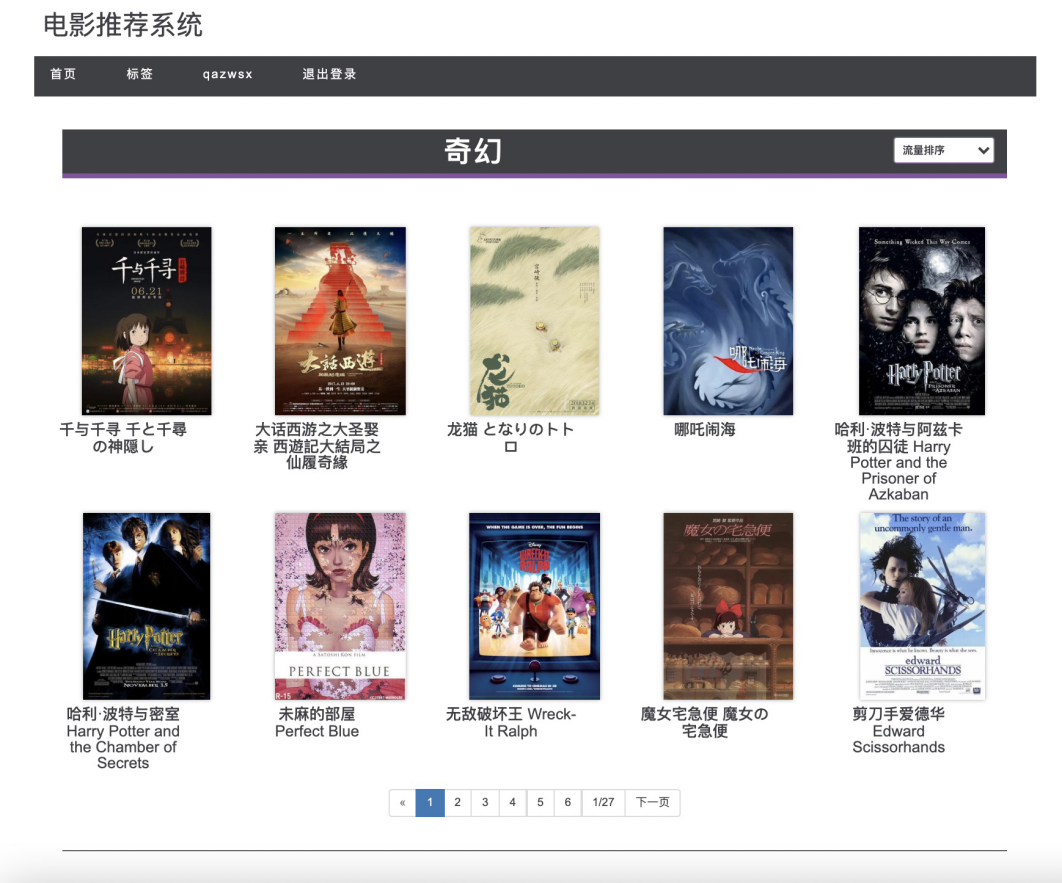


图5.15 按奇幻标签查询

（3）基于用户的电影推荐功能

描述：基于用户对电影的打分数据推荐，用户每次打分后，页面用户推荐数据将清除缓存，重新计算数据，并重新生成推荐数据，展示在页面新的基于用户推荐数据。算法主要通过计算该用户与 10 位用户的距离，找到距离最近的用户，即喜好最相似的用户，根据此用户的数据向目前用户用户推荐电影。

界面：

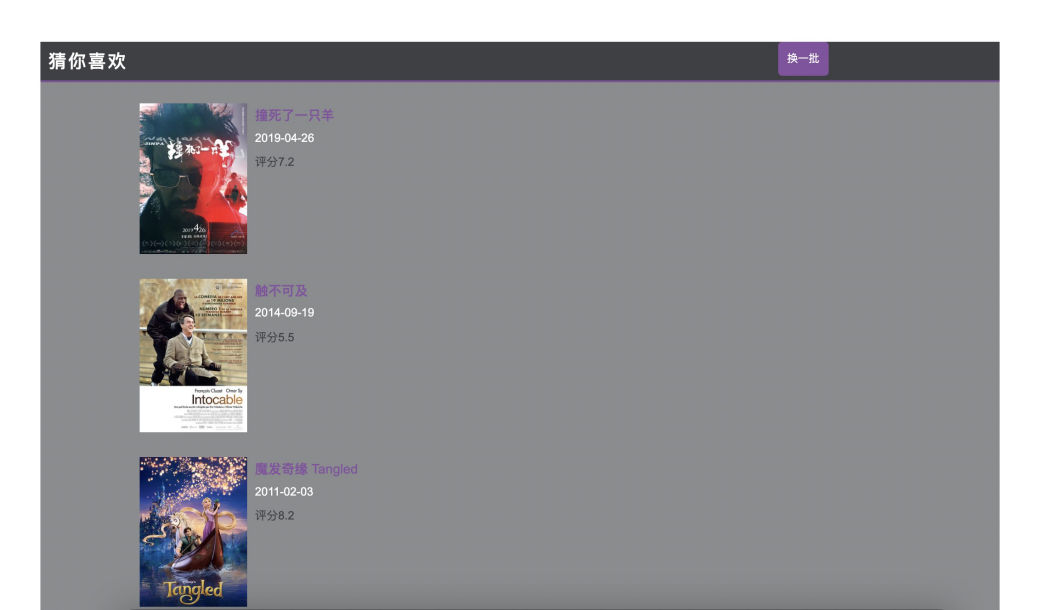


图5.16 猜你喜欢

（4）基于物品的电影推荐功能以及用户评分

描述：主要通过找到被共同用户评分的两个电影，余弦计算两电影相似度，找出与当前电影相似的电影，生成用户电影推荐列表并展示出来。

界面：



图5.17 为你推荐和添加评分

（5）用户收藏电影

描述：用户可在电影详情界面收藏或取消收藏电影，并可在我的信息中我的收藏查看收藏的电影。

界面：



图5.18 收藏电影

（6）用户修改个人信息

描述：用户可在个人信息界面对个人信息进行修改。可以修改账号、密码及邮箱或重置，还包括可以查看我的收藏、我的评论以及我的评分。

界面：

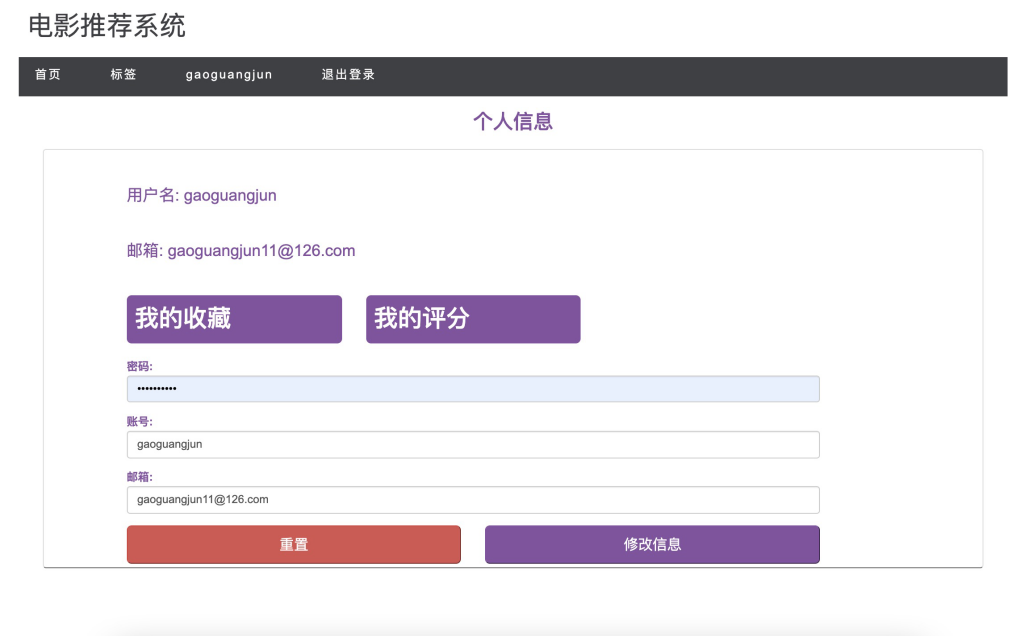


图5.19 用户修改信息功能展示

（7）用户观影界面

描述：用户可进行影片搜索并查看搜索结果，点击前往观看按钮，即可链接到第三方网站，从而观看该电影。

界面：

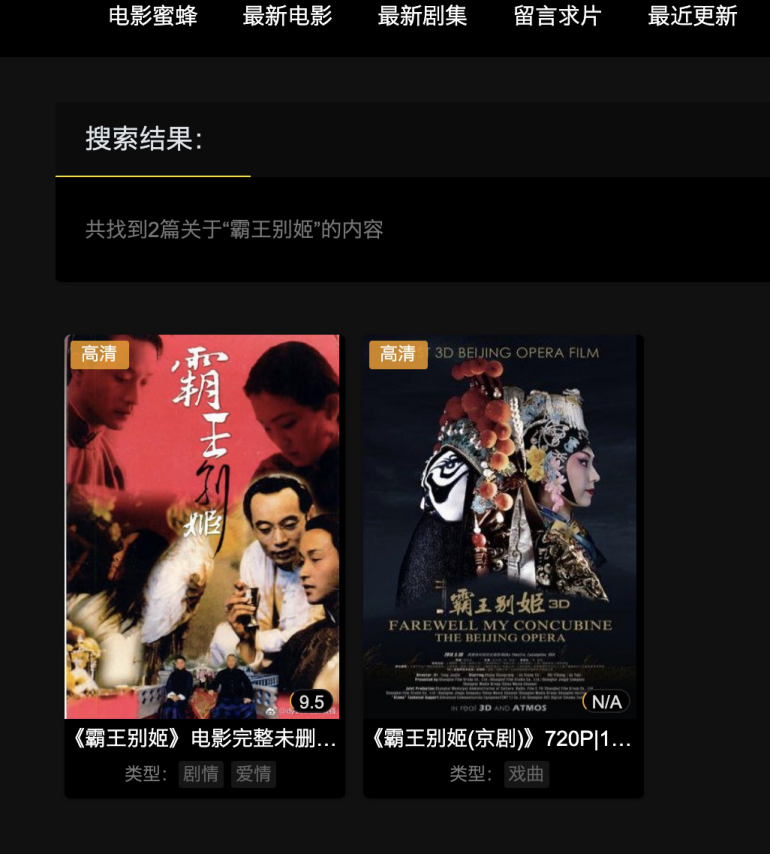


图5.20 用户观影界面

（8）后台管理模块

描述：后台管理为 Djiango 框架中所集成的可视化管理模块，能方便管理数据库中用户信息，电影信息等各项信息。但需要通过 python manage.py createsuperuser 命令建立超级用户进入后台管理。

