个性化视频推荐系统

# 摘 要

随着信息技术和互联网的发展，人们逐渐从信息匮乏的时代走入信息过载的时代。在这个时代，无论是信息消费者还是信息生产者都遇到了很大的挑战，作为信息消费者，如何从大量信息中找到自己感兴趣的信息是一件非常困难的事情；作为信息生产者，如何让自己生产的信息脱颖而出，而受到广大用户的关注，也是一件非常困难的事情。推荐系统就是解决这一问题的主要工具。

本论文研究的是个性化视频推荐系统，主要是收集用户的爱好，以及web的操作日志，例如用户的观看记录，观看后对视频的评分等信息。然后通过基于用户的协同过滤算法给用户推荐出符合用户的视频。

系统分为以下几个模块，用户模块:用户的相关操作，日志模块:收集用户在界面的操作日志，推荐模块：按照一定的算法给用户推荐视频，管理模块：对视频和用户的管理，例如添加新视频，修改视频相关信息，定时任务模块：计算推荐数据和执行相关的定时任务。各个模块通过共享数据库来衔接。系统前台使用Bootstrap,jQuery,后台使用Python语言，Django web框架，采用Oracle数据库来开发。

关键词： 信息过载 协同过滤 推荐系统 模块 Django

Personalized video recommendation system

# Abstract

With the development of information technology and the Internet, people gradually enter the era of information overload from the age of information shortage. In this era, both the information consumer and the information producer have met a great challenge. As a information consumer, it is a very difficult thing to find the information that they are interested in from a large amount of information. As a producer of information, how to make their own production stand out and receive a lot of users. It is also a very difficult thing to pay attention to. Recommender system is the main tool to solve this problem.

This paper studies the personalized video recommendation system, which mainly collects users' hobbies, as well as the web operation log, such as the user's viewing records, and the ratings of video after watching. Then we recommend users' video to users based on user collaborative filtering algorithm.

The system is divided into the following modules: user module: user related operation, log module: collect user's operation log in interface, recommend module: recommend video to users according to certain algorithm, management module: management of video and user, for example, adding new video, modifying video related information, timing task module: Calculate recommended data and perform related timing tasks. Each module is connected by a shared database. The system front end uses Bootstrap, jQuery, the background uses Python language, Django web frame, adopts Oracle database to develop.

**Key Words:** information overload collaborative filtering module recommendation system Django.

目 录

[摘要 1](#_Toc513711515)

[Abstract 2](#_Toc513711516)

[第一章 绪论 7](#_Toc513711517)

[1.1 研究背景和意义 7](#_Toc513711518)

[1.1.1 研究背景 7](#_Toc513711519)

[1.1.2 研究意义 7](#_Toc513711520)

[1.2 国内外发展状况 8](#_Toc513711521)

[1.2.1 推荐系统国外研究现状 8](#_Toc513711522)

[1.2.2 推荐系统国内研究现状 8](#_Toc513711523)

[1.3 论文内容结构 8](#_Toc513711524)

[1.4 本章小结 9](#_Toc513711525)

[第二章 个性化视频推荐系统相关技术 10](#_Toc513711526)

[2.1 推荐系统开发环境及技术 10](#_Toc513711527)

[2.1.1 PyCharm简介 10](#_Toc513711528)

[2.1.2 Django简介 10](#_Toc513711529)

[2.1.3 JavaScript简介 10](#_Toc513711530)

[2.2 推荐系统架构 11](#_Toc513711531)

[2.2.1 推荐系统的一般架构 11](#_Toc513711532)

[2.3 推荐算法分析 11](#_Toc513711533)

[2.3.1 基于内容 12](#_Toc513711534)

[2.3.2 基于协同 12](#_Toc513711535)

[2.3.3 关联规则 12](#_Toc513711536)

[2.3.4 基于效用 13](#_Toc513711537)

[2.3.5 基于知识 13](#_Toc513711538)

[2.3.6 组合推荐 13](#_Toc513711539)

[2.3.7 推荐算法比较 13](#_Toc513711540)

[第三章 系统需求分析和概要设计 15](#_Toc513711541)

[3.1 系统需求分析 15](#_Toc513711542)

[3.1.1 系统结构需求分析 15](#_Toc513711543)

[3.1.2 系统功能的需求 15](#_Toc513711544)

[3.2 系统的概要设计 15](#_Toc513711545)

[3.2.1 系统的框架设计 15](#_Toc513711546)

[3.2.2 系统模块的划分 16](#_Toc513711547)

[3.2.3 用户模块的设计 17](#_Toc513711548)

[3.2.4 推荐模块的设计 18](#_Toc513711549)

[3.2.5 日志模块的设计 18](#_Toc513711550)

[3.2.6 管理员的模块设计 19](#_Toc513711551)

[3.2.7 定时任务的模块设计 19](#_Toc513711552)

[3.3 数据库的设计 19](#_Toc513711553)

[3.3.1 E-R图 19](#_Toc513711554)

[3.3.2 表的详细设计 20](#_Toc513711555)

[3.4 本章小结 22](#_Toc513711556)

[第四章 视频推荐系统的详细设计与实现 23](#_Toc513711557)

[4.1 系统实现 23](#_Toc513711558)

[4.1.1 Model层的开发 24](#_Toc513711559)

[4.1.2 View层的开发 25](#_Toc513711560)

[4.1.3 Template层的开发 26](#_Toc513711561)

[4.2 功能实现 26](#_Toc513711562)

[4.2.1 用户模块的实现 27](#_Toc513711563)

[4.2.2 推荐模块的实现 31](#_Toc513711564)

[4.2.3 日志模块的实现 33](#_Toc513711565)

[4.2.4 管理模块的实现 33](#_Toc513711566)

[4.2.5 定时任务模块的实现 35](#_Toc513711567)

[4.3 本章小结 38](#_Toc513711568)

[第五章 个性化视频推荐系统测试 39](#_Toc513711569)

[5.1 单元测试 39](#_Toc513711570)

[5.1.1 用户界面测试 39](#_Toc513711571)

[5.1.2 系统基本功能测试 40](#_Toc513711572)

[5.1.3 系统推荐功能测试 40](#_Toc513711573)

[5.1.4 管理功能测试 41](#_Toc513711574)

[5.2 兼容性测试 41](#_Toc513711575)

[5.3 本章小结 41](#_Toc513711576)

[总结与展望 42](#_Toc513711577)

[参考文献 43](#_Toc513711578)

[致谢 44](#_Toc513711579)

