山西应用科技学院
SHANXI COLLEGE OF APPLIED SCIENCE AND TECHNOLOGY

大数据综合课程设计

题　　目:	基于 Django 的电影评论数据可视化分析
学　　院:	信息工程学院
专业班级:	大数据管理与应用 2101 班
姓　　名:	郭未
学　　号:	2113051011
指导教师:	高美玲

山西应用科技学院

二〇二四年六月

目　录

1 绪论..... 1
2 研究目的..... 1
3 开发技术介绍..... 2
4 软件架构的描述..... 4
4．1 分解视图 4
4．2 依赖视图 5
4．3.执行视图..... 5
4．4.实现视图..... 6
4．5.工作分配视图..... 7
5．设计模式..... 8
6．数据库设计..... 9
7．系统开发环境和技术选型..... 10
8．概念原型的核心工作机制..... 10
9 结论..... 11
参考文献..... 12
致　　谢..... 13
附　　录..... 14

基于 Django 的电影评论数据可视化分析

1 绪论

随着大数据技术的不断发展和普及，人们在日常生活中产生的数据量呈爆炸性增长。电影评论数据作为一种丰富的信息源，包含了观众对电影的各种评价和喜好。在这个信息爆炸的时代，如何从海量的电影评论中提炼有价值的信息，为用户提供更智能、个性化的电影推荐服务成为一个备受关注的问题。

本项目选取豆瓣作为数据源，结合 Python 和 Django 等先进技术，构建了一个综合性的豆瓣电影评论可视化分析推荐系统。通过对大规模评论数据的采集和处理，我们能够深入挖掘用户的观影趋势、口碑评价等信息。在这个基础上，利用数据可视化技术，以直观的图表和图形展示用户的观影偏好，为用户提供了更深入的电影分析服务。

该项目旨在结合大数据、可视化和推荐系统的技术优势，为电影爱好者提供一种全新的电影探索和选择方式，提升用户体验。通过对豆瓣电影评论数据的深度挖掘，我们能够更好地理解用户的需求，为他们提供更精准、个性化的电影推荐，推动了电影推荐系统的发展和创新。同时，项目的实施也展示了 Python/Django 等技术在构建复杂大数据系统中的卓越应用，为相关领域的研究和应用提供了有益的经验。

2 研究目的

1.深入挖掘电影评论数据： 通过构建基于 Python/Django 的豆瓣电影评论可视化分析推荐系统，旨在深入挖掘电影评论数据中蕴含的用户偏好、口碑评价等信息。通过对评论数据的系统性分析，揭示用户对电影的喜好和趋势。

2.构建全面的电影信息数据库： 通过爬取豆瓣电影评论数据，进行数据清洗和处理，构建一个全面而准确的电影信息数据库。该数据库将包含丰富的电影元数据，为系统提供充足的信息基础，支持后续的分析和推荐。

3.实现数据可视化展示： 利用 Python 中强大的数据处理和可视化库，将分析结

果以直观的图表、图形展示给用户。通过直观的可视化展示，使用户更容易理解电影数据背后的信息，为用户提供更深入的电影分析服务。

4.设计智能化的电影推荐算法： 基于对电影评论数据的深度分析，设计智能化的推荐算法。通过考虑用户的历史喜好、观影习惯等因素，为用户提供个性化、精准的电影推荐服务，提升用户体验。

5.展示 Python/Django 在大数据应用中的优越性：通过该项目的实施，展示 Python 和 Django 等先进技术在大数据应用中的卓越性能[1]。强调这些技术在构建复杂系统、处理大规模数据时的高效性和可扩展性，为相关领域的研究和应用提供实用经验[2]。

总体而言，研究旨在通过构建综合性的电影评论可视化分析推荐系统，挖掘电影评论数据的潜在价值，提升用户对电影的选择和理解体验，同时突显 Python/Django 等技术在大数据领域的应用前景。