界面：

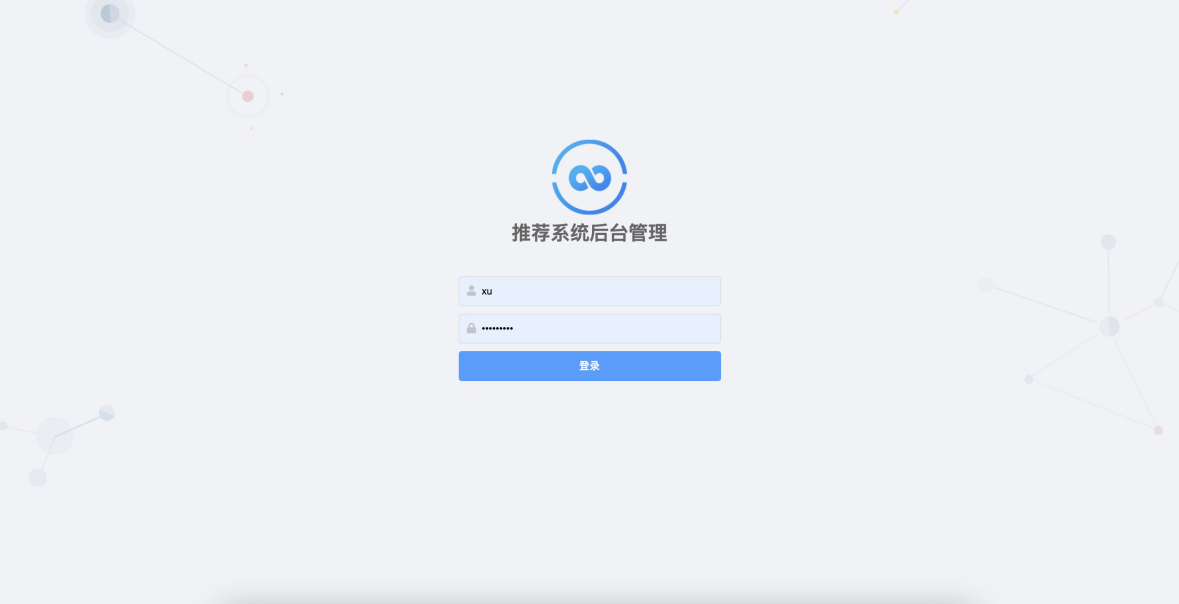


图5.21 后台管理登录展示

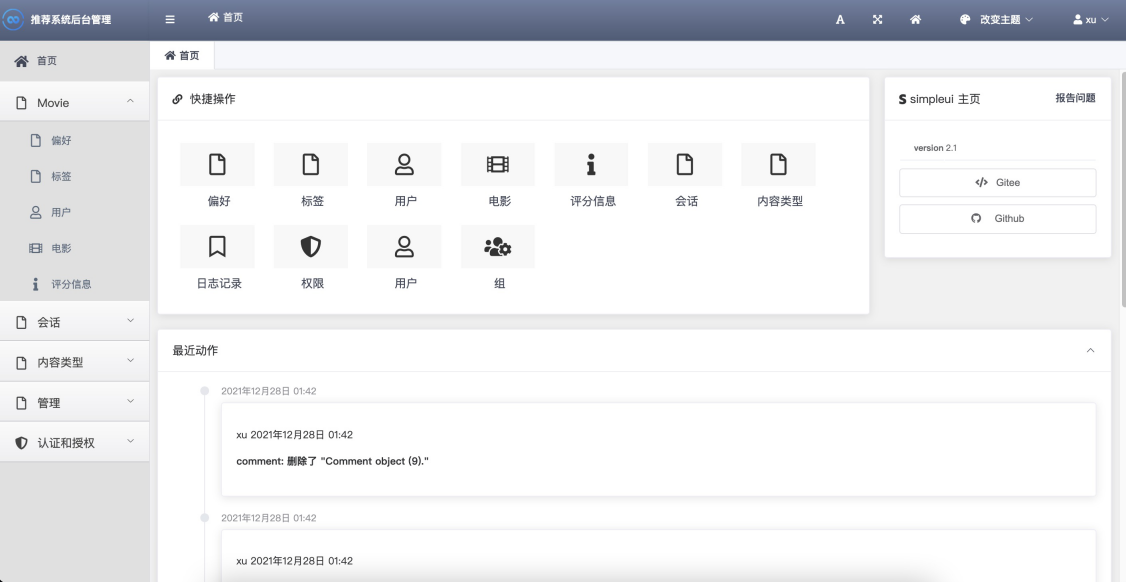


图5.22 后台管理页面展示

# 6 代码清单

## 6.1 数据爬虫代码

豆瓣影片TOP250爬虫代码如下：

import asyncio

import aiohttp

from bs4 import BeautifulSoup

base\_url = 'https://movie.douban.com/top250'

results = {}

new\_url = 'https://www.douban.com/doulist/30299/?start=0&sort=seq&playable=0&sub\_type='

ids = 0

import csv

tasks = []

class Movie:

def \_\_init\_\_(self, id, title, description, star, leader, tags, years, country, director\_description, image\_link):

self.id = id

self.star = star

self.description = description

self.title = title

self.leader = leader

self.tags = tags

self.years = years

self.country = country

self.director\_description = director\_description

self.image\_link = image\_link

async def fetch(url):

async with aiohttp.ClientSession()as session:

async with session.get(url) as response:

assert response.status == 200

return await response.text()

async def write\_images(image\_link, image\_name):

print('write images....', image\_link)

async with aiohttp.ClientSession()as session:

async with session.get(image\_link) as response:

assert response.status == 200

with open('movie\_images/' + image\_name + '.png', 'wb')as opener:

while True:

chunk = await response.content.read(1024)

if not chunk:

break

opener.write(chunk)

async def parse\_list(html):

soup = BeautifulSoup(html, 'html.parser')

movies = soup.find\_all('div', {'class': 'doulist-item'})

movie\_list = []

for movie in movies:

try:

title = movie.find('div', {'class': 'title'}).text.strip().replace('/', '\_')

except Exception as e:

print(movie)

print(e)

continue

try:

image\_link = movie.find('div', {'class': 'post'}).find('img')

if image\_link is None:

continue

else:

image\_link = image\_link['src']

await write\_images(image\_link, title)

rate = movie.find('div', {'class': 'rating'})

dou\_rate = rate.find('span', {'class': 'rating\_nums'})

if dou\_rate is None:

dou\_rate = '0'

else:

dou\_rate = dou\_rate.text

# rate\_num = rate.find\_all('span')[-1].text

abstract = movie.find('div', {'class': 'abstract'}).text.strip().split('\n')

tags = country = leader = year = director\_ = ''

for ab in abstract:

if len(ab.strip()) == 0:

continue

ab\_list = ab.split(':')

key = ab\_list[0].strip()

value = ab\_list[1].strip()

if key == '导演':

director\_ = value

elif key == '主演':

leader = value

elif key == '年份':

year = value

elif key == '制片国家/地区':

country = value

elif key == '类型':

tags = value

except Exception as e:

print(movie)

raise e

global ids

ids += 1

movie\_list.append(Movie(image\_link=image\_link, title=title, star=dou\_rate, leader=leader, tags=tags, country=country, director\_description=director\_, years=year, id=ids, description=''))

new\_link = soup.find('span', {'class': 'next'})

if new\_link is not None:

try:

new\_link = new\_link.a['href']

except Exception:

new\_link = None

return movie\_list, new\_link

async def parse\_250(html):

soup = BeautifulSoup(html, 'html.parser')

movies\_info = soup.find('ol', {'class': 'grid\_view'})

movies = []

for movie\_info in movies\_info.find\_all('li'):

pic = movie\_info.find('div', {'class': 'image\_link'})

picture\_url = pic.find('img').attrs['src']

movie\_id = pic.find('em').text

url = movie\_info.find('div', {"class": "info"})

title = url.find('span', {'class': 'title'}).text

# 保存图片文件到本地

print('write image ', picture\_url)

await write\_images(picture\_url, title)

info = url.find('div', {'class': 'bd'})

movie\_detail = info.find('p')

quote = info.find('p', {'class': 'quote'})

if quote is not None:

description = quote.find('span').text

else:

description = ''

print(title + 'description is None')

star = info.find('div', {"class": 'star'}).find('span', {'class': 'rating\_num'}).text

tags = movie\_detail.text.strip().split('\n')[-1].split('/')[-1].split(' ')

tags = [tag.strip() for tag in tags]

years = movie\_detail.text.strip().split('\n')[-1].split('/')[0].strip()

country = movie\_detail.text.strip().split('\n')[-1].split('/')[1].strip()

temp = movie\_detail.text.strip().split('\n')

try:

director\_description = temp[0].split('/')[0].strip().split('\xa0')[0].split(':')[1]

except IndexError:

director\_description = ''

try:

leader = temp[0].split('/')[0].strip().split('\xa0')[-1].split(':')[1]

except IndexError:

leader = ''

assert title is not None

assert star is not None

assert leader is not None

assert years is not None

assert country is not None

assert director\_description is not None

assert tags is not None

assert picture\_url is not None

assert movie\_id is not None

movies.append(Movie(title=title, description=description, star=star, leader=leader, years=years, country=country, director\_description=director\_description, tags=tags, image\_link=picture\_url,

id=movie\_id

)

)

next\_page = soup.find('link', {'rel': 'next'})

if next\_page is not None and next\_page.attrs.get('href'):

next\_link = base\_url + next\_page.attrs['href']

else:

print('finished!')

next\_link = None

return movies, next\_link

def write\_movies(movies):

print('write movies..')

with open('movies\_2.csv', 'a+')as opener:

writer = csv.writer(opener)

if opener.tell() == 0:

writer.writerow(['id', 'title ', 'image\_link ', 'country ', 'years ', 'director\_description', 'leader', 'star ', 'description', 'tags'])

for movie in movies:

writer.writerow([str(movie.id), movie.title, movie.image\_link, movie.country, movie.years, movie.director\_description, movie.leader, movie.star, movie.description, '/'.join(movie.tags)])

async def parse\_movie\_link(url):

content = await fetch(url)

soup = BeautifulSoup(content, 'html.parser')

movies\_info = soup.find('ol', {'class': 'grid\_view'})

for movie\_info in movies\_info.find\_all('li'):

url = movie\_info.find('div', {"class": "info"}).find('div', {'class': 'hd'}).find('a')['href']

print('write', url)

with open('top250\_link.csv', 'a+')as opener:

opener.write(url + '\n')

def get\_movie\_link\_event\_loop():

loop = asyncio.get\_event\_loop()

base\_url = 'https://movie.douban.com/top250?start={}&filter='

for i in range(0, 250, 25):

url = base\_url.format(i)

print('add url to queue', url)

tasks.append(parse\_movie\_link(url))

loop.run\_until\_complete(asyncio.wait(tasks))

get\_movie\_link\_event\_loop()