# 第一章 绪论

## 1.1 研究背景和意义

### 1.1.1 研究背景

随着互联网的普及，互联网上信息量呈现爆炸式增长，信息重复和信息过多导致的被动获取的信息过载，信息过载成为了人们快速有效获取信息一个主要障碍。解决信息过载的主要手段有搜索引擎和推荐系统。搜索引擎方便我们快速找到我们想要了解信息，加快对于知识了解以及获取速度，对我们每个人都有毋庸置疑的价值、意义。但是，搜索是根据和群体行为相关的权重排序信息，带有任务的用户即希望能快速搜索到感兴趣的内容之后再深入阅读，再带着新的任务继续搜索，而实现的情形是个体用户不得不调整关键词反复搜索处于长尾的信息。推荐系统是解决这些问题最有潜力的方法。其作用在于：1）帮助用户快速发现感兴趣和高质量的信息，提升用户体验。2）增加用户使用产品的时间。3）减少用户浏览到重复或者厌恶的信息带来的不利影响。4）提供个性化信息，信息的推荐更为精准。

传统上，我们在进行决策时会寻求以下方式：找朋友聊聊、从可信的第三方获取信息、雇佣专家团队、在互联网上咨询、使用决策论的方法或者直接凭借直觉，所以不同的用户接收到的建议都是不同的。因此构建支持用户在线决策的系统正是当下推荐系统领域的主要目标，然而这个目标要为大规模用户提供便捷访问的高质量推荐。对未来生活的设想中，生活中的方方面面都离不开推荐系统的支持，设想一下，一个愉快的周末没有什么比愉悦的音乐和阅读更重要，打开音乐网站是陌生但是很对味和心情的旋律，打开新闻网站是已经准备好新鲜又正中下怀的新闻资讯，最后打开电商网站，推荐的是实惠又及时的商品。

### 1.1.2 研究意义

对于推荐系统我认为如果能将用户关注领域前沿信息、领域过去发展脉络、领域牛人、领域最经典文章推荐给关注该领域用户，个人认为是一件对于信息传播及其有价值，也是令人兴奋的一件事，技术上也有极大挑战。

如果一个个性化推荐系统能将新的、有意思领域推荐给我们，并且将我们关注领域最新进展、历史脉络、有价值信息等等推荐给我们，我认为这件事对于消除信息壁垒，增加信息价值是很好手段、方法。值得我们为此付出努力和心血。并且认为这是一件对人、说大点对人类有价值、有意义的事情。

## 1.2 国内外发展状况

### 1.2.1 推荐系统国外研究现状

“个性化推荐系统”一词是在1995年3月，斯坦福大学的MarkoBalabanovic等人在美国人工智能协会上所提出的。

### 1.2.2 推荐系统国内研究现状

国内的互联网虽然没有国外的起步早，但是近几年发展非常迅速。推荐系统方面也不例外。2009年7月，国内首个推荐系统科研团队北京百分点信息科技有限公司成立，该团队专注于推荐引擎技术与解决方案，在其推荐引擎技术与数据平台上汇集了国内外百余家知名电子商务网站与资讯类网站，并通过这些B2C网站每天为数以千万计的消费者提供实时智能的商品推荐。2011年9月，百度世界大会2011上，李彦宏将推荐引擎与云计算、搜索引擎并列为未来互联网重要战略规划以及发展方向。百度新首页将逐步实现个性化，智能地推荐出用户喜欢的网站和经常使用的APP。渐渐的国内在推荐系统应用方面也逐步扩大，在电商中有淘宝京东推荐商品，音乐方面有网易云，阅读方面的有知乎，视频相关的有优酷，土豆等还有最近很热的快手，抖音等等的小视频都有大量的应用。

## 1.3 论文内容结构

本论文分为五章来讲解，基本内容如下：

第一章，对此论文研究的背景、意义，以及国内外在推荐系统方面的研究状况进行概述。

第二章，讲解推荐系统的相关技术，推荐系统的一般架构，推荐算法等。

第三章，从软件工程的角度考虑问题，讲解系统的需求分析，概要设计，数据库的设计。

第四章，从开发实现的角度讲解，给出具体的实现流程，落实于代码。

第五章，为了系统能够稳定安全的运行，进行系统测试。

## 1.4 本章小结

本章主要讲解个性化视频推荐系统的研究意义和目的，以及推荐系统在国内外的发展状况，第三小节对论文的结构内容进行了概括，以便读者能够提前了解论文的讲解思路。

# 第二章 个性化视频推荐系统相关技术

## 2.1 推荐系统开发环境及技术

个性化视频推荐系统用面向对象的解释性Python语言开发，在强大的PyCharm IDE软件的支持下，利用开源的Django Web框架所开发，这样能够降低设计的成本，节省开发时间。

### 2.1.1 PyCharm简介

PyCharm是由JetBrains开发的一款专业的Python IDE，可以帮助用户在使用Python语言开发时提高效率的工具，有调试，语法高亮，project管理，智能提示，单元测试和版本控制等功能，该IDE还支持一些高级功能，例如支持Django框架下的专业web开发。

### 2.1.2 Django简介

Django是一个开放源代码的Web应用框架，由Python写。采用了MTV的框架模式，即模型M，模板T和视图V。如图2.1是Django的工作模型。



图2.1 Django工作模型

### 2.1.3 JavaScript简介

JavaScript是由网景公司与sun公司与1995年共同开发设计的一种直译式脚本语言，一种动态类型、弱类型、基于原型的语言。为了看上去能够像Java，因此取名为JavaScript。它包括ECMAScript：描述了该语言的语法和基本对象、文本对象模型（DOM）：描述处理网页内容的方法和接口和浏览器对象模型（BOM）：描述与浏览器进行交互的方法和接口三部分。

## 2.2 推荐系统架构

### 2.2.1 推荐系统的一般架构

推荐系统一般分为三个模块，推荐模块，UI模块，日志模块。推荐模块主要为推荐引擎，推荐引擎的构建可以来源于多个数据源（用户特征多样化）+不同的推荐算法，推荐引擎的架构可以是多样化的（实时推荐+离线推荐），最后融合推荐结果，融合方式可以是线性加权或者切换式的。UI模块一般包含三块：1）通过一定的方式推荐物品（物品标题、缩略图、简介）；2）给出推荐理由；3）数据反馈改进个性化推荐模型。日志模块主要采集用户反馈的结果，为推荐模块提供有效的数据，方便训练符合用户特点的模型给用户推荐恰当的物品。如图2.2是推荐系统一般架构，包含各个模块之间的联系。