3 开发技术介绍

Django（发音为"jan-go"）是一个高级的 Python web 框架，它鼓励快速开发和干净、可重用的设计。以下是 Django 框架的一些详细介绍：

系统整体采用 Django 的框架来进行前后端的开发，Python 下有 Flask、Django、Tornado 等许多款不同可供选择的 Web 框架,其中 Django 框架是主流框架中最有代表性的一位，它最早是被用于开发管理劳伦斯集团下的一些以新闻内容为主网站的内容管理系统软件。Django 创造了许多成功的网站和应用，使用该框架可以使 Web 开发能够以最小的代价构建 Web 应用并且使其维护变的方便。同时 Django 能为频繁进行编程的模块提供了快速的解决方案[3]。

Django 是基于 MVC 的架构进行开发的，MVC 即为 Model-View-Controller（模型-视图-控制器），各部分含义如下： Model（模型）代表一个存取数据的对象及其数据模型； View（视图）代表模型包含的数据的表达方式，一般表达为可视化的界面接口；Controller（控制器）作用于模型和视图上，控制数据流向模型对象，并在数据变化时更新视图。控制器可以使视图与模型分离开解耦合。最简单的 MVC 原理图可表示如下：

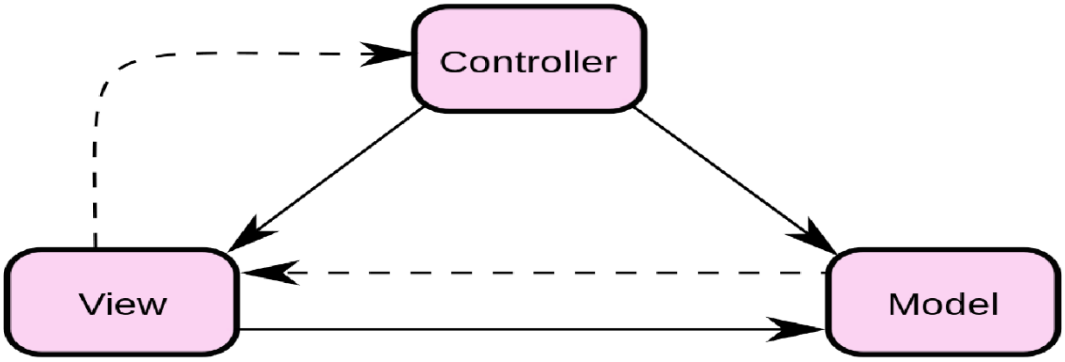

图 3-1 MVC 原理图

其中包含的动作流程有：控制器创建模型； 控制器创建一个或多个视图，并将它们与模型相关联； 控制器负责改变模型的状态； 当模型的状态发生改变时，模型会通知与之相关的视图进行更新。这是目前大型系统开发较为主流的一个过程，最直接的优点就是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，还有重用性高、部署快、可维护性高等优点[4]。在具体的 Django 开发中，对应的 MVC 应用可表示如下：

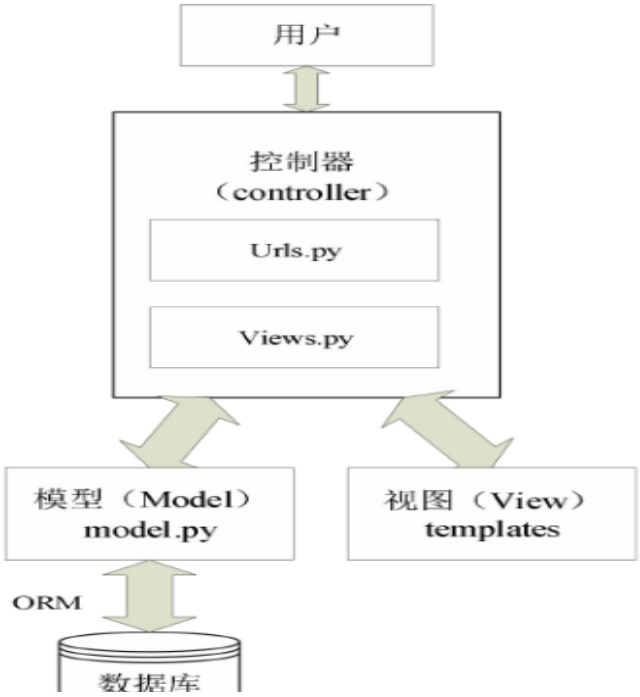

图 3-2 MVC 原理图

这个图中，我们可以很清晰的看出基本符合 MVC 模式的特点，同时 Django 的体量较小，对于轻量级的系统开发和部署，具有很好的效果，不仅可以最大程度的提升开发速度，同时也可以很方便进行测试和维护。