## 6.2 动态爬虫抓取链接代码

动态爬虫抓取链接代码如下：

import random

import time

from selenium import webdriver

def get\_movie\_links(nums=1000):

browser.get(url)

link\_list = set()

links = browser.find\_elements\_by\_xpath("html//div[@class='list-wp']//a[@target='\_blank']")

browser.execute\_script("window.scrollTo(0, document.body.scrollHeight)")

time.sleep(1)

while len(links) < nums:

print('nums:', len(links))

browser.execute\_script("window.scrollTo(0, document.body.scrollHeight)")

time.sleep(random.randint(1, 10) / 10)

more = browser.find\_element\_by\_xpath("html//a[@class='more']")

while len(browser.window\_handles) > 1:

browser.switch\_to.window(browser.window\_handles[1])

browser.close()

browser.switch\_to.window(browser.window\_handles[0])

print(more.get\_attribute('href'))

more.click()

links = browser.find\_elements\_by\_xpath("html//div[@class='list-wp']//a[@target='\_blank']")

num = 0

while num < nums:

for link in links:

href = link.get\_attribute('href')

print('href', href)

if href not in link\_list:

link\_list.add(href)

num += 1

with open('movie\_links\_1.csv', 'a')as opener:

for href in link\_list:

opener.write(href + '\n')

if \_\_name\_\_ == '\_\_main\_\_':

url = 'https://movie.douban.com/tag/#/?sort=T&range=0,10&tags=%E7%94%B5%E5%BD%B1,2010%E5%B9%B4%E4%BB%A3'

option = webdriver.ChromeOptions()

# option.add\_experimental\_option('excludeSwitches', ['enable-automation'])

browser = webdriver.Chrome(executable\_path='/usr/local/bin/chromedriver', options=option)

get\_movie\_links(2000)

## 6.3 抓取电影详细信息代码

抓取电影详细信息代码如下：

import asyncio

import logging

import aiohttp

from bs4 import BeautifulSoup

results = {}

ids = 0

import csv

tasks = []

class Movie:

def \_\_init\_\_(self, id, title, description, star, leader, tags, years, country, director\_description, image\_link, language, time\_length, imdb\_link):

self.imdb\_link = imdb\_link

self.time\_length = time\_length

self.id = id

self.star = star

self.description = description

self.title = title

self.leader = leader

self.tags = tags

self.years = years

self.country = country

self.director\_description = director\_description

self.image\_link = image\_link

self.language = language

async def fetch(url, index):

print('fetching url', url)

async with aiohttp.ClientSession()as session:

async with session.get(url) as response:

assert response.status == 200

print("this is index", index)

return await response.text()

async def write\_images(image\_link, image\_name):

print('write images....', image\_link)

async with aiohttp.ClientSession()as session:

async with session.get(image\_link) as response:

assert response.status == 200

with open('movie\_images/' + image\_name + '.png', 'wb')as opener:

while True:

chunk = await response.content.read(1024)

if not chunk:

break

opener.write(chunk)

async def parse\_movie\_page(html):

soup = BeautifulSoup(html, 'html.parser')

title = soup.find('span', {'property': 'v:itemreviewed'}).text

image\_link = soup.find('a', {'class': 'nbgnbg'}).find('img')['src']

info = soup.find('div', {'id': 'info'})

director = info.find\_all('a', {'rel': 'v:directedBy'})[0].text

leader = [a.text for a in info.find\_all('a', {'rel': 'v:starring'})]

tags = [a.text for a in info.find\_all('span', {'property': 'v:genre'})]

country = info.find('span', string='制片国家/地区:').next\_sibling

language = info.find('span', string='语言:').next\_sibling

show\_time = info.find('span', {'property': 'v:initialReleaseDate'}).text

time\_length = info.find('span', {'property': 'v:runtime'}).text

imdb\_link = info.find('span', string='IMDb链接:').fetchNextSiblings()[0]['href']

description = soup.find('span', {'property': 'v:summary'}).text

star = soup.find('strong', {'property': 'v:average'}).text

# comments = soup.find\_all('div', {'class': 'comment-item'})

return Movie(image\_link=image\_link, title=title, star=star, leader=leader, tags=tags, country=country, director\_description=director, years=show\_time, id=ids, description=description, time\_length=time\_length, imdb\_link=imdb\_link, language=language)

async def write\_movies(movie, index, filename):

print('write movies..')

with open(filename, 'a+')as opener:

writer = csv.writer(opener)

if opener.tell() == 0:

writer.writerow(['id', 'title ', 'image\_link ', 'country ', 'years ', 'director\_description', 'leader', 'star ', 'description', 'tags', 'imdb', 'language', 'time\_length', ''])

writer.writerow([index, movie.title, movie.image\_link, movie.country, movie.years, movie.director\_description, movie.leader, movie.star, movie.description, '/'.join(movie.tags), movie.imdb\_link, movie.language, movie.time\_length])

async def get\_results(index, url, parser, filename):

html = await fetch(url, index)

movies = await parser(html)

await write\_images(image\_link=movies.image\_link, image\_name=movies.title.replace('/', '\_'))

await write\_movies(movies, index, filename)

async def handle\_tasks(work\_queue, parser, filename):

while not work\_queue.empty():

index, current\_url = await work\_queue.get()

current\_url = current\_url[0]

print('index', current\_url, index)

try:

await get\_results(index, current\_url, parser, filename)

except Exception as e:

logging.exception('Error for {}'.format(current\_url), exc\_info=True)

def envent\_loop(link\_filename, write\_filename):

q = asyncio.Queue()

with open(link\_filename, 'r')as opener:

reader = csv.reader(opener)

for index, link in enumerate(reader):

q.put\_nowait((index, link))

loop = asyncio.get\_event\_loop()

tasks.append(handle\_tasks(q, parse\_movie\_page, write\_filename))

loop.run\_until\_complete(asyncio.wait(tasks))

if \_\_name\_\_ == '\_\_main\_\_':

# 读写250电影内容

envent\_loop(link\_filename='../csv\_data/top250\_link.csv', write\_filename='top250.csv')

## 6.4 连接数据库代码

"""

Django settings for movie project.

Generated by 'django-admin startproject' using Django 2.2.

For more information on this file, see

https://docs.djangoproject.com/en/2.2/topics/settings/

For the full list of settings and their values, see

https://docs.djangoproject.com/en/2.2/ref/settings/

"""

import os

import pymysql

pymysql.install\_as\_MySQLdb()

# Build paths inside the project like this: os.path.join(BASE\_DIR, ...)

BASE\_DIR = os.path.dirname(os.path.dirname(os.path.abspath(\_\_file\_\_)))

# Quick-start development settings - unsuitable for production

# See https://docs.djangoproject.com/en/2.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!

SECRET\_KEY = "r^n(t&=!%nn!+5\*^qsto\_7o0b41x-\*uo1u01)^px-71xbj!u4s"

# SECURITY WARNING: don't run with debug turned on in production!

DEBUG = True

ENV\_PROFILE = os.getenv("ENV")

ALLOWED\_HOSTS = ['localhost', '127.0.0.1', 'movie2.colaplusice.com','movie2.colaplusice.com']

# Application definition

INSTALLED\_APPS = [

"simpleui",

"django.contrib.admin",

"django.contrib.auth",

"django.contrib.contenttypes",

"django.contrib.sessions",

"django.contrib.messages",

"django.contrib.staticfiles",

"rest\_framework",

'corsheaders',

"user",

]

MIDDLEWARE = [

'corsheaders.middleware.CorsMiddleware',

"django.middleware.security.SecurityMiddleware",

"django.contrib.sessions.middleware.SessionMiddleware",

"django.middleware.common.CommonMiddleware",

"django.middleware.csrf.CsrfViewMiddleware",

"django.contrib.auth.middleware.AuthenticationMiddleware",

"django.contrib.messages.middleware.MessageMiddleware",

"django.middleware.clickjacking.XFrameOptionsMiddleware",

]

# CORS\_ORIGIN\_ALLOW\_ALL = True

ROOT\_URLCONF = "movie.urls"

TEMPLATES = [

{

"BACKEND": "django.template.backends.django.DjangoTemplates",

"DIRS": [],

"APP\_DIRS": True,

"OPTIONS": {

"context\_processors": [

"django.template.context\_processors.debug",

"django.template.context\_processors.request",

"django.contrib.auth.context\_processors.auth",

"django.contrib.messages.context\_processors.messages",

"user.context\_processors.env\_profile",

'django.template.context\_processors.media', # set this explicitly

]

},

}

]

SIMPLEUI\_HOME\_INFO = False

WSGI\_APPLICATION = "movie.wsgi.application"

# Database

# https://docs.djangoproject.com/en/2.2/ref/settings/#databases

DATABASES = {

"default": {

"ENGINE": "django.db.backends.sqlite3",

"NAME": os.path.join(BASE\_DIR, "db.sqlite3"),

}

}

# DATABASES = {

# 'default': {

# 'ENGINE': 'django.db.backends.mysql',

# 'NAME': 'movie2',

# 'USER': 'root',

# 'PASSWORD': '780076',

# 'HOST': 'localhost',

# 'PORT': '3306',

# }

# }

#

CACHES = {

'default': {

'BACKEND': 'django.core.cache.backends.filebased.FileBasedCache',

'LOCATION': '/var/tmp/django\_cache',

}

}

# Password validation

# https://docs.djangoproject.com/en/2.2/ref/settings/#auth-password-validators

AUTH\_PASSWORD\_VALIDATORS = [

{

"NAME": "django.contrib.auth.password\_validation.UserAttributeSimilarityValidator"

},

{"NAME": "django.contrib.auth.password\_validation.MinimumLengthValidator"},

{"NAME": "django.contrib.auth.password\_validation.CommonPasswordValidator"},

{"NAME": "django.contrib.auth.password\_validation.NumericPasswordValidator"},

]

# Internationalization

# https://docs.djangoproject.com/en/2.2/topics/i18n/

LANGUAGE\_CODE = "zh-hans"

TIME\_ZONE = "Asia/Shanghai"

USE\_I18N = True

USE\_L10N = True

USE\_TZ = True

# Static files (CSS, JavaScript, Images)

# https://docs.djangoproject.com/en/2.2/howto/static-files/

static = os.path.dirname(os.path.abspath(\_\_file\_\_))

static = os.path.join(static, "../")

STATICFILES\_DIRS = (os.path.join(static, "static"),)

STATIC\_URL = "/static/"

MEDIA\_ROOT = os.path.join(BASE\_DIR, "media")

MEDIA\_URL = "/media/"

# 线上设置

# cdn url # collect static后，文件会在这里 方便在远程部署同步文件到nginx

if ENV\_PROFILE == 'production':

# cdn地址

STATIC\_URL = "http://static.colaplusice.com/movie2/static/"

# nginx配置的文件目录

STATIC\_ROOT = "/www/movie2/static"

#

# 第三方配置

GRAVATAR\_DEFAULT\_URL = "http://www.example.com/mydefaultavatar.jpg"