图2.2 推荐系统一般架构

## 2.3 推荐算法分析

有推荐功能的系统相比普通系统当然是多了推荐功能，推荐系统的核心是推荐算法，常见的推荐算法有基于内容、基于协同、关联规则、基于效用、基于知识和组合推荐等等。推荐算法分类见图2.3。接下来对常见的推荐算法做一简单描述。



图2.3 推荐算法分类

### 2.3.1 基于内容

基于内容的基本思想是，根据用户过去喜欢的物品，为用户推荐和他过去喜欢的物品相似的物品。例如：在京东当中，每当进入任何一个物品页面的时候都会有一个“为你推荐”栏目，这时候他就会根据你经常购买的物品给你推荐相似的物品。对于我经常购买互联网类的书籍，所以他会给我推荐类似的书籍。

### 2.3.2 基于协同

协同过滤是比较著名的推荐算法。算法通过对用户历史行为数据的挖掘发现用户的偏好，基于不同的偏好对用户进行群组划分并推荐品味相似的商品。协同过滤推荐算法分为两类，分别是基于用户的协同过滤算法，和基于物品的协同过滤算法。简单的说就是：人以类聚，物以群分。

### 2.3.3 关联规则

关联规则是反映一个事物与其他事物之间的相互依存性和关联性，常用于实体商店或在线电商，通过对顾客的购买记录数据库进行关联规则挖掘，最终目的是发现顾客群体的购买习惯的内在共性。最经典的应用案例莫过于“啤酒和尿布”。

### 2.3.4 基于效用

基于效用的推荐是建立在对用户使用项目的效用情况上计算的，其核心问题是怎么样为每一个用户去创建一个效用函数，因此，用户资料模型很大程度上是由系统所采用的效用函数决定的。

### 2.3.5 基于知识

基于知识的推荐在某种程度是可以看成是一种推理技术，它不是建立在用户需要和偏好基础上推荐的。基于知识的方法因它们所用的功能知识不同而有明显区别。

### 2.3.6 组合推荐

金无足赤，人无完人。当然，算法也不列外，以上各个算法都有优缺点（见2.3.7 推荐算法比较），在实际运用当中，可以把某几个算法相结合，通过组合避免或者弥补推荐算法的弱点。例如用基于内容的方法和协同过滤推荐方法各个产生一个推荐预测结果，然后根据某方法组合其结果。

### 2.3.7 推荐算法比较

表2.1 推荐算法比较

|  |  |  |
| --- | --- | --- |
| 推荐方法 | 优点 | 缺点 |
| 基于内容推荐 | 推荐结果直观，容易解释；  不需要领域知识 | 新用户问题；  复杂属性不好处理；  要有足够数据构造分类器 |
| 协同过滤推荐 | 新异兴趣发现、不需要领域知识；  随着时间推移性能提高；  推荐个性化、自动化程度高；  能处理复杂的非结构化对象 | 稀疏问题；  可扩展性问题；  新用户问题；  质量取决于历史数据集；  系统开始时推荐质量差； |
| 基于规则推荐 | 能发现新兴趣点；  不要领域知识 | 规则抽取难、耗时；  产品名同义性问题；  个性化程度低； |
| 基于效用推荐 | 无冷开始和稀疏问题；  对用户偏好变化敏感；  能考虑非产品特性 | 用户必须输入效用函数；  推荐是静态的，灵活性差；  属性重叠问题； |
| 基于知识推荐 | 能把用户需求映射到产品上；  能考虑非产品属性 | 知识难获得；  推荐是静态的 |

## 2.4 本章小结

本章介绍了此系统开发过程用到的一些主要技术和工具，接着对推荐系统架构从更高一层级，更抽象的角度做了简单分析介绍，最后对常用的推荐算法做了分析介绍，各种算法都用什么优缺点，以便用符合自己系统的推荐算法。

# 第三章 系统需求分析和概要设计

## 3.1 系统需求分析

系统需求分析是确定“为了解决这个问题，目标系统必须做什么”，确定目标系统必须有哪些功能，需求分析阶段确定的系统逻辑模型，是以后设计和实现目标系统的基础，也是系统开发人员、开发过程和开发成本等一系列前期需求制定的详细计划。如果没有详细的需求分析，可能在以后开发过程中会出现更改数据库，更改架构甚至重新开始，这样会浪费人力财力。

### 3.1.1 系统结构需求分析

本系统采用当下流行的BS结构（Browser/Server,浏览器/服务器结构）设计，对于开发和维护来说，只要开发完成，发布在服务器上，维护也在服务端维护就可以，客户端可以说是零维护。对于用户，只需要一个浏览器，就可以访问，不需要像CS结构（Client/Server,客户端/服务器结构）一样需要安装客户端，这样方便用户使用。

### 3.1.2 系统功能的需求

通过对当下的视频网站和有推荐功能的网站进行分析，此类系统主要分类两个部分，前台和后台，前台供用户使用，后台供管理使用。前台是用户交互的界面，用户可以注册成为新用户，告诉系统自己的兴趣爱好，系统会根据你的爱好推荐相应的视频，这样可以解决冷启动的问题。用户观看视频后进行评分，系统手机用户的评分以及用户浏览的历史，用这些数据不断的训练模型，推荐恰当的视频给用户。后台管理员是对整个系统的维护，不断的采集新的视频数据，然后添加到系统，供用户观看，然后可以对一些视频信息做更改以及系统的用户的维护。

## 3.2 系统的概要设计

### 3.2.1 系统的框架设计

生活中，盖一栋大楼，其中的人员都是分工明确，各司其职，我们的软件也是一样。把一个大的系统逐步分成为小的模块，最后分给不同的人员来完成。我们可以按照不同的维度来划分。纵向可以把整个系统的设计为三层，template层、view层和model层，横向划分就是我们3.2.2节的模块划分。三层主要功能如下：

Model层：主要是用来负责业务对象和数据库对象。

Template层：负责把页面如何展示给用户。

View层：负责业务逻辑，并在适当的时候调用Model和Template。

三层之间的调用如图3.1。



图3.1 三层调用

### 3.2.2 系统模块的划分

为了更好的管理，开发，维护，本系统划分为以下几个模块，用户模块，推荐模块，日志模块，后台管理模块和定时任务模块。各个模块的功能如下：

用户模块：用户的登录、注册、查找等功能。

推荐功能：根据用户的请求，查询离线计算的推荐数据，推荐符合用户的视频。

日志模块：收集用户的浏览记录，对视频的评分。

后台管理：实现对视频的管理，包括新视频的发布，修改视频的信息，还有用户管理。

定时任务：此系统的推荐设计为离线推荐，不论机器的环境有多好，我们的数据都在不断的增加，如果实时计算，初始阶段可能不存在性能问题，久而久之后，当用户数量增加，视频数量增加，就会存在性能瓶颈，因此，开始设计阶段就设计为离线计算。把推荐数据提前计算好，定时更新变化的数据，提高系统的性能，给用户更好的体验。图3.2是系统的五大模块。