由于采用的是 Django 的框架来进行前后端的开发，所以，本系统采用的是 B/S(Browser/Server)架构，即浏览器/服务器架构，是在 C/S（Client/Service，客户机/服务器）模式的基础上发展起来的一种体系结构，在开发 Web 应用时有明显的技术优势。针对本系统而言，可以使得系统的扩展较为容易且不需要安装专门的软件，使用也较为方便[5]。

4．软件架构的描述

软件架构模型是通过一组关键视图来描述的，同一个软件架构，由于选取的视角（Perspective）和抽象层次不同可以得到不同的视图，这样一组关键视图搭配起来可以完整地描述一个逻辑自洽的软件架构模型。一般来说，我们常用的几种视图有分解视图、依赖视图、执行视图、实现视图和工作任务分配视图。下面我们来进行具体的分析：

4．1 分解视图

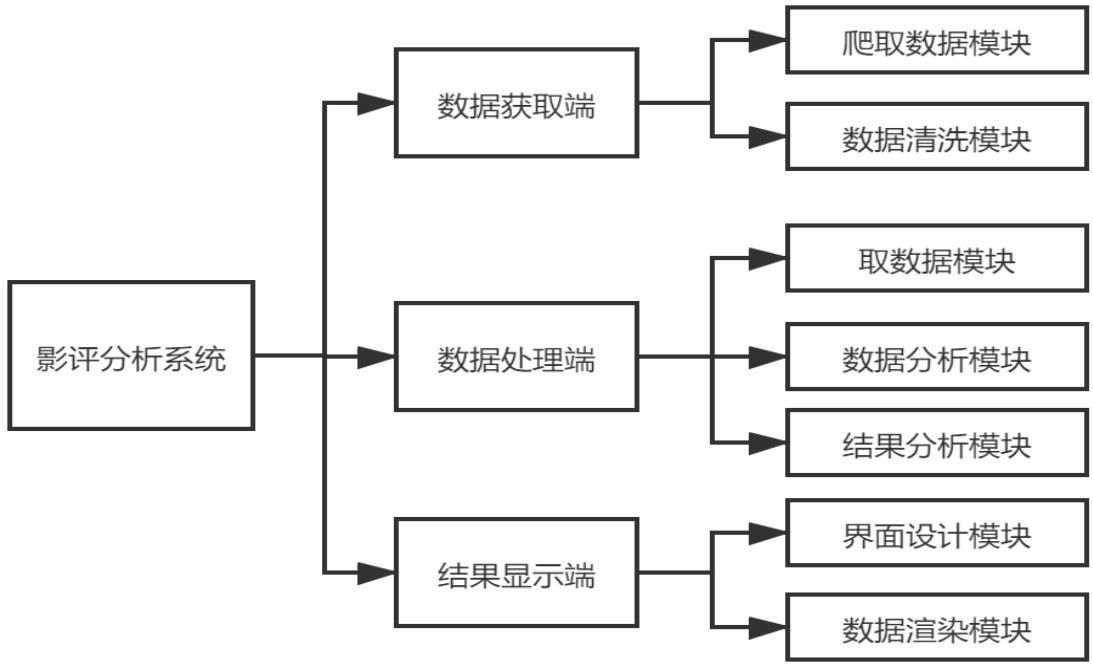

图 4-1 分解视图

分解视图也是描述软件架构模型的关键视图，一般分解视图呈现为较为明晰的分解结构（breakdown structure）特点。简单的说也就是将系统的功能进行分解，形成几个小的部分，然后这些小部分又包含各自所要处理的业务，我们所要做的就是对这些具体业务的设计和开发。本系统中，即分解为数据获取端、数据处理端、结果显示端三个大模块，然后在各个模块内部细分为一些小的功能类，这样使得系统的结构较为清晰，同时也能看清楚各层次之间的联系，使得开发更加系统化。