SIMPLEUI\_HOME\_INFO = False

SIMPLEUI\_ANALYSIS = False

## 6.5 将数据写入数据库

将爬取的TOP250影片数据写入数据库如下：

import ast

import datetime

import os

import django

# 将top250数据写入数据库

os.environ.setdefault("DJANGO\_SETTINGS\_MODULE", "movie.settings")

django.setup()

import csv

import re

from user.models import Tags, Movie

from populate\_data.clear\_movies import clear\_movie\_tags

# ss = '1994-09-10(多伦多电影节)'

def parse\_time(time\_str):

time = time\_str.split('-')

years = time[0]

if len(time) > 1:

month = time[1]

else:

month = 1

if len(time) > 2:

day = time[2]

else:

day = 1

time = datetime.date(int(years), int(month), int(day))

return str(time)

def replace\_special\_char(name):

special\_char = r'[\\/:\*?#@！%!"<>|：\s]'

return re.sub(special\_char, '\_', name)

def populate\_movies(filename):

opener = open(filename, 'r')

reader = csv.reader(opener)

next(reader)

for line in reader:

id, title, image\_link, country, years, director\_description, leader, star, description, tags, imdb, language, time\_length, douban\_link = line

origin\_years = years

# 数据清洗

years = re.search(pattern=r'\d{4}?(-\d{0,2})?(-\d{0,2})', string=years)

if years is None:

years = origin\_years.split('(')[0]

else:

years = years[0]

res = re.match('\d\*', star)

int\_d\_rate\_num = int(res[0]) if res else 0

# res = re.match('[\u4e00-\u9fa5]+', title)

# if res is not None:

# title = res.group()

pic\_name = 'movie\_cover/' + image\_link.split('/')[-1]

years = parse\_time(years)

leader = '/'.join(ast.literal\_eval(leader))

# 存入数据库

movie, created = Movie.objects.get\_or\_create(name=title, defaults={'image\_link': pic\_name, 'country': country,

'years': parse\_time(years), 'leader': leader,

'd\_rate\_nums': int\_d\_rate\_num,

'd\_rate': star, 'intro': description,

'director': director\_description,

'imdb\_link': imdb

})

if not created:

print('movie', movie)

tags = [tag.strip() for tag in tags.split('/')]

for tag in tags:

tag\_obj, created = Tags.objects.get\_or\_create(name=tag)

movie.tags.add(tag\_obj.id)

#

if \_\_name\_\_ == '\_\_main\_\_':

file1 = '../csv\_data/movies\_250.csv'

file2 = '../csv\_data/movies\_2000.csv'

# 清除数据

clear\_movie\_tags()

populate\_movies(file1)

populate\_movies(file2)

## 6.6 推荐算法代码

推荐算法代码如下：

import os

os.environ["DJANGO\_SETTINGS\_MODULE"] = "movie.settings"

import django

django.setup()

from user.models import \*

from math import sqrt, pow

import operator

from django.db.models import Q, Count, Subquery

from collections import defaultdict

import pickle

# from django.shortcuts import render,render\_to\_response

class UserCf:

# 获得初始化数据

def \_\_init\_\_(self, all\_user):

self.all\_user = all\_user

# 通过用户名获得商品列表，仅调试使用

def getItems(self, username1, username2):

return self.all\_user[username1], self.all\_user[username2]

# 计算两个用户的皮尔逊相关系数

def pearson(self, user1, user2): # 数据格式为：商品id，浏览此

sum\_xy = 0.0 # user1,user2 每项打分的成绩的累加

n = 0 # 公共浏览次数

sum\_x = 0.0 # user1 的打分总和

sum\_y = 0.0 # user2 的打分总和

sumX2 = 0.0 # user1每项打分平方的累加

sumY2 = 0.0 # user2每项打分平方的累加

for movie1, score1 in user1.items():

if movie1 in user2.keys(): # 计算公共的浏览次数

n += 1

sum\_xy += score1 \* user2[movie1]

sum\_x += score1

sum\_y += user2[movie1]

sumX2 += pow(score1, 2)

sumY2 += pow(user2[movie1], 2)

if n == 0:

# print("p氏距离为0")

return 0

molecule = sum\_xy - (sum\_x \* sum\_y) / n # 分子

denominator = sqrt((sumX2 - pow(sum\_x, 2) / n) \* (sumY2 - pow(sum\_y, 2) / n)) # 分母

if denominator == 0:

return 0

r = molecule / denominator

return r

# 计算与当前用户的距离，获得最临近的用户

def nearest\_user(self, current\_user, n=1):

distances = {}

# 用户，相似度

# 遍历整个数据集

for user, rate\_set in self.all\_user.items():

# 非当前的用户

if user != current\_user:

distance = self.pearson(self.all\_user[current\_user], self.all\_user[user])

# 计算两个用户的相似度

distances[user] = distance

closest\_distance = sorted(

distances.items(), key=operator.itemgetter(1), reverse=True

)

# 最相似的N个用户

# print("closest user:", closest\_distance[:n])

return closest\_distance[:n]

# 给用户推荐商品

def recommend(self, username, n=3):

recommend = {}

nearest\_user = self.nearest\_user(username, n)

for user, score in dict(nearest\_user).items(): # 最相近的n个用户

for movies, scores in self.all\_user[user].items(): # 推荐的用户的商品列表

if movies not in self.all\_user[username].keys(): # 当前username没有看过

if movies not in recommend.keys(): # 添加到推荐列表中

recommend[movies] = scores

# 对推荐的结果按照商品浏览次数排序

return sorted(recommend.items(), key=operator.itemgetter(1), reverse=True)

# 某个用户给电影打分后，更新all\_user dict

def update\_all\_user(self, user):

rates = user.rate\_set.all()

rate = {}

# 用户有给电影打分 在rate和all\_user中进行设置

if rates:

for i in rates:

rate.setdefault(str(i.movie.id), i.mark)

all\_user.setdefault(user.username, rate)

def get\_all\_user():

all\_user = {}

users\_rate = Rate.objects.values('user').annotate(mark\_num=Count('user')).order\_by('-mark\_num')

user\_ids = [user\_rate['user'] for user\_rate in users\_rate]

# user\_ids.append(user\_id)

users = User.objects.filter(id\_\_in=user\_ids)

for user in users:

rates = user.rate\_set.all()

rate = {}

# 用户有给电影打分 在rate和all\_user中进行设置

if rates:

for i in rates:

rate.setdefault(str(i.movie.id), i.mark)

all\_user.setdefault(user.username, rate)

else:

# 用户没有为电影打过分，设为0

all\_user.setdefault(user.username, {})

print('user\_recommend initial finished')

return all\_user

# 入口函数

def recommend\_by\_user\_id(user\_id):

user\_prefer = UserTagPrefer.objects.filter(user\_id=user\_id).order\_by('-score').values\_list('tag\_id', flat=True)

current\_user = User.objects.get(id=user\_id)

# 如果当前用户没有打分 则看是否选择过标签，选过的话，就从标签中找

# 没有的话，就按照浏览度推荐15个

if current\_user.rate\_set.count() == 0:

if len(user\_prefer) != 0:

movie\_list = Movie.objects.filter(tags\_\_in=user\_prefer)[:15]

else:

movie\_list = Movie.objects.order\_by("-num")[:15]

return movie\_list

import random

recommend\_list = [each[0] for each in user\_cf.recommend(current\_user.username, 15)]

movie\_list = list(Movie.objects.filter(id\_\_in=recommend\_list))[:15]

random.shuffle(movie\_list)

other\_length = 15 - len(movie\_list)

if other\_length > 0:

fix\_list = Movie.objects.filter(~Q(rate\_\_user\_id=user\_id)).order\_by('-favorite')

for fix in fix\_list:

if fix not in movie\_list:

movie\_list.append(fix)

if len(movie\_list) >= 15:

break

return movie\_list

# item\_based

class ItemBasedCF:

# 初始化参数

def \_\_init\_\_(self):

# 找到相似的20部电影，为目标用户推荐10部电影

self.n\_sim\_movie = 100

self.n\_rec\_movie = 15

# 用户相似度矩阵

self.movie\_sim\_matrix = defaultdict(lambda: defaultdict(float))

# 物品共现矩阵

self.cooccur = defaultdict(lambda: defaultdict(int))

self.movie\_popular = defaultdict(int)

self.movie\_count = 0

print('Similar user number = %d' % self.n\_sim\_movie)

print('Recommended user number = %d' % self.n\_rec\_movie)

self.calc\_movie\_sim()

# 计算电影之间的相似度

def calc\_movie\_sim(self):

model\_path = 'item\_rec.pkl'

# 已有的话，就不重新计算

# try:

# 重新计算

# except FileNotFoundError:

users = User.objects.all()

for user in users:

movies = Rate.objects.filter(user=user).values\_list('movie\_id', flat=True)

for movie in movies:

self.movie\_popular[movie] += 1

self.movie\_count = len(self.movie\_popular)

print("Total user number = %d" % self.movie\_count)

for user in users:

movies = Rate.objects.filter(user=user).values\_list('movie\_id', flat=True)

for m1 in movies:

for m2 in movies:

if m1 == m2:

continue

self.cooccur[m1][m2] += 1

# self.movie\_sim\_matrix[m1][m2] += 1

print("Build co-rated users matrix success!")

# 计算电影之间的相似性

print("Calculating user similarity matrix ...")

for m1, related\_movies in self.cooccur.items():

for m2, count in related\_movies.items():

# 注意0向量的处理，即某电影的用户数为0

if self.movie\_popular[m1] == 0 or self.movie\_popular[m2] == 0:

self.movie\_sim\_matrix[m1][m2] = 0

else:

# 根据公式计算w[i][j]

self.movie\_sim\_matrix[m1][m2] = count / sqrt(self.movie\_popular[m1] \* self.movie\_popular[m2])

print('Calculate user similarity matrix success!')

# 保存模型

with open(model\_path, 'wb')as opener:

pickle.dump(dict(self.movie\_sim\_matrix), opener)

print('保存模型成功!')

# 针对目标用户U，找到K部相似的电影，并推荐其N部电影

def recommend(self, user\_id):

K = self.n\_sim\_movie

N = self.n\_rec\_movie

rank = defaultdict(int)

# user = User.objects.get(id=user\_id)

watched\_movies = Rate.objects.filter(user\_id=user\_id).values\_list('movie\_id', 'mark')

print('this is watched movies', watched\_movies)

watched\_ids = [w[0] for w in watched\_movies]

try:

for movie, rating in watched\_movies:

for related\_movie, w in sorted(self.movie\_sim\_matrix[movie].items(), key=operator.itemgetter(1), reverse=True)[:K]:

# print('this is related', related\_movie)

if related\_movie in watched\_ids:

continue

rank[related\_movie] += w \* float(rating)

except KeyError:

self.calc\_movie\_sim()

return self.recommend(user\_id)

return sorted(rank.items(), key=operator.itemgetter(1), reverse=True)[:N]

# 在评分后去更新相似度矩阵

def update\_item\_movie\_sim\_matrix(movie\_id, user\_id):

# 更新电影被喜欢的num

item\_cf.movie\_popular[movie\_id] += 1

# 更新movie\_sim\_matrix

movies = Rate.objects.filter(user\_id=user\_id).values\_list('movie\_id', flat=True)

for m1 in movies:

if m1 == movie\_id:

continue

# 更新共现矩阵

item\_cf.cooccur[m1][movie\_id] += 1

item\_cf.cooccur[movie\_id][m1] += 1

# 重新计算相似度矩阵

for movie1\_id, count in item\_cf.cooccur[movie\_id].items():

if item\_cf.movie\_popular[movie1\_id] == 0 or item\_cf.movie\_popular[movie\_id] == 0:

item\_cf.movie\_sim\_matrix[movie\_id][movie1\_id] = 0

else:

# 根据公式计算w[i][j]

# 更新相似度矩阵

value = count / sqrt(item\_cf.movie\_popular[movie1\_id] \* item\_cf.movie\_popular[movie\_id])

item\_cf.movie\_sim\_matrix[movie1\_id][movie\_id] = value

item\_cf.movie\_sim\_matrix[movie\_id][movie1\_id] = value

print('update user similarity matrix success!')