图3.2 系统模块

### 3.2.3 用户模块的设计

前台界面为用户提供的，因此多应该站在用户的角度考虑问题。为了提升用户的体验效果，我们的系统不只是提供推荐功能，还有对视频做分类，用户可以根据自己的爱好，挑选自己喜欢的视频类型进行观看。如果用户有明确想看的视频，通过搜索功能可以更精确的找到视频，最后对观看过的视频可以做出评价，系统会根据你的评价给你推荐最符合你的视频。当然为了让系统知道你是谁，你需要登录到系统。如果你是新用户，不要担心，我们还提供了注册功能，注册的时候还可以告诉系统你喜欢的视频类型。这样就更能推荐出合适的视频。用户功能如图3.3。



图3.3 用户功能

### 3.2.4 推荐模块的设计

推荐模块当然是系统的最主要的部分，本系统的设计为离线计算不是实时计算（原因见3.2.7节定时任务模块），因此在推荐模块不实现推荐算法，推荐算法由定时任务调用的存储过程来实现，此模块接受用户的推荐请求，对推荐算法计算的结果查询过滤，最终推荐给用户。我们主要采用的是基于用户的协同过滤算法做推荐，为了解决冷启动问题，我们结合多种推荐方式，所以在推荐模块就得做控制功能，何时该才用什么算法，或者多种算法相结合实现推荐。

### 3.2.5 日志模块的设计

推荐用户最喜欢的视频是系统的关键。那么怎么知道用户喜欢什么？这就得根据用户的浏览记录来判断用户喜欢什么。因此收集用户浏览记录就不可缺少。例如收集看过的视频，观看时间长度，类型，用户观看完后对视频的评分等信息，这就是日志模块该做的事情。

### 3.2.6 管理员的模块设计

如果视频的资源不更新，系统做的再好也不会有大量的用户。大量的用户中当然有非法的用户，因此对视频和用户的管理就是理所当然的事情。管理员模块主要分为视频管理和用户管理。在视频管理中，主要做新视频的增加，修改视频的信息，顶置视频等功能。用户管理主要对不合法的用户做禁用等功能。

### 3.2.7 定时任务的模块设计

系统的设计尽可能要有扩展性，一个系统刚开始用户数量可能不多，视频的数量也可能不多，随着时间的推移，这两者都会慢慢的增加。那么如果我们的推荐算法是实时计算的话，可能时间会变长，这样推荐给用户的时间增加，用户在等待推荐结果，体检效果很不佳，因此，在设计的时候就应该考虑这一点。为了避免以上问题的发生。我们采用离线计算推荐结果。系统初始化的时候计算好已有的数据，当有数据变化的时候，更新相应的数据。我们采用定时任务来完成。定时的检验数据是否有变化，有变化则更新变化的数据。这样可以提升系统的性能。而我们的数据都是存储在数据库，大量的计算如果采用python语言，我们会在数据的查询和插入花费大量的时间，因此采用数据库的存储过程来计算，减少连接数据库的开销时间。

## 3.3 数据库的设计

上一节当中对系统更详细的设计，每个模块都有什么功能做了详细的阐述。这些功能的数据都要存储起来，要么存储在文件，要么存储在数据库。数据库的速度快，读写方式也比较简便，因此我们采用数据库。

### 3.3.1 表的详细设计

更好的能体现推荐功能，需要用户登录到系统。为了存储用户的信息，我们设计表3.1用户表。有id来标识唯一用户，以及注册的时候收集用户的爱好信息。密码和用户名为了登录到系统当中，状态来标识你是否为一个合法的用户。各个字段的数据类型以及名称等信息见表3.1。

表3.1 用户表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 允许空值 | 备注 |
| ID | NUMBER | - | 否 | 用户id |
| user\_name | VARCHAR2 | 255 | 是 | 用户名称 |
| Password | VARCHAR2 | 255 | 是 | 密码 |
| Email | VARCHAR2 | 255 | 是 | 电子邮件 |
| Hobby | VARCHAR2 | 255 | 是 | 爱好 |
| State | VARCHAR2 | 1 | 是 | 状态 |

视频是我们推荐的数据，每个视频都有一个id唯一标识，用户需要了解了视频的信息后才选择是否观看，因此视频应该有题目，年份，属于那种类型，以及视频的简介信息，为了观看还应该有播放的地址，为了顶置一个视频，我们用一个字段标识该视频是否为新视频。表3.2是视频表的具体设计。

表3.2 视频表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 允许空值 | 备注 |
| MOVIE\_ID | NUMBER | - | 否 | 电影id |
| TITLE | VARCHAR2 | 255 | 否 | 标题 |
| YEAR | VARCHAR2 | 4 | 是 | 年份 |
| GENRES | VARCHAR2 | 255 | 是 | 分类，以|分割 |
| DESC | CLOB | - | 是 | 描述 |
| LINK | VARCHAR2 | 255 | 是 | 链接 |
| ISNEW | NUMBER | - | 是 | 是否为新视频 |

训练合适的模型，给用户推荐数据，我们需要收集用户观看后对视频的评分信息。评分表见表3.3。每一个用户都可以对每个视频进行评分，因此应该有用户信息，视频信息和评分，当用户给一个视频评分以后，用户的推荐数据就有可能变化，因此我们设计一个字段，来标识是否需要更新，定时任务来检查该表，来更新推荐数据。

表3.3 评分表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 允许空值 | 备注 |
| USER\_ID | NUMBER | - | 否 | 用户id |
| MOVIE\_ID | NUMBER | - | 否 | 视频id |
| RATING | NUMBER | - | 是 | 评分 |
| ISUPDATE | VARCHAR2 | 1 | 是 | 是否更新 |
| TIMESTAMP | VARCHAR2 | 100 | 是 | 时间戳 |

推荐算法的实现分为两步，第一步计算每个用户的相似用户的相似值，第二步计算推荐每个视频的推荐值。因此相似用户表设计为两个用户和两个用户的相似值，见表3.4。

表3.4 相似用户表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 允许空值 | 备注 |
| USER\_ID1 | NUMBER | - | 否 | 用户1 |
| USER\_ID2 | NUMBER | - | 否 | 用户2 |
| VALUE | NUMBER | - | 否 | 相似值 |