4．2 依赖视图

依赖视图展现了软件模块之间的依赖关系。比如一个软件模块 A 调用了另一个软件模块 B，那么我们说软件模块 A 直接依赖软件模块 B。如果一个软件模块依赖另一个软件模块产生的数据，那么这两个软件模块也具有一定的依赖关系。本系统中，三大模块之间均存在数据的传递和调用，所以产生如下的依赖视图：

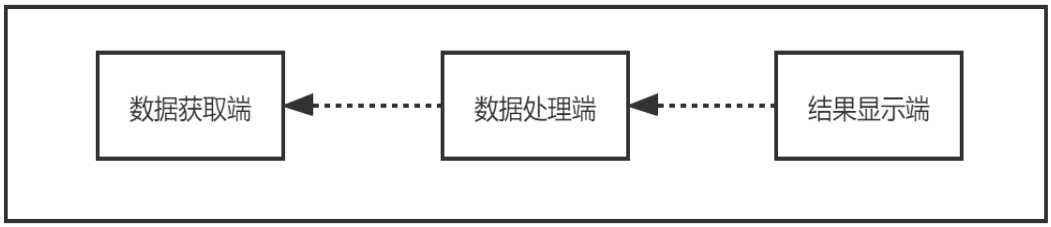

图 4-2 依赖视图

即数据处理端依赖于数据获取端从影评网站上爬取的数据来进行具体的数据分析，并给出最终生成的结果传递给结果显示端；而结果显示端也依赖于数据处理端传递来的数据来进行页面的数据渲染。

4．3. 执行视图

执行视图展示了系统运行时的时序结构特点，比如流程图、时序图等。执行视图中的每一个执行实体，一般称为组件（Component），都是不同于其他组件的执行实体。执行实体可以最终分解到软件的基本元素和软件的基本结构，因而与软件代码具有比较直接的映射关系。在设计与实现过程中，我们一般将执行视图转换为伪代码之后，再进一步转换为实现代码。根据系统的各部分的功能实现，我们画出系统的整体流程图如下：