# 在更新完成后重新写入本地

with open('item\_rec.pkl', 'wb')as opener:

pickle.dump(dict(item\_cf.movie\_sim\_matrix), opener)

print('保存更新成功!')

# 计算相似度

def similarity(movie1\_id, movie2\_id):

movie1\_set = Rate.objects.filter(movie\_id=movie1\_id)

# movie1的打分用户数

movie1\_sum = movie1\_set.count()

# movie\_2的打分用户数

movie2\_sum = Rate.objects.filter(movie\_id=movie2\_id).count()

# 两者的交集

common = Rate.objects.filter(user\_id\_\_in=Subquery(movie1\_set.values('user\_id')), movie=movie2\_id).values('user\_id').count()

# 没有人给当前电影打分

if movie1\_sum == 0 or movie2\_sum == 0:

return 0

similar\_value = common / sqrt(movie1\_sum \* movie2\_sum)

return similar\_value

#

def recommend\_by\_item\_id(user\_id, k=15):

# 前三的tag

user\_prefer = UserTagPrefer.objects.filter(user\_id=user\_id).order\_by('-score').values\_list('tag\_id', flat=True)

user\_prefer = list(user\_prefer)[:3]

current\_user = User.objects.get(id=user\_id)

# 如果当前用户没有打分 则看是否选择过标签，选过的话，就从标签中找

# 没有的话，就按照浏览度推荐15个

if current\_user.rate\_set.count() == 0:

if len(user\_prefer) != 0:

movie\_list = Movie.objects.filter(tags\_\_in=user\_prefer)[:15]

else:

movie\_list = Movie.objects.order\_by("-num")[:15]

print('from here')

return movie\_list

# most\_tags = Tags.objects.annotate(tags\_sum=Count('name')).order\_by('-tags\_sum').filter(movie\_\_rate\_\_user\_id=user\_id).order\_by('-tags\_sum')

# 选用户最喜欢的标签中的电影，用户没看过的30部，对这30部电影，计算距离最近

un\_watched = Movie.objects.filter(~Q(rate\_\_user\_id=user\_id), tags\_\_in=user\_prefer).order\_by('?')[:30] # 看过的电影

watched = Rate.objects.filter(user\_id=user\_id).values\_list('movie\_id', 'mark')

distances = []

names = []

# 在未看过的电影中找到

# 后续改进，选择top15

for un\_watched\_movie in un\_watched:

for watched\_movie in watched:

if un\_watched\_movie not in names:

names.append(un\_watched\_movie)

distances.append((similarity(un\_watched\_movie.id, watched\_movie[0]) \* watched\_movie[1], un\_watched\_movie))

distances.sort(key=lambda x: x[0], reverse=True)

print('this is distances', distances[:15])

recommend\_list = []

for mark, movie in distances:

if len(recommend\_list) >= k:

break

if movie not in recommend\_list:

recommend\_list.append(movie)

# print('this is recommend list', recommend\_list)

# 如果得不到有效数量的推荐 按照未看过的电影中的热度进行填充

print('recommend list', recommend\_list)

return recommend\_list

import sys

all\_user = get\_all\_user()

user\_cf = UserCf(all\_user=all\_user)

if 'runserver' in sys.argv:

# 在django启动时计算矩阵

item\_cf = ItemBasedCF()

## 6.7 电影图文展示代码

# 添加和读取数据到数据库中

import os

os.environ.setdefault("DJANGO\_SETTINGS\_MODULE", "movie.settings")

import asyncio

import aiohttp

import pandas as pd

async def write\_images(name, image\_link, image\_name):

print('write images....', image\_link)

async with aiohttp.ClientSession()as session:

async with session.get(image\_link) as response:

if response.status == 404:

print('404', name)

# assert response.status == 200

with open('movie\_images/' + image\_name, 'wb')as opener:

while True:

chunk = await response.content.read(1024)

if not chunk:

break

opener.write(chunk)

tasks = []

def envent\_loop():

df1 = pd.read\_csv('../csv\_data/movies\_2000.csv')

df2 = pd.read\_csv('../csv\_data/movies\_250.csv')

urls1 = df1['image\_link'].tolist()

urls2 = df2['image\_link'].tolist()

names1 = df1['title'].tolist()

names2 = df2['title'].tolist()

names1.extend(names2)

urls1.extend(urls2)

for name, url in zip(names1, urls1):

title = url.split('/')[-1]

if os.path.exists('movie\_images/' + title):

print('exists')

continue

print(url, title)

tasks.append(write\_images(name, url, title))

loop = asyncio.get\_event\_loop()

loop.run\_until\_complete(asyncio.wait(tasks))

if \_\_name\_\_ == '\_\_main\_\_':

envent\_loop()

# coding=utf-8

"""

@time: 2021/4/9 11:03 AM

@author: colaplusice

@contact: fjl2401@163.com vx:18392358995

"""

import os

os.environ.setdefault("DJANGO\_SETTINGS\_MODULE", "movie.settings")

import django

django.setup()

from user.models import Movie

movies = Movie.objects.all()

all\_images=os.listdir('../media/movie\_cover')

print(all\_images[:3])

for movie in movies:

s=str(movie.image\_link.name).split('/')[-1]

print(s)

all\_images.remove(s)

# if not os.path.exists('../media/' + str(movie.image\_link)):

# print(movie.image\_link)

print(all\_images)

## 6.8 前端代码

(function() {

'use strict';

var timeParsePatterns = [

// 9

{

re: /^\d{1,2}$/i,

handler: function(bits) {

if (bits[0].length === 1) {

return '0' + bits[0] + ':00';

} else {

return bits[0] + ':00';

}

}

},

// 13:00

{

re: /^\d{2}[:.]\d{2}$/i,

handler: function(bits) {

return bits[0].replace('.', ':');

}

},

// 9:00

{

re: /^\d[:.]\d{2}$/i,

handler: function(bits) {

return '0' + bits[0].replace('.', ':');

}

},

// 3 am / 3 a.m. / 3am

{

re: /^(\d+)\s\*([ap])(?:.?m.?)?$/i,

handler: function(bits) {

var hour = parseInt(bits[1]);

if (hour === 12) {

hour = 0;

}

if (bits[2].toLowerCase() === 'p') {

if (hour === 12) {

hour = 0;

}

return (hour + 12) + ':00';

} else {

if (hour < 10) {

return '0' + hour + ':00';

} else {

return hour + ':00';

}

}

}

},

// 3.30 am / 3:15 a.m. / 3.00am

{

re: /^(\d+)[.:](\d{2})\s\*([ap]).?m.?$/i,

handler: function(bits) {

var hour = parseInt(bits[1]);

var mins = parseInt(bits[2]);

if (mins < 10) {

mins = '0' + mins;

}

if (hour === 12) {

hour = 0;

}

if (bits[3].toLowerCase() === 'p') {

if (hour === 12) {

hour = 0;

}

return (hour + 12) + ':' + mins;

} else {

if (hour < 10) {

return '0' + hour + ':' + mins;

} else {

return hour + ':' + mins;

}

}

}

},

// noon

{

re: /^no/i,

handler: function(bits) {

return '12:00';

}

},

// midnight

{

re: /^mid/i,

handler: function(bits) {

return '00:00';

}

}

];

function parseTimeString(s) {

for (var i = 0; i < timeParsePatterns.length; i++) {

var re = timeParsePatterns[i].re;

var handler = timeParsePatterns[i].handler;

var bits = re.exec(s);

if (bits) {

return handler(bits);

}

}

return s;

}

window.parseTimeString = parseTimeString;

})();

/\*global SelectBox, gettext, interpolate, quickElement, SelectFilter\*/

/\*

SelectFilter2 - Turns a multiple-select box into a filter interface.

Requires jQuery, core.js, and SelectBox.js.

\*/

(function($) {

'use strict';

function findForm(node) {

// returns the node of the form containing the given node

if (node.tagName.toLowerCase() !== 'form') {

return findForm(node.parentNode);

}

return node;

}

window.SelectFilter = {

init: function(field\_id, field\_name, is\_stacked) {

if (field\_id.match(/\_\_prefix\_\_/)) {

// Don't initialize on empty forms.

return;

}

var from\_box = document.getElementById(field\_id);

from\_box.id += '\_from'; // change its ID

from\_box.className = 'filtered';

var ps = from\_box.parentNode.getElementsByTagName('p');

for (var i = 0; i < ps.length; i++) {

if (ps[i].className.indexOf("info") !== -1) {

// Remove <p class="info">, because it just gets in the way.

from\_box.parentNode.removeChild(ps[i]);

} else if (ps[i].className.indexOf("help") !== -1) {

// Move help text up to the top so it isn't below the select

// boxes or wrapped off on the side to the right of the add

// button:

from\_box.parentNode.insertBefore(ps[i], from\_box.parentNode.firstChild);

}

}

// <div class="selector"> or <div class="selector stacked">

var selector\_div = quickElement('div', from\_box.parentNode);

selector\_div.className = is\_stacked ? 'selector stacked' : 'selector';

// <div class="selector-available">

var selector\_available = quickElement('div', selector\_div);

selector\_available.className = 'selector-available';

var title\_available = quickElement('h2', selector\_available, interpolate(gettext('Available %s') + ' ', [field\_name]));

quickElement(

'span', title\_available, '',

'class', 'help help-tooltip help-icon',

'title', interpolate(

gettext(

'This is the list of available %s. You may choose some by ' +

'selecting them in the box below and then clicking the ' +

'"Choose" arrow between the two boxes.'