根据相似用户计算推荐值，因此有用户信息，视频的信息，以及推荐值，用户我们用用户id表示，视频用视频id表示。具体的见表3.5。

表3.5 推荐列表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名 | 数据类型 | 长度 | 允许空值 | 备注 |
| USER\_ID | NUMBER | - | 否 | 用户id |
| MOVIE\_ID | NUMBER | - | 否 | 视频id |
| RECOMMEND\_VALUE | NUMBER | - | 否 | 推荐值 |

### 3.3.2 E-R图

E-R图是现实世界的概念模型，能更清楚的表示每个实体的属性以及各个表之间的关系。图3.4描述了本系统中主要的三张表（评分表，电影表和用户表）的E-R模型。



图3.4 E-R模型

## 3.4 本章小结

本章从软件系统的角度来讲解，主要包括系统的需求分析，概要设计，和数据库的设计。做好设计是开发软件的前提。

# 第四章 视频推荐系统的详细设计与实现

## 4.1 系统实现

在上一章中，对系统做了需求分析和设计，清楚了系统该有那些功能，大体的实现有了方向。本章中，从实现方面来做详细叙述。本系统采用Python语言开发，Django Web框架，前台页面主要用HTML ，CSS，JavaScript，BootStrap等前端框架。数据库采用Oracle数据库。本章重点讲在个性化视频推荐系统开发过程的技术问题，编码问题和主要的流程。图4.1是各个功能之间通过共享数据库之间的联系:



图4.1 各个功能之间的联系

### 4.1.1 Model层的开发

在本层当中主要对数据库中表对象的增加、删除、修改和查询等操作，本层在整个系统中的作用如下：



图4.2 Model层的作用

在数据访问这一层主要封装了对数据库的访问细节，在原始的开发当中，我们如果操作数据库，我们一般都会进行如下的操作，1）创建数据库，设计表结构和字段；2）使用MySQLdb（假设用MySQL数据库）来连接数据库，并编写数据访问层代码（建立连接，编写sql语句，执行sql，关闭连接）；3）业务逻辑层去调用数据访问层执行数据库操作。而在本系统当中，我们使用Django的ORM（Object Relational Mapping，关系对象映射）框架。类和表之间的对应关系如图4.3；在访问数据库时，只需要用对应的类，调用恰当的API就可以完成。减少代码的编写提高开发效率以及后期的维护。这也是我们选择Django框架的原因。



图4.3 类到表的映射

那么，我们的表和类怎么关联起来呢？分为两种情况：第一：根据类创建表，第二根据表创建类。Django框架已经帮我们封装好了，只要执行相应的语句就可以让表跟类关联起来，然后操作相应的API。

### 4.1.2 View层的开发

业务逻辑层是专门处理软件业务需求的一层，处于数据库之上，服务层之下，完成一些列对Domain Object的CRUD，作为一组微服务提供给服务层来组织在暴露给表现层。

业务逻辑层包含领域对象模型，领域实体，业务规则，验证规则，业务流程。领域对象模型为系统结构描述，包含实体功能描述，实体之间的关系。领域模型处于天生的复杂性。领域实体：业务层是一些操作业务对象（BO）的处理。业务对象包含数据和行为，是一个完整的业务对象。业务最大的逻辑就在处理一些列现实世界的规则，这也是软件中最容易变化的部分，这里通常会出现我们众多的if-else或者switch-case的地方。这也是我个人觉得在我们的项目最应该关系和分离需求的层次。验证规则：业务规则很大程度上也是对对象的数据验证，验证业务对象的当前数据状态。我觉的在每个业务对象上都应该存在一个对外部对象暴露的验证接口。

如图4.8是业务逻辑层在软件结构中的位置：

****

图4.4 业务逻辑层的位置

### 4.1.3 Template层的开发

对页面设计进行的任何操作都必须对后台代码进行相应的修改。因此如果可以在不改变后台代码修改的情况下变更设计，那将会方便很多。所以出现了模板的开发。开发模板层时主要包括html+模板语法，模板包括在使用时会被替换掉的变量，和控制模板逻辑的标签。

例如：在用户模块，我们使用if else逻辑标签判断用户是否登录，如登录使用session中的用户替换模板，若没有，显示登录按钮让用户去登录。

## 4.2 功能实现

本小节讲解各个功能的具体的实现。在开发过程中，前台的请求以Ajax的方式请求，后台以Json的格式返回数据。Json的格式信息规范如表4.1：

表4.1 返回数据类型规范

### 4.2.1 用户模块的实现

#### 1．用户注册



图4.5 用户注册界面

为了能给用户带来更好的体验，推荐符合用户的视频，需要让系统知道你是谁，如果是老用户，已经有账号，直接登录如果没有账号，就是新用户，新用户需要先注册才能登录，注册页面如4.10。图4.11是新用户注册功能的处理流程。



图4.6 用户注册流程图

在系统的首页有登录->注册的按钮，点击进去之后填写相应的信息。填写完成提交之后前后台都会做相应的校验工作，然后判断用户名是否已注册，未注册则给用户生成ID（此ID使用Oracle的Sequence），把对应的其他信息插入到数据库，则注册成功，用户可以去登录页面登录。

#### 2．用户登录



图4.7 用户登录界面

用户登录的具体流程为：当用户提交登录信息后，前台用JavaScript做校验，用户名和密码都不能为空，密码不能低于六位。在前台校验成功后，然后封装参数以ajax的方式请求到后台，这样可以减少服务器的压力。后台获取登录的信息后，也必须校验。在某些情况下，用户可以不通过前台页面直接请求到后台，因此后台的校验也是必须的。校验通过后再从数据库查找，判断该用户是否存在以及用户名密码是否正确。若存在并且正确，则把用户信息放入session当中，供前台页面显示。否则返回错误提示。流程如图4.13。