可见，上图中基本概括了系统的执行时序和整体的动作，作为偏算法类的项目，所以具体的分析处理模块未明确列出，但是对于数据的传递路径以及结果的回调等，

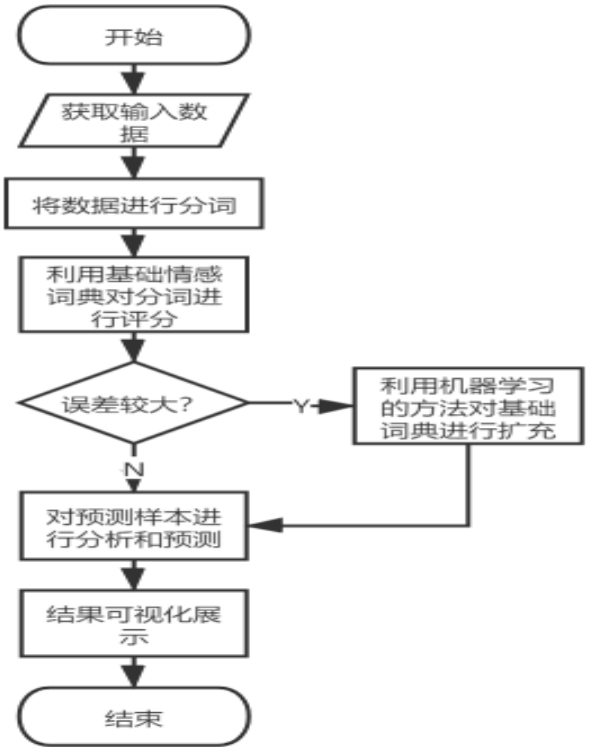

图 4-3 系统的整体流程图

均有了一个简单的执行顺序。

4．4. 实现视图

实现视图是描述软件架构与源文件之间的映射关系。比如软件架构的静态结构以

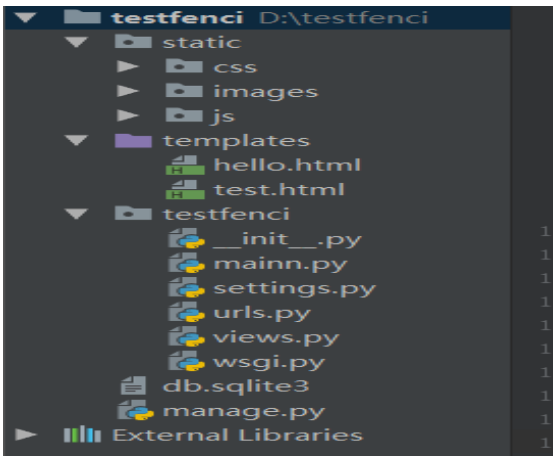

图 4-4 系统源代码的目录文件结构图

包图或设计类图的方式来描述，但是这些包和类都是在哪些目录的哪些源文件中具体实现的呢？一般我们通过目录和源文件的命名来对应软件架构中的包、类等静态结构单元，这样典型的实现视图就可以由软件项目的源文件目录树来呈现。

上图即为系统源代码的目录文件结构，从上往下依次为 css 文件，图片文件和 js

文件，其中分别对应着样式、图片和网页动作等；下面的包即为模板包，内含各个具体的前端页面，即相当于 MVC 中的 V 即视图模块，能将数据进行具体的展示；再下一个文件夹为 Django 框架中的处理相关的文件，内含默认初始化文件、测试文件、默认设置文件、url 路径文件、逻辑实现文件以及部署文件,url.py 和 views.py 两个文件的功能即相当于 MVC 中的 controller 的作用，其中具体的数据分析模型和取数据的模块嵌入在 views.py 当中；再往下即为数据库和全局管理文件。

实现视图有助于我们在海量源代码文件中找到具体的某个软件单元的实现，因为会使得具体的实现代码的层次更加清晰，对源代码的某个模块的定位也会更加精准。实现视图与软件架构的静态结构之间映射关系越是对应的一致性高，越有利于软件的维护，因此实现视图是一种非常关键的架构视图。

4．5. 工作分配视图

工作分配视图将系统分解成可独立完成的工作任务，以便分配给各项目团队和成员。工作分配视图有利于跟踪不同项目团队和成员的工作任务的进度，也有利于在个项目团队和成员之间合理地分配和调整项目资源，甚至在项目计划阶段工作分配视图对于进度规划、项目评估和经费预算都能起到有益的作用。由于本系统的模块区分较为明显，即为数据获取、数据处理、结果展示三大模块，所以组内的三名成员分工也较为明确：

表 4-1 成员分工表

数据获取和清洗	成员 A
数据处理和分析	成员 B
结果可视化和系统维护	成员 C

表 4-2 系统开发的进度安排表

时间	工作	阶段成果
1个月	查找文献资料、完成相关技术研究	概要设计报告
1.5个月	完成评分系统的详细设计	详细设计报告
3.5个月	进行系统的的代码编写，测试	系统代码，完整的系统
1个月	整理文档，撰写工程实践论文，进行最终答辩	工程实践论文