),

[field\_name]

)

);

var filter\_p = quickElement('p', selector\_available, '', 'id', field\_id + '\_filter');

filter\_p.className = 'selector-filter';

var search\_filter\_label = quickElement('label', filter\_p, '', 'for', field\_id + '\_input');

quickElement(

'span', search\_filter\_label, '',

'class', 'help-tooltip search-label-icon',

'title', interpolate(gettext("Type into this box to filter down the list of available %s."), [field\_name])

);

filter\_p.appendChild(document.createTextNode(' '));

var filter\_input = quickElement('input', filter\_p, '', 'type', 'text', 'placeholder', gettext("Filter"));

filter\_input.id = field\_id + '\_input';

selector\_available.appendChild(from\_box);

var choose\_all = quickElement('a', selector\_available, gettext('Choose all'), 'title', interpolate(gettext('Click to choose all %s at once.'), [field\_name]), 'href', '#', 'id', field\_id + '\_add\_all\_link');

choose\_all.className = 'selector-chooseall';

// <ul class="selector-chooser">

var selector\_chooser = quickElement('ul', selector\_div);

selector\_chooser.className = 'selector-chooser';

var add\_link = quickElement('a', quickElement('li', selector\_chooser), gettext('Choose'), 'title', gettext('Choose'), 'href', '#', 'id', field\_id + '\_add\_link');

add\_link.className = 'selector-add';

var remove\_link = quickElement('a', quickElement('li', selector\_chooser), gettext('Remove'), 'title', gettext('Remove'), 'href', '#', 'id', field\_id + '\_remove\_link');

remove\_link.className = 'selector-remove';

// <div class="selector-chosen">

var selector\_chosen = quickElement('div', selector\_div);

selector\_chosen.className = 'selector-chosen';

var title\_chosen = quickElement('h2', selector\_chosen, interpolate(gettext('Chosen %s') + ' ', [field\_name]));

quickElement(

'span', title\_chosen, '',

'class', 'help help-tooltip help-icon',

'title', interpolate(

gettext(

'This is the list of chosen %s. You may remove some by ' +

'selecting them in the box below and then clicking the ' +

'"Remove" arrow between the two boxes.'

),

[field\_name]

)

);

var to\_box = quickElement('select', selector\_chosen, '', 'id', field\_id + '\_to', 'multiple', 'multiple', 'size', from\_box.size, 'name', from\_box.getAttribute('name'));

to\_box.className = 'filtered';

var clear\_all = quickElement('a', selector\_chosen, gettext('Remove all'), 'title', interpolate(gettext('Click to remove all chosen %s at once.'), [field\_name]), 'href', '#', 'id', field\_id + '\_remove\_all\_link');

clear\_all.className = 'selector-clearall';

from\_box.setAttribute('name', from\_box.getAttribute('name') + '\_old');

// Set up the JavaScript event handlers for the select box filter interface

var move\_selection = function(e, elem, move\_func, from, to) {

if (elem.className.indexOf('active') !== -1) {

move\_func(from, to);

SelectFilter.refresh\_icons(field\_id);

}

e.preventDefault();

};

choose\_all.addEventListener('click', function(e) {

move\_selection(e, this, SelectBox.move\_all, field\_id + '\_from', field\_id + '\_to');

});

add\_link.addEventListener('click', function(e) {

move\_selection(e, this, SelectBox.move, field\_id + '\_from', field\_id + '\_to');

});

remove\_link.addEventListener('click', function(e) {

move\_selection(e, this, SelectBox.move, field\_id + '\_to', field\_id + '\_from');

});

clear\_all.addEventListener('click', function(e) {

move\_selection(e, this, SelectBox.move\_all, field\_id + '\_to', field\_id + '\_from');

});

filter\_input.addEventListener('keypress', function(e) {

SelectFilter.filter\_key\_press(e, field\_id);

});

filter\_input.addEventListener('keyup', function(e) {

SelectFilter.filter\_key\_up(e, field\_id);

});

filter\_input.addEventListener('keydown', function(e) {

SelectFilter.filter\_key\_down(e, field\_id);

});

selector\_div.addEventListener('change', function(e) {

if (e.target.tagName === 'SELECT') {

SelectFilter.refresh\_icons(field\_id);

}

});

selector\_div.addEventListener('dblclick', function(e) {

if (e.target.tagName === 'OPTION') {

if (e.target.closest('select').id === field\_id + '\_to') {

SelectBox.move(field\_id + '\_to', field\_id + '\_from');

} else {

SelectBox.move(field\_id + '\_from', field\_id + '\_to');

}

SelectFilter.refresh\_icons(field\_id);

}

});

findForm(from\_box).addEventListener('submit', function() {

SelectBox.select\_all(field\_id + '\_to');

});

SelectBox.init(field\_id + '\_from');

SelectBox.init(field\_id + '\_to');

// Move selected from\_box options to to\_box

SelectBox.move(field\_id + '\_from', field\_id + '\_to');

if (!is\_stacked) {

// In horizontal mode, give the same height to the two boxes.

var j\_from\_box = $('#' + field\_id + '\_from');

var j\_to\_box = $('#' + field\_id + '\_to');

var resize\_filters = function() { j\_to\_box.height($(filter\_p).outerHeight() + j\_from\_box.outerHeight()); };

if (j\_from\_box.outerHeight() > 0) {

resize\_filters(); // This fieldset is already open. Resize now.

} else {

// This fieldset is probably collapsed. Wait for its 'show' event.

j\_to\_box.closest('fieldset').one('show.fieldset', resize\_filters);

}

}

// Initial icon refresh

SelectFilter.refresh\_icons(field\_id);

},

any\_selected: function(field) {

var any\_selected = false;

try {

// Temporarily add the required attribute and check validity.

// This is much faster in WebKit browsers than the fallback.

field.attr('required', 'required');

any\_selected = field.is(':valid');

field.removeAttr('required');

} catch (e) {

// Browsers that don't support :valid (IE < 10)

any\_selected = field.find('option:selected').length > 0;

}

return any\_selected;

},

refresh\_icons: function(field\_id) {

var from = $('#' + field\_id + '\_from');

var to = $('#' + field\_id + '\_to');

// Active if at least one item is selected

$('#' + field\_id + '\_add\_link').toggleClass('active', SelectFilter.any\_selected(from));

$('#' + field\_id + '\_remove\_link').toggleClass('active', SelectFilter.any\_selected(to));

// Active if the corresponding box isn't empty

$('#' + field\_id + '\_add\_all\_link').toggleClass('active', from.find('option').length > 0);

$('#' + field\_id + '\_remove\_all\_link').toggleClass('active', to.find('option').length > 0);

},

filter\_key\_press: function(event, field\_id) {

var from = document.getElementById(field\_id + '\_from');

// don't submit form if user pressed Enter

if ((event.which && event.which === 13) || (event.keyCode && event.keyCode === 13)) {

from.selectedIndex = 0;

SelectBox.move(field\_id + '\_from', field\_id + '\_to');

from.selectedIndex = 0;

event.preventDefault();

return false;

}

},

filter\_key\_up: function(event, field\_id) {

var from = document.getElementById(field\_id + '\_from');

var temp = from.selectedIndex;

SelectBox.filter(field\_id + '\_from', document.getElementById(field\_id + '\_input').value);

from.selectedIndex = temp;

return true;

},

filter\_key\_down: function(event, field\_id) {

var from = document.getElementById(field\_id + '\_from');

// right arrow -- move across

if ((event.which && event.which === 39) || (event.keyCode && event.keyCode === 39)) {

var old\_index = from.selectedIndex;

SelectBox.move(field\_id + '\_from', field\_id + '\_to');

from.selectedIndex = (old\_index === from.length) ? from.length - 1 : old\_index;

return false;

}

// down arrow -- wrap around

if ((event.which && event.which === 40) || (event.keyCode && event.keyCode === 40)) {

from.selectedIndex = (from.length === from.selectedIndex + 1) ? 0 : from.selectedIndex + 1;

}

// up arrow -- wrap around

if ((event.which && event.which === 38) || (event.keyCode && event.keyCode === 38)) {

from.selectedIndex = (from.selectedIndex === 0) ? from.length - 1 : from.selectedIndex - 1;

}

return true;

}

};

window.addEventListener('load', function(e) {

$('select.selectfilter, select.selectfilterstacked').each(function() {

var $el = $(this),

data = $el.data();

SelectFilter.init($el.attr('id'), data.fieldName, parseInt(data.isStacked, 10));

});

});

})(django.jQuery);

/\*global XRegExp\*/

(function() {

'use strict';

var ALL\_DOWNCODE\_MAPS = [

LATIN\_MAP,

LATIN\_SYMBOLS\_MAP,

GREEK\_MAP,

TURKISH\_MAP,

ROMANIAN\_MAP,

RUSSIAN\_MAP,

UKRAINIAN\_MAP,

CZECH\_MAP,

SLOVAK\_MAP,

POLISH\_MAP,

LATVIAN\_MAP,

ARABIC\_MAP,

LITHUANIAN\_MAP,

SERBIAN\_MAP,

AZERBAIJANI\_MAP,

GEORGIAN\_MAP

];

var Downcoder = {

'Initialize': function() {

if (Downcoder.map) { // already made

return;

}

Downcoder.map = {};

Downcoder.chars = [];

for (var i = 0; i < ALL\_DOWNCODE\_MAPS.length; i++) {

var lookup = ALL\_DOWNCODE\_MAPS[i];

for (var c in lookup) {

if (lookup.hasOwnProperty(c)) {

Downcoder.map[c] = lookup[c];

}

}

}

for (var k in Downcoder.map) {

if (Downcoder.map.hasOwnProperty(k)) {

Downcoder.chars.push(k);

}

}

Downcoder.regex = new RegExp(Downcoder.chars.join('|'), 'g');

}

};

function downcode(slug) {

Downcoder.Initialize();

return slug.replace(Downcoder.regex, function(m) {

return Downcoder.map[m];

});

}

function URLify(s, num\_chars, allowUnicode) {

// changes, e.g., "Petty theft" to "petty-theft"

// remove all these words from the string before urlifying

if (!allowUnicode) {

s = downcode(s);

}

var hasUnicodeChars = /[^\u0000-\u007f]/.test(s);

// Remove English words only if the string contains ASCII (English)

// characters.

if (!hasUnicodeChars) {

var removeList = [

"a", "an", "as", "at", "before", "but", "by", "for", "from",

"is", "in", "into", "like", "of", "off", "on", "onto", "per",

"since", "than", "the", "this", "that", "to", "up", "via",

"with"

];

var r = new RegExp('\\b(' + removeList.join('|') + ')\\b', 'gi');

s = s.replace(r, '');

}

// if downcode doesn't hit, the char will be stripped here

if (allowUnicode) {

// Keep Unicode letters including both lowercase and uppercase

// characters, whitespace, and dash; remove other characters.

s = XRegExp.replace(s, XRegExp('[^-\_\\p{L}\\p{N}\\s]', 'g'), '');

} else {

s = s.replace(/[^-\w\s]/g, ''); // remove unneeded chars

}

s = s.replace(/^\s+|\s+$/g, ''); // trim leading/trailing spaces

s = s.replace(/[-\s]+/g, '-'); // convert spaces to hyphens

s = s.substring(0, num\_chars); // trim to first num\_chars chars

s = s.replace(/-+$/g, ''); // trim any trailing hyphens

return s.toLowerCase(); // convert to lowercase

}

window.URLify = URLify;

})();

/\*!