图4.8 用户登录流程图

用户登录成功后的主页面如图4.14。

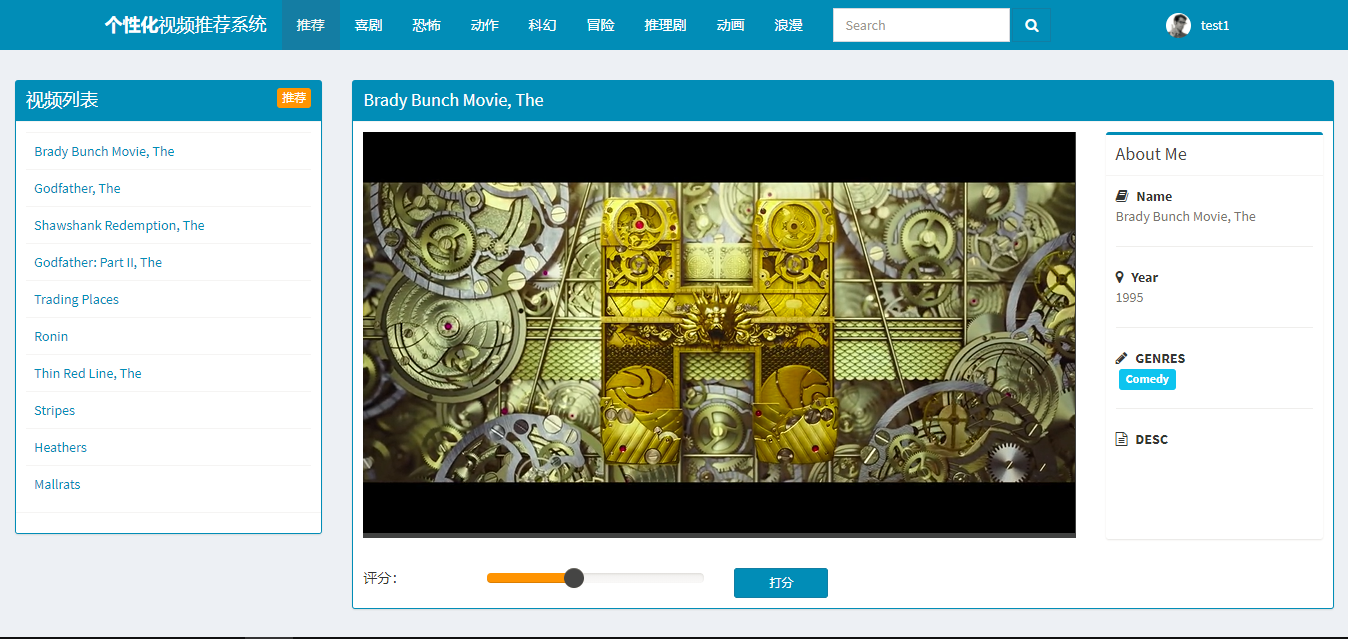


图4.9 用户主界面

用户有登录操作相对用的就有退出操作，用户退出主要是销毁session中用户的信息，界面显示把用户名称更换为登录的按钮。前台点击退出按钮后发送ajax请求到后台，后台返回成功后前台重新加载页面，此时session中已没有用户的信息，所以显示登录的按钮。

#### 3． 用户其他操作

界面菜单如图4.16。用户在界面可以进行查找，在搜索框输入关键词后，点击查找按钮之后，系统会根据输入的关键词按视频的名称进行模糊匹配。然后将查到的视频返回给用户。



图4.10 界面菜单

界面中还对视频进行分类，在导航栏中有推荐和视频的分类。用户点击某一个分类后台根据类型会为用户查找出相应的视频，为了解决新视频的冷启动问题，如果查出来的视频中含新视频，我们会把新视频顶置，以便用户可以看到新视频。此处都是用户查找操作，因此我们把按分类和用户的搜索抽象成一段公用的逻辑，符合软件设计的规范。

### 4.2.2 推荐模块的实现

推荐模块是我们系统的核心模块。根据我们第三章的设计，推荐算法的实现交给我们的定时任务模块调用的存储过程来实现。因此推荐模块相对简单。先看我们的推荐模块处理流程，如图4.18：



图4.11 推荐流程

1 首先获取session中的用户信息；

2 判断是否有用户登录；

3 有用户登录按照推荐算法推荐，当为新用户第一次登录的时候，推荐算法也不能推荐，此时推荐出来的数据为0条。这就是用户的冷启动问题，为了解决此问题我们判断是新用户的话，按照用户注册时的爱好信息，根据电影的类型给用户推荐，当用户观看一段时间后，再按照推荐算法推荐。没有用户登录，我们给用户推荐最新的视频，和其他用户对某个视频评分较高的来推荐。

### 4.2.3 日志模块的实现

日志模块主要记录用户看过视频的评分信息，此系统的推荐主要也依赖评分信息。如果用户是第一次给某电影评分，则插入一条记录，此种情况会发生在分类的视频或者查找的视频或者推荐的视频，推荐的视频都是用户未评分的。如果插入的记录已经存在，我们会更新原来的评分。此种情况发生在分类视频和查找的视频当中，因为其中可能有用户看过的视频。因此推荐的类型就掌握在用户的手中，你可以修改评分从而导致推荐结果。