同时，针对本系统开发的进度安排，小组进度计划如下表所示：

商标即为整体规划，整个系统大概 7 个月的时间完成，包括系统的源代码和伴随的说明文档以及开发文档等。

5．设计模式

设计模式的本质是面向对象设计原则的实际运用总结出的经验模型。正确使用设计模式具有以下优点：

可以提高程序员的思维能力、编程能力和设计能力。

使程序设计更加标准化、代码编制更加工程化，使软件开发效率大大提高，从而

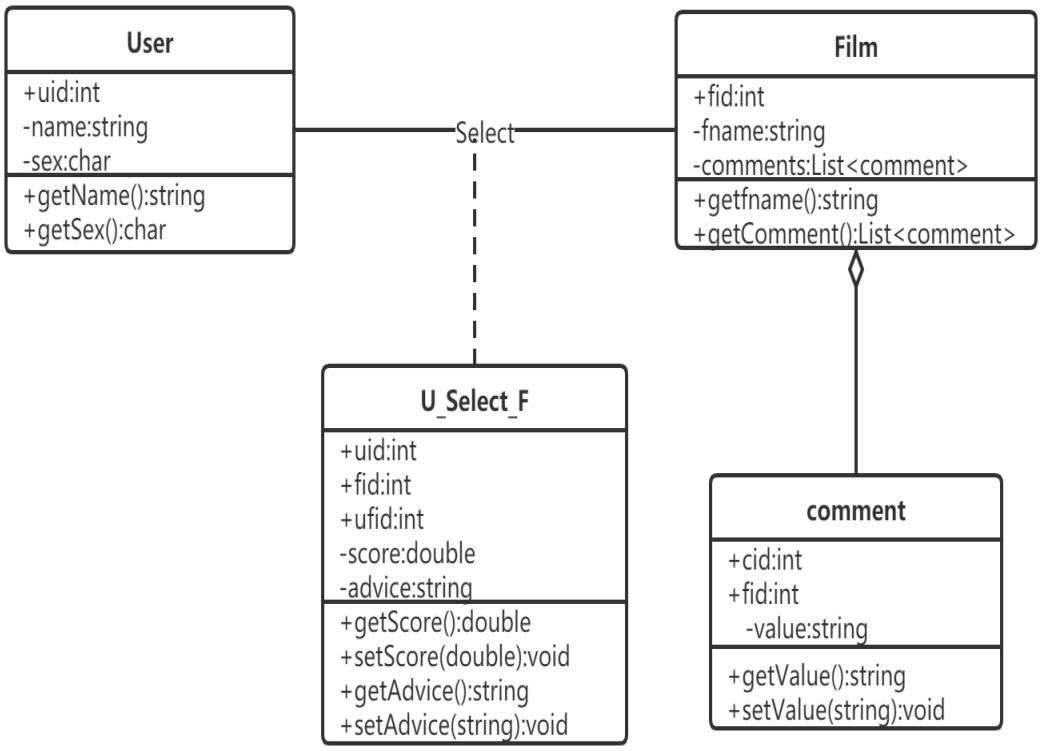

图 5-1 业务类图

缩短软件的开发周期。

使设计的代码可重用性高、可读性强、可靠性高、灵活性好、可维护性强。

在本系统中，我们组要是使用的组合模式，结合系统中的业务类图，下面来进行简要的分析：

针对以上的业务类图，即可以看出用户选择观看的电影，系统即根据选择的电影的评论生成分数和建议，最终反馈给用户，而在实际应用中，系统的主要工作过程也

是对电影的评论的分析，并最终给出系统生成的分数和建议。 从图中，我们可以看出在实现改功能的过程中使用了组合模式，可以使电影的评论动态的添加到电影本身当中来，因为我们的结果是和电影直接绑定的，也就是说单个的评论不能直接影响到最终的结果，这样的好处即简化了分析时的复杂性，根据一部电影即可调出全部的评论，然后进行最终的评分和结果展示，极大的简化了开发的流程。

6．数据库设计

根据第三部分的业务类图，可以看出来系统涉及的主要数据模型即为四个大的存储表，即为用户 User 表，电影 Film 表和 comment 表以及用户选择电影后生成的 U_Select_F 表，具体的表结构和描述如下表格所示：

表 6-1 User 表

Field	Type	Null	Key	Comment
uid	int	—	PRI	用户 id
name	varchar(20)	—	—	用户姓名
sex	varchar(5)	—	—	用户性别

表 6-2 Film 表

Field	Type	Null	Key	Comment
fid	int	—	PRI	电影 id
fname	varchar(20)	—	—	电影名称

表 6-3 comment 表

Field	Type	Null	Key	Comment
cid	int	—	PRI	评论 id
fid	int	—	FRI	电影 id
value	varchar(50)	—	—	评论内容