\* jQuery Raty - A Star Rating Plugin

\*

\* The MIT License

\*

\* author: Washington Botelho

\* github: wbotelhos/raty

\* version: 2.8.0

\*

\*/

(function($) {

'use strict';

var methods = {

init: function(options) {

return this.each(function() {

this.self = $(this);

methods.destroy.call(this.self);

this.opt = $.extend(true, {}, $.fn.raty.defaults, options, this.self.data());

methods.\_adjustCallback.call(this);

methods.\_adjustNumber.call(this);

methods.\_adjustHints.call(this);

this.opt.score = methods.\_adjustedScore.call(this, this.opt.score);

if (this.opt.starType !== 'img') {

methods.\_adjustStarType.call(this);

}

methods.\_adjustPath.call(this);

methods.\_createStars.call(this);

if (this.opt.cancel) {

methods.\_createCancel.call(this);

}

if (this.opt.precision) {

methods.\_adjustPrecision.call(this);

}

methods.\_createScore.call(this);

methods.\_apply.call(this, this.opt.score);

methods.\_setTitle.call(this, this.opt.score);

methods.\_target.call(this, this.opt.score);

if (this.opt.readOnly) {

methods.\_lock.call(this);

} else {

this.style.cursor = 'pointer';

methods.\_binds.call(this);

}

});

},

\_adjustCallback: function() {

var options = ['number', 'readOnly', 'score', 'scoreName', 'target', 'path'];

for (var i = 0; i < options.length; i++) {

if (typeof this.opt[options[i]] === 'function') {

this.opt[options[i]] = this.opt[options[i]].call(this);

}

}

},

\_adjustedScore: function(score) {

if (!score) {

return score;

}

return methods.\_between(score, 0, this.opt.number);

},

\_adjustHints: function() {

if (!this.opt.hints) {

this.opt.hints = [];

}

if (!this.opt.halfShow && !this.opt.half) {

return;

}

var steps = this.opt.precision ? 10 : 2;

for (var i = 0; i < this.opt.number; i++) {

var group = this.opt.hints[i];

if (Object.prototype.toString.call(group) !== '[object Array]') {

group = [group];

}

this.opt.hints[i] = [];

for (var j = 0; j < steps; j++) {

var

hint = group[j],

last = group[group.length - 1];

if (last === undefined) {

last = null;

}

this.opt.hints[i][j] = hint === undefined ? last : hint;

}

}

},

\_adjustNumber: function() {

this.opt.number = methods.\_between(this.opt.number, 1, this.opt.numberMax);

},

\_adjustPath: function() {

this.opt.path = this.opt.path || '';

if (this.opt.path && this.opt.path.charAt(this.opt.path.length - 1) !== '/') {

this.opt.path += '/';

}

},

\_adjustPrecision: function() {

this.opt.half = true;

},

\_adjustStarType: function() {

var replaces = ['cancelOff', 'cancelOn', 'starHalf', 'starOff', 'starOn'];

this.opt.path = '';

for (var i = 0; i < replaces.length; i++) {

this.opt[replaces[i]] = this.opt[replaces[i]].replace('.', '-');

}

},

\_apply: function(score) {

methods.\_fill.call(this, score);

if (score) {

if (score > 0) {

this.score.val(score);

}

methods.\_roundStars.call(this, score);

}

},

\_between: function(value, min, max) {

return Math.min(Math.max(parseFloat(value), min), max);

},

\_binds: function() {

if (this.cancel) {

methods.\_bindOverCancel.call(this);

methods.\_bindClickCancel.call(this);

methods.\_bindOutCancel.call(this);

}

methods.\_bindOver.call(this);

methods.\_bindClick.call(this);

methods.\_bindOut.call(this);

},

\_bindClick: function() {

var that = this;

that.stars.on('click.raty', function(evt) {

var

execute = true,

score = (that.opt.half || that.opt.precision) ? that.self.data('score') : (this.alt || $(this).data('alt'));

if (that.opt.click) {

execute = that.opt.click.call(that, +score, evt);

}

if (execute || execute === undefined) {

if (that.opt.half && !that.opt.precision) {

score = methods.\_roundHalfScore.call(that, score);

}

methods.\_apply.call(that, score);

}

});

},

\_bindClickCancel: function() {

var that = this;

that.cancel.on('click.raty', function(evt) {

that.score.removeAttr('value');

if (that.opt.click) {

that.opt.click.call(that, null, evt);

}

});

},

\_bindOut: function() {

var that = this;

that.self.on('mouseleave.raty', function(evt) {

var score = +that.score.val() || undefined;

methods.\_apply.call(that, score);

methods.\_target.call(that, score, evt);

methods.\_resetTitle.call(that);

if (that.opt.mouseout) {

that.opt.mouseout.call(that, score, evt);

}

});

},

\_bindOutCancel: function() {

var that = this;

that.cancel.on('mouseleave.raty', function(evt) {

var icon = that.opt.cancelOff;

if (that.opt.starType !== 'img') {

icon = that.opt.cancelClass + ' ' + icon;

}

methods.\_setIcon.call(that, this, icon);

if (that.opt.mouseout) {

var score = +that.score.val() || undefined;

that.opt.mouseout.call(that, score, evt);

}

});

},

\_bindOver: function() {

var

that = this,

action = that.opt.half ? 'mousemove.raty' : 'mouseover.raty';

that.stars.on(action, function(evt) {

var score = methods.\_getScoreByPosition.call(that, evt, this);

methods.\_fill.call(that, score);

if (that.opt.half) {

methods.\_roundStars.call(that, score, evt);

methods.\_setTitle.call(that, score, evt);

that.self.data('score', score);

}

methods.\_target.call(that, score, evt);

if (that.opt.mouseover) {

that.opt.mouseover.call(that, score, evt);

}

});

},

\_bindOverCancel: function() {

var that = this;

that.cancel.on('mouseover.raty', function(evt) {

var

starOff = that.opt.path + that.opt.starOff,

icon = that.opt.cancelOn;

if (that.opt.starType === 'img') {

that.stars.attr('src', starOff);

} else {

icon = that.opt.cancelClass + ' ' + icon;

that.stars.attr('class', starOff);

}

methods.\_setIcon.call(that, this, icon);

methods.\_target.call(that, null, evt);

if (that.opt.mouseover) {

that.opt.mouseover.call(that, null);

}

});

},

\_buildScoreField: function() {

return $('<input />', { name: this.opt.scoreName, type: 'hidden' }).appendTo(this);

},

\_createCancel: function() {

var

icon = this.opt.path + this.opt.cancelOff,

cancel = $('<' + this.opt.starType + ' />', { title: this.opt.cancelHint, 'class': this.opt.cancelClass });

if (this.opt.starType === 'img') {

cancel.attr({ src: icon, alt: 'x' });

} else {

// TODO: use $.data

cancel.attr('data-alt', 'x').addClass(icon);

}

if (this.opt.cancelPlace === 'left') {

this.self.prepend('&#160;').prepend(cancel);

} else {

this.self.append('&#160;').append(cancel);

}

this.cancel = cancel;

},

\_createScore: function() {

var score = $(this.opt.targetScore);

this.score = score.length ? score : methods.\_buildScoreField.call(this);

},

\_createStars: function() {

for (var i = 1; i <= this.opt.number; i++) {

var

name = methods.\_nameForIndex.call(this, i),

attrs = { alt: i, src: this.opt.path + this.opt[name] };

if (this.opt.starType !== 'img') {

attrs = { 'data-alt': i, 'class': attrs.src }; // TODO: use $.data.

}

attrs.title = methods.\_getHint.call(this, i);

$('<' + this.opt.starType + ' />', attrs).appendTo(this);

if (this.opt.space) {

this.self.append(i < this.opt.number ? '&#160;' : '');

}

}

this.stars = this.self.children(this.opt.starType);

},

\_error: function(message) {

$(this).text(message);

$.error(message);

},

\_fill: function(score) {

var hash = 0;

for (var i = 1; i <= this.stars.length; i++) {

var

icon,

star = this.stars[i - 1],

turnOn = methods.\_turnOn.call(this, i, score);

if (this.opt.iconRange && this.opt.iconRange.length > hash) {

var irange = this.opt.iconRange[hash];

icon = methods.\_getRangeIcon.call(this, irange, turnOn);

if (i <= irange.range) {

methods.\_setIcon.call(this, star, icon);

}

if (i === irange.range) {

hash++;

}

} else {

icon = this.opt[turnOn ? 'starOn' : 'starOff'];

methods.\_setIcon.call(this, star, icon);

}

}

},

\_getFirstDecimal: function(number) {

var

decimal = number.toString().split('.')[1],

result = 0;

if (decimal) {

result = parseInt(decimal.charAt(0), 10);

if (decimal.slice(1, 5) === '9999') {

result++;

}

}

return result;

},

\_getRangeIcon: function(irange, turnOn) {

return turnOn ? irange.on || this.opt.starOn : irange.off || this.opt.starOff;

},

\_getScoreByPosition: function(evt, icon) {

var score = parseInt(icon.alt || icon.getAttribute('data-alt'), 10);

if (this.opt.half) {

var

size = methods.\_getWidth.call(this),

percent = parseFloat((evt.pageX - $(icon).offset().left) / size);

score = score - 1 + percent;

}

return score;

},

\_getHint: function(score, evt) {

if (score !== 0 && !score) {

return this.opt.noRatedMsg;

}

var

decimal = methods.\_getFirstDecimal.call(this, score),

integer = Math.ceil(score),

group = this.opt.hints[(integer || 1) - 1],

hint = group,

set = !evt || this.move;

if (this.opt.precision) {

if (set) {

decimal = decimal === 0 ? 9 : decimal - 1;

}

hint = group[decimal];

} else if (this.opt.halfShow || this.opt.half) {

decimal = set && decimal === 0 ? 1 : decimal > 5 ? 1 : 0;

hint = group[decimal];

}

return hint === '' ? '' : hint || score;

},

\_getWidth: function() {

var width = this.stars[0].width || parseFloat(this.stars.eq(0).css('font-size'));

if (!width) {

methods.\_error.call(this, 'Could not get the icon width!');

}

return width;

},

\_lock: function() {

var hint = methods.\_getHint.call(this, this.score.val());

this.style.cursor = '';

this.title = hint;

this.score.prop('readonly', true);

this.stars.prop('title', hint);

if (this.cancel) {

this.cancel.hide();

}

this.self.data('readonly', true);

},

\_nameForIndex: function(i) {

return this.opt.score && this.opt.score >= i ? 'starOn' : 'starOff';

},

\_resetTitle: function() {

for (var i = 0; i < this.opt.number; i++) {

this.stars[i].title = methods.\_getHint.call(this, i + 1);

}

},

\_roundHalfScore: function(score) {

var

integer = parseInt(score, 10),

decimal = methods.\_getFirstDecimal.call(this, score);

if (decimal !== 0) {

decimal = decimal > 5 ? 1 : 0.5;

}

return integer + decimal;