### 4.2.4 管理模块的实现

管理模块主要对视频和用户管理，管理的界面如图4.20：

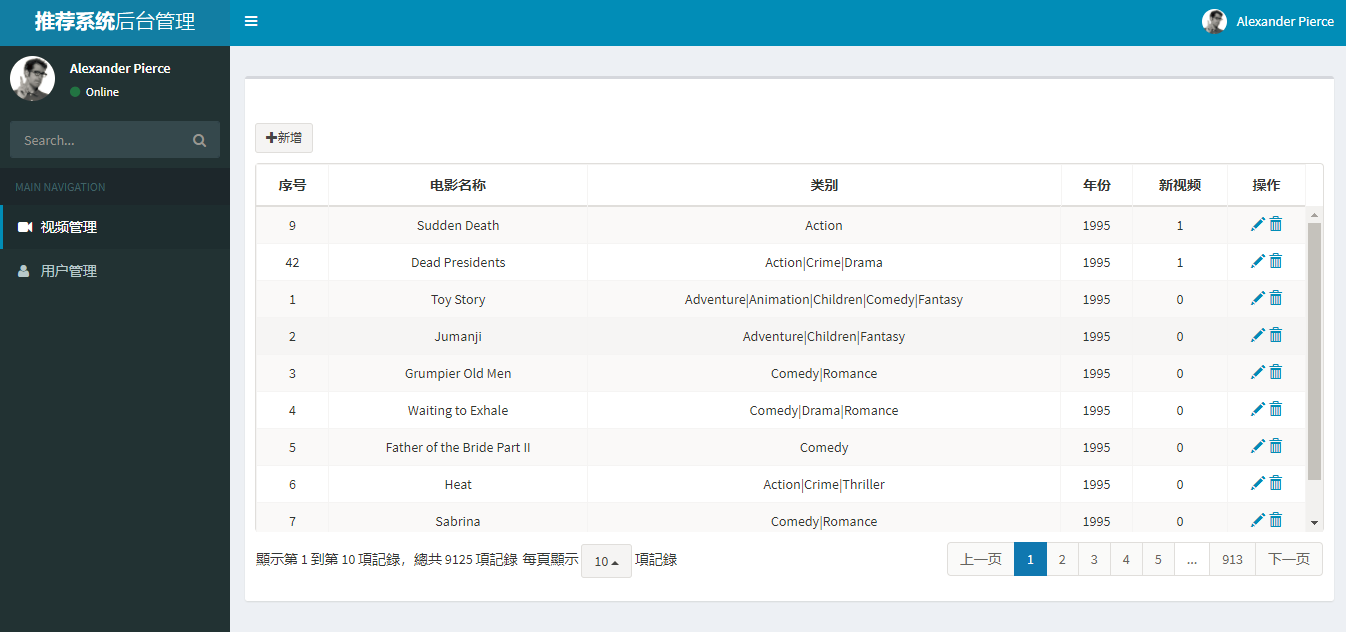


图4.12 视频管理后台界面

在左边可以切换视频管理和用户管理。我们先看视频管理，在右边会列出所有的视频，我们会把新视频顶置，一行代表一个视频，在每行的最右边有一个操作的栏目，可以对此视频进行删除操作和修改操作。在最上面有一个添加的按钮，可以添加新的视频。

视频删除：点击删除的时候回给出确认框如图4.21:



图4.13 删除操作确认框

确实是否删除此视频，如果确认则向后台传视频的ID进行删除，取消则不删除。

视频修改：点击修改的图标回显此视频的信息如图4.23：

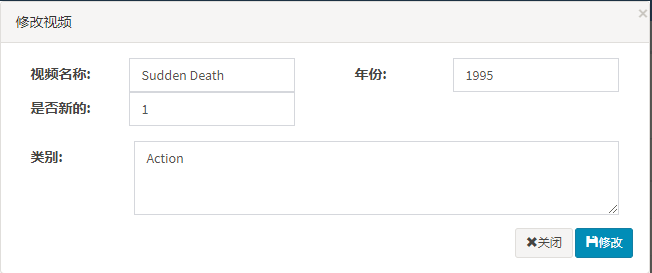


图4.14 修改视频界面

此时可以选择关闭或者更改信息后点击修改按钮。前台把页面的数据传给后台进行修改。后台获取参数，进行校验，校验合法然后修改：

视频增加：点击添加按钮之后会有一个弹出框如图4.25来填写视频的信息：

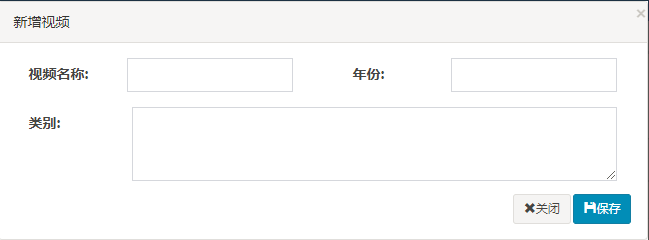


图4.15 增加视频界面

填写完成后点击保存即可增加新视频：后台也是获取参数与校验操作，与更新不同的是，这里先查找到视频库最大的视频的ID，然后给此ID+1作为新视频的ID；

图4.26是用户管理的界面。用户管理比较简单，主要对非法的用户进行禁用，在用户列表的最右边的操作中有禁用的按钮，点击此按钮就可禁用此用户，禁用后用户不可以登录到此系统。

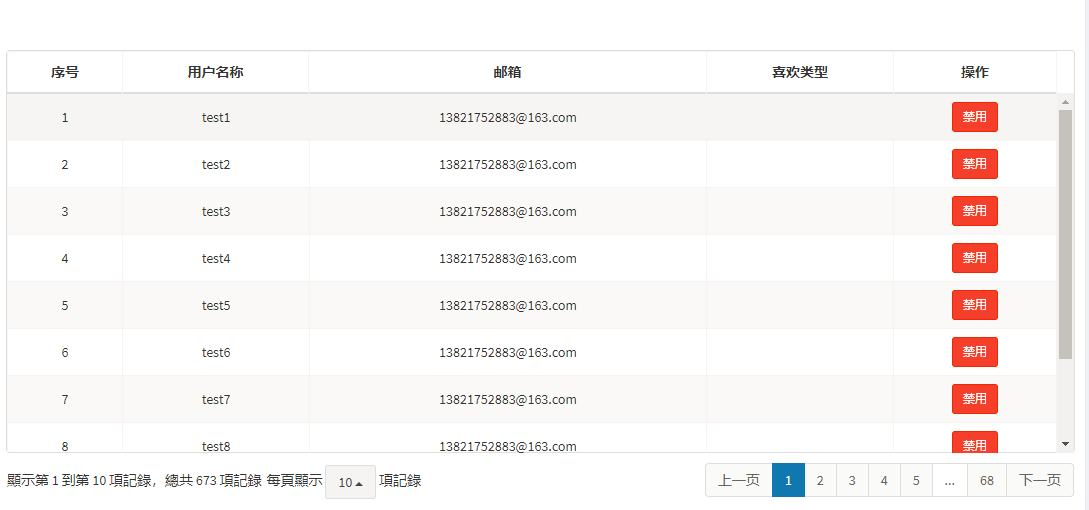


图4.16 用户管理界面

### 4.2.5 定时任务模块的实现

#### 1． 基于用户的协同过滤算法

基于用户的协同过滤算法主要包括两个步骤。

（1）找到和目标用户兴趣相似的用户集合。

（2）找到这个集合中的用户喜欢的，且目标用户没有听说过的物品推荐给目标用户。

步骤（1）的关键就是计算两个用户的兴趣相似度。设给定用户u和v，令N(u)表示用户u曾经有过正反馈的物品集合，令N(v)为用户v曾经有过正反馈的物品的集合。通过余弦相似度计算：

下面以用户行为记录为例，举例明UserCF计算用户兴趣相似度的例子：



图4.17 用户物品图

利用余弦相似度公式计算用户A和用户B的兴趣相似度为

同理，我们可以计算出用户A和用户C、D的相似度：

得到用户之间的兴趣相似度之后，UserCF算法会给用户推荐和他兴趣最相似的K个用户喜欢的物品，如下公式度量了UserCF算法中用户u对物品i的感兴趣度：

其中，包含和用户兴趣最接近的个用户，是对物品有过行为的用户集合，是用户和用户的兴趣相似度，代表用户对物品的兴趣，因为使用的是单一行为的隐反馈数据，所以所有的=1。