表 6-4 U_Select_F 表

Field	Type	Null	Key	Comment	Field
ufid	int	—	PRI	事务 id	ufid
uid	int	—	FRI	用户 id	uid
fid	int	—	FRI	电影 id	fid
score	double	—	—	评论得出的分数	score

上述的四个表结构以及表中的数据均存在与关系型数据库中，由于电影这个实体类在代码设计时会直接包含该电影所属的评论，所以在实现的过程中也是直接利用外建的方式将电影的 id 存储到评论之中，在系统实现时，直接查询数据库中的评论即可，较为方便，同时也可以不导致数据冗余（一部电影对应的评论可能很多）。

7. 系统开发环境和技术选型

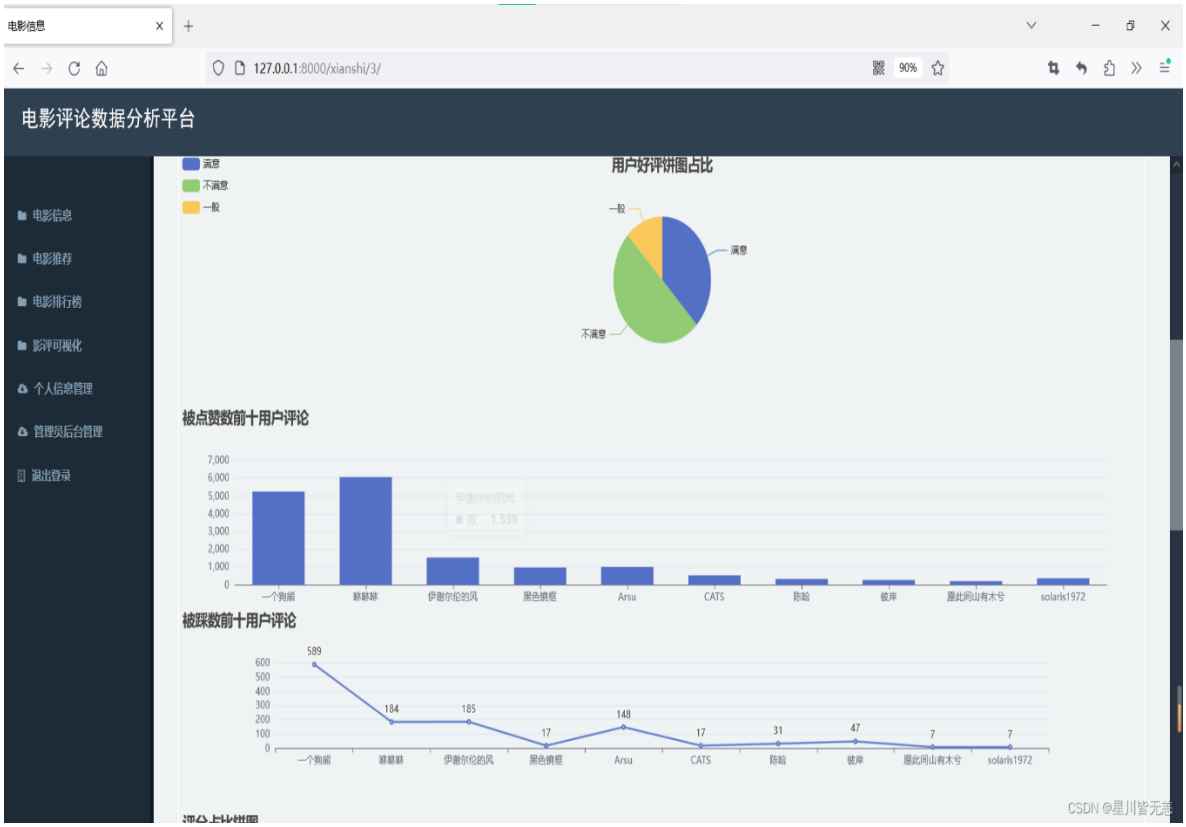

图 7-1 分析结果图

本系统拟采用结构化方法来进行开发，将系统实现的主要功能进行分解，各模块分别进行开发和维护，是的最终的结果利于实现和维护；本系统基于 64 位 Windows10 系统进行开发，开发环境 Python3.6、Django2.0.1 以及 Tensorflow2.0，编译器为 JetBrains Pycharm 2024.3.3，获取数据即为 Python 爬虫技术。