},

\_roundStars: function(score, evt) {

var

decimal = (score % 1).toFixed(2),

name ;

if (evt || this.move) {

name = decimal > 0.5 ? 'starOn' : 'starHalf';

} else if (decimal > this.opt.round.down) { // Up: [x.76 .. x.99]

name = 'starOn';

if (this.opt.halfShow && decimal < this.opt.round.up) { // Half: [x.26 .. x.75]

name = 'starHalf';

} else if (decimal < this.opt.round.full) { // Down: [x.00 .. x.5]

name = 'starOff';

}

}

if (name) {

var

icon = this.opt[name],

star = this.stars[Math.ceil(score) - 1];

methods.\_setIcon.call(this, star, icon);

} // Full down: [x.00 .. x.25]

},

\_setIcon: function(star, icon) {

star[this.opt.starType === 'img' ? 'src' : 'className'] = this.opt.path + icon;

},

\_setTarget: function(target, score) {

if (score) {

score = this.opt.targetFormat.toString().replace('{score}', score);

}

if (target.is(':input')) {

target.val(score);

} else {

target.html(score);

}

},

\_setTitle: function(score, evt) {

if (score) {

var

integer = parseInt(Math.ceil(score), 10),

star = this.stars[integer - 1];

star.title = methods.\_getHint.call(this, score, evt);

}

},

\_target: function(score, evt) {

if (this.opt.target) {

var target = $(this.opt.target);

if (!target.length) {

methods.\_error.call(this, 'Target selector invalid or missing!');

}

var mouseover = evt && evt.type === 'mouseover';

if (score === undefined) {

score = this.opt.targetText;

} else if (score === null) {

score = mouseover ? this.opt.cancelHint : this.opt.targetText;

} else {

if (this.opt.targetType === 'hint') {

score = methods.\_getHint.call(this, score, evt);

} else if (this.opt.precision) {

score = parseFloat(score).toFixed(1);

}

var mousemove = evt && evt.type === 'mousemove';

if (!mouseover && !mousemove && !this.opt.targetKeep) {

score = this.opt.targetText;

}

}

methods.\_setTarget.call(this, target, score);

}

},

\_turnOn: function(i, score) {

return this.opt.single ? (i === score) : (i <= score);

},

\_unlock: function() {

this.style.cursor = 'pointer';

this.removeAttribute('title');

this.score.removeAttr('readonly');

this.self.data('readonly', false);

for (var i = 0; i < this.opt.number; i++) {

this.stars[i].title = methods.\_getHint.call(this, i + 1);

}

if (this.cancel) {

this.cancel.css('display', '');

}

},

cancel: function(click) {

return this.each(function() {

var self = $(this);

if (self.data('readonly') !== true) {

methods[click ? 'click' : 'score'].call(self, null);

this.score.removeAttr('value');

}

});

},

click: function(score) {

return this.each(function() {

if ($(this).data('readonly') !== true) {

score = methods.\_adjustedScore.call(this, score);

methods.\_apply.call(this, score);

if (this.opt.click) {

this.opt.click.call(this, score, $.Event('click'));

}

methods.\_target.call(this, score);

}

});

},

destroy: function() {

return this.each(function() {

var

self = $(this),

raw = self.data('raw');

if (raw) {

self.off('.raty').empty().css({ cursor: raw.style.cursor }).removeData('readonly');

} else {

self.data('raw', self.clone()[0]);

}

});

},

getScore: function() {

var

score = [],

value ;

this.each(function() {

value = this.score.val();

score.push(value ? +value : undefined);

});

return (score.length > 1) ? score : score[0];

},

move: function(score) {

return this.each(function() {

var

integer = parseInt(score, 10),

decimal = methods.\_getFirstDecimal.call(this, score);

if (integer >= this.opt.number) {

integer = this.opt.number - 1;

decimal = 10;

}

var

width = methods.\_getWidth.call(this),

steps = width / 10,

star = $(this.stars[integer]),

percent = star.offset().left + steps \* decimal,

evt = $.Event('mousemove', { pageX: percent });

this.move = true;

star.trigger(evt);

this.move = false;

});

},

readOnly: function(readonly) {

return this.each(function() {

var self = $(this);

if (self.data('readonly') !== readonly) {

if (readonly) {

self.off('.raty').children(this.opt.starType).off('.raty');

methods.\_lock.call(this);

} else {

methods.\_binds.call(this);

methods.\_unlock.call(this);

}

self.data('readonly', readonly);

}

});

},

reload: function() {

return methods.set.call(this, {});

},

score: function() {

var self = $(this);

return arguments.length ? methods.setScore.apply(self, arguments) : methods.getScore.call(self);

},

set: function(options) {

return this.each(function() {

$(this).raty($.extend({}, this.opt, options));

});

},

setScore: function(score) {

return this.each(function() {

if ($(this).data('readonly') !== true) {

score = methods.\_adjustedScore.call(this, score);

methods.\_apply.call(this, score);

methods.\_target.call(this, score);

}

});

}

};

$.fn.raty = function(method) {

if (methods[method]) {

return methods[method].apply(this, Array.prototype.slice.call(arguments, 1));

} else if (typeof method === 'object' || !method) {

return methods.init.apply(this, arguments);

} else {

$.error('Method ' + method + ' does not exist!');

}

};

$.fn.raty.defaults = {

cancel: false,

cancelClass: 'raty-cancel',

cancelHint: 'Cancel this rating!',

cancelOff: 'cancel-off.png',

cancelOn: 'cancel-on.png',

cancelPlace: 'left',

click: undefined,

half: false,

halfShow: true,

hints: ['bad', 'poor', 'regular', 'good', 'gorgeous'],

iconRange: undefined,

mouseout: undefined,

mouseover: undefined,

noRatedMsg: 'Not rated yet!',

number: 5,

numberMax: 20,

path: undefined,

precision: false,

readOnly: false,

round: { down: 0.25, full: 0.6, up: 0.76 },

score: undefined,

scoreName: 'score',

single: false,

space: true,

starHalf: 'star-half.png',

starOff: 'star-off.png',

starOn: 'star-on.png',

starType: 'img',

target: undefined,

targetFormat: '{score}',

targetKeep: false,

targetScore: undefined,

targetText: '',

targetType: 'hint'

};

})(jQuery);

# 总 结

本电影推荐系统的设计与实现是由各个阶段逐步进行的，主要有团队建设、需求分析、系统设计与实现，其中需求分析主要进行了功能分析和性能分析，并使用各种图表进行更为直观的进一步表示，是系统设计与实现过程中基础的重要的一步。系统设计阶段主要通过推荐算法的选择、设计原则、系统架构、功能模块等方面进行对该电影推荐系统的设计，采用主流的推荐算法以及大数据平台进行后续的开发及实现，并提出相应的改进策略。详细设计通过系统界面展示的形式进行说明，包括功能描述、界面视图展示及核心代码。本文采用的大数据处理平台是Python，再结合本文提出的算法，为后续离线、在线以及热门进行推荐。

本系统结合现有的较热门的MVC框架，采用两种不同推荐算法实现对用户的个性化电影推荐，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。不过，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，后期可以考虑尝试其他推荐算法，采用更新、效率更高的推荐算法。另外，数据集为固定的数据集，可考虑加入爬虫技术以实时更新数据集，使用户能获取最新的电影观看。

总体来说，本电影个性化推荐系统采用了两种协同过滤算法：基于用户的协同过滤和基于项目的协同过滤，使用户能从两种不同角度获得推荐，提高用户的可选择性。设计的系统包含用户信息功能模块、电影展示功能模块、用户操作功能模块、推荐结果模块及管理员后台管理模块，简明清晰，易操作。

# 参考文献

[1] 蔡骐,李玲.信息过载时代的新媒介素养[J].现代传播(中国传媒大学学 报),2013,35(09):120-124.

[2] Ghemawat S, Gobioff H, Leung S T. The Google file system[C]//ACMSIGOPS Operating Systems Review. ACM, 2003, 37(5): 29-43.

[3] Dean J,Ghemawat S.MapReduce:simplified data processing on large clusters[J].Communications of the ACM, 2008, 51(1): 107-113.

[4] 孙辉,马跃,杨海波,张红松.一种相似度改进的用户聚类协同过滤推荐算法[J].小型微型计算机系统,2014,35(09):1967-1970.

[5] 印鉴,陈忆群,张钢.搜索引擎技术研究与发展[J].计算机工程,2005(14):54-56+104.

[6] 许明杰,蔚承建,沈航.Spark 并行化基于物品协同过滤算法[J].计算机工程与设计,2017,38(07):1817-1822.

[7] 周志华,杨强.机器学习及其应用[M].北京:清华大学出版社,2011.

[8] JiaWei Han,Micheline Kamber.数据挖掘概念与技术[M].范明,译.机械工业出版社,2001.

[9] Y.Carbonne and C. Jacob. Genetic Algorithm as Machine Learning for profiles recognition[J].2015 7th International Joint Conference on Computational Intelligence (IJCCI), Lisbon, Portugal,2015:157-166.

[10] 唐振坤.基于 Spark 的机器学习平台设计与实现[D].厦门大学,2014.

[11] K. Dwivedi and S. K. Dubey. Analytical review on Hadoop Distributed file system[J].Confluence The Next Generation Information Technology Summit (Confluence), 2014 5th International Conference -, Noida, 2014:175-181.

[12] Shvachko K, Kuang H, Radia S,et al. The hadoop distributed file system[C]//Mass Storage Systems and Technologies (MSST),2010 IEEE 26th Symposium on. IEEE, 2010: 1-10.

[13] 王升升,赵海燕,陈庆奎,曹健.个性化推荐中的隐语义模型[J].小型微型计算机系统,2016,37(05):881-889.

[14] 杨阳,向阳,熊磊.基于矩阵分解与用户近邻模型的协同过滤推荐算法[J].计算机应用,2012,32(2):395-398.

[15] F. Ullah, G. Sarwar, S.C. Lee, Y.K. Park, K.D. Moon, J.T. Kim, Hybrid recommender system with temporal information, Proceedings of IEEE International Conference on Information Networking (2012) 421-425.

[16] 吴金龙.Netflix Prize中的协同过滤算法[D].北京:北京大学,2010.

[17] 胡俊,胡贤德,程家兴.基于Spark的大数据混合计算模型[J].计算机系统应用,2015,24(4)215-218.

[18] 李改,李磊.基于矩阵分解的协同过滤算法[J].计算机工程与应用,2011,47(30): 5-7.

[19] Guo G,Wang H,Bell D,Bi Y,Greer K. KNN model-based approach in classification[M]. On The Move to Meaningful Internet Systems 2003:986-996.

[20] George T, Merugu S. A scalable collaborative filtering framework based on co-clustering[C].//IEEE International Conference on Data Mining.IEEE, 2005: 625-628.

[21] 杨志伟.基于 Spark 平台推荐系统研究[D].中国科学技术大学,2015.

[22] 冯琳.集群计算引擎 Spark 中的内存优化研究与实现[D].清华大学,2013.