图 7-2 评分前十电影评论词云图

由于系统的数据集最终分为训练样本和预测样本，所以针对训练样本的作用，既作为训练的基础数据集，同时在结果显示出来之后，又做为一个最直接的测试样本，即可直接通过训练样本来进行初步测试，整合好了词典之后，再利用预测样本的结果作为最终的测试结果。

8. 概念原型的核心工作机制

经过上述的分析，可以总结出概念原型的工作机制，简而言之就是用户在系统中选择自己想看的电影，系统根据用户选择的电影从数据库中找到对应的评论，进而对这些评论进行分析，给出最后分析的电影评分和建议，供用户观影之前的参考；我们开发小组也会对该系统进行不断地测试和维护，当系统的评分模型出现较大的误差时，我们便会从数据和算法模型入手，去做数据清洗工作或者算法的改进工作，提高评分的准确率。

9 结论

经过本次的对软件系统的结构特点和架构风格的分析，我对自己的工程实践的开发又有了一些新的思路和认识，尤其是利用各种视图来进行软件系统概念原型的描述时，会让我们更加深入的去分析我们这个系统的目标是什么、我们怎么实现这个目标，尤其是算法类的项目，我们要达到的精度以及完整度才是我们应该追求的，而不仅仅是说为了去分析而分析。经过两次的概念原型的建模和分析，我对软件工程的方法也有了一定的深入了解，现在也会慢慢的思考这些方法的目的和优点所在，知其然也要知其所以然，在以后的需求分析和原型设计中，也要时刻想着利用这些常用的方法去进行分析和设计。

参考文献

[1]蔡文乐,周晴晴,刘玉婷,等基于 Python 爬虫的豆瓣电影影评数据可视化 2021,5(18):5DOI:1019850/j.cnki.2096-4706.2021.18.022.

[2]蔡文乐等."基于 Python 爬虫的豆瓣电影影评数据可视化分析."现代信息科技 5.18(2021):5.

[3]蔡文乐,周晴晴,刘玉婷,&秦立静.(2021).基于 python 爬虫的豆瓣电影影评数据可视化分析.现代信息科技,5(18),5.

[4]高巍,孙盼盼,and 李大舟."基于 Python 爬虫的电影数据可视化分析."沈阳化工大学学报 34.1(2020):6.

[5]孙建立&贾卓生.(2017).基于 Python 网络爬虫的实现及内容分析研究.中国计算机用户协会网络应用分会 2017 年第二十一届网络新技术与应用年会论文集.

...

致　　谢

感谢山西应用科技学院对我几年的培养！

感谢信息工程学院的各位老师各位领导对学子的教导，让学生掌握了基本的专业知识与技能！

感谢高美玲老师对我在学术上的谆谆教诲。让我不仅学到了知识，而且学到了做人的准则和严谨的治学作风。

在此，我表示衷心的感谢和崇高的敬意！

附　　录

```
# 根据用户推荐信息给其他人
def recommend(self,user):
    try:
        # 相似度最高的用户
        top_sim_user = self.top10_simliar(user)[0][0]
        print(top_sim_user)
        # 相似度最高的用户的观影记录
        items = self.data[top_sim_user]
        recommendations = []
        # 筛选出该用户未观看的信息并添加到列表中
        for item in items.keys():
            if item not in self.data[user].keys():
                recommendations.append((item, items[item]))
        recommendations.sort(key=lambda val: val[1], reverse=True)  # 按照评分排序
        # 返回评分最高的 10 部信息
        if len(recommendations) == 1:
            recommendations = []
            lists = []
            for key,value in self.data.items():
                for keys,values in value.items():
                    lists.append((keys,values))
            for i in range(4):
                recommendations.append(random.choice(lists))
            recommendations = list(set(recommendations))
        return recommendations[:10]except:return ''
```