#### 2． 算法实现

在实现推荐算法的过程中考虑到每次都要从数据库获取到数据，再计算，最后存储到数据库，这样访问数据库有大量的系统资源和时间的消耗，为了解决此问题，我们不采用Python语言来实现，而是采用Oracle的存储过程。这样大大的提升了计算速度。

相似矩阵的计算：

首先删除数据库中要计算的用户原来的相似矩阵，其次遍历所有用户计算两个用户之间的相似度：



图4.18 相似用户计算流程图

#### 3． 定时任务实现

我们采用APScheduler，APScheduler是基于Quartz的一个Python定时任务框架，实现了Quartz的所有功能，使用起来十分方便。提供了基于日期、固定时间间隔以及crontab类型的任务，并且可以持久化任务。

我们的定时任务不断的判断是否要更新推荐列表，当用户对某个视频的评分改变之后，这条记录在数据库表示为需要更新，定时任务查询到后调用存储过程来更新该用户的推荐列表。最后更新该条记录为不再需要更新的状态。

定时任务还需要实现另一个功能，当新视频上架之后，什么情况下不是新视频。这个功能我们也用定时任务来完成，规定当超过十个用户观看之后取消新视频的标志。

## 4.3 本章小结

本章从系统的实现方面进行了讲解，对复杂的逻辑给出了流程图，每个实现还有文字描述，至此，我们的系统开发阶段完成。

# 第五章 个性化视频推荐系统测试

到目前为止，我们从需求分析，设计，编码方面已经做了讲解，个性化视频推荐系统已经初步完成。但是为了此系统的正常使用，需要对系统进行测试，这也是软件开发过程中必不可少的一个环节。针对系统的特点和实际情况，本系统采用手工的方式对系统测试和验收，测试的内容主要为单元测试和兼容性测试。

## 5.1 单元测试

为了保证划分的各个模块能够协同工作，给用户提供准确适合用户用户的推荐服务，首先对各个模块进行单元测试(又称模块测试)，针对软件设计中的最小单位-程序模块，进行正确性检查的测试工作。这些被测试的方法包含用户前台操作的模块，系统推荐的模块和定时任务更新推荐数据的模块。

### 5.1.1 用户界面测试

用户界面测试的基本流程为：对系统页面中的按钮，链接，输入框和搜索框进行点击测试，看能否点击输入等操作。具体的方案如表5.1：

表5.1 用户界面测试

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 位置 | 测试用例 | 预期结果 |
| 1 | 主页 | 点击登录按钮 | 弹出登录框 |
| 2 | 登录框 | 点击注册按钮 | 弹出注册框 |
| 3 | 登录框 | 点击登录提交按钮 | 登录框消失，显示主页 |
| 4 | 注册框 | 点击注册提交按钮 | 注册框消失，显示登录框 |
| 5 | 主页视频列表 | 点击视频名称 | 右边显示视频详细信息 |
| 6 | 主页导航栏 | 点击搜索按钮 | 主页视频列表显示搜索的视频 |

### 5.1.2 系统基本功能测试

老用户可以直接登录，当然新用户先注册在登录，登录成功后显示用户基本信息，用户在观看视频之后可以对视频做出评价打分，推荐引擎根据你的评价给你推荐相应的视频。并且系统提供搜索功能，进行关键字搜索，方便用户查找。

表5.2 系统基本功能测试

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 位置 | 测试用例 | 预期结果 |
| 1 | 登录框 | 输入正确的用户名和密码 | 登录框消失，主页显示用户的基本信息 |
| 2 | 登录框 | 输入不正确的用户名或密码 | 弹出提示信息，用户重新填写信息再登录 |
| 3 | 注册框 | 填写注册信息并提交 | 校验成功后弹出登录框登录 |
| 4 | 主页视频详情 | 点击评分按钮 | 评分成功给出提示 |
| 5 | 主页视频详情 | 点击播放按钮 | 播放视频 |
| 6 | 主页视频列表 | 点击更多按钮 | 展示当前分类的下一页视频 |
| 7 | 主页视频列表 | 点击推荐按钮 | 推荐引擎推荐更多视频 |

### 5.1.3 系统推荐功能测试

推荐功能测试主要测试推荐引擎是否按照之前设计的推荐逻辑来推荐：

表5.3 系统推荐功能测试

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 位置 | 测试用例 | 预期结果 |
| 1 | 主页->导航->推荐 | 用户没有登录 | 推荐新视频和评分较高的视频 |
| 2 | 主页->导航->推荐 | 用户登录没有观看记录 | 按照用户注册时的爱好推荐 |
| 3 | 主页->导航->推荐 | 用户登录有观看记录 | 按照推荐算法推荐 |

### 5.1.4 管理功能测试

管理功能测试主要对视频的增删改功能进行测试。

表5.4 管理功能测试

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 位置 | 测试用例 | 预期结果 |
| 1 | 管理页面->视频管理 | 点击删除按钮 | 弹出确认框确认是否删除 |
| 2 | 管理页面->视频管理 | 点击修改按钮 | 弹出对话框回显该视频的详细信息 |
| 3 | 管理页面->视频管理 | 点击增加按钮 | 弹出对话框，等待用户填写视频信息 |
| 4 | 管理页面->用户管理 | 点击禁用按钮 | 禁用当前用户 |

## 5.2 兼容性测试

Bs系统的架构是浏览器/服务器结构，对用户来说就只要一个浏览器就可以使用该系统，经测试在常用的Google，火狐等浏览器中可以正常显示运行，因此，兼容性测试通过。

## 5.3 本章小结

本章对系统的各个功能进行了全面的测试，所有的功能都达到了预期的效果。考虑到用户所用的浏览器可能不同，因此对兼容性进行了测试。到此，本系统开发完成。

# 总结与展望

个性化视频推荐系统已经按最初的设计开发完成。系统允许用户注册为新用户，根据类别查看自己感兴趣的视频，然后对视频做出评分，本系统也给用户推荐恰当的视频，当然用户可以在搜索框中搜索具体的视频来观看。在推荐的时候采用基于用户的协同过滤算法给用户推荐。为了克服冷启动问题，我们注册的时候收集用户的爱好，客户用户冷启动问题，在视频分类当中，顶置最新的视频，来克服新视频的冷启动问题。后台提供方便的界面操作，对视频进行管理。

由于个人能力和时间有限，有些功能点不够完善[1]，还需要进一步的设计完善。在界面当中，因为收集的数据的限制，没有视频的缩略图，只展现了视频的名称，不能给用户一个更好的体验。在推荐算法方面，只采用了单一的基于用户的协同过滤算法，收集的用户的日志也只有评分信息，因此在推荐算法方面还有很多需要改进的。

# 参考文献

1. 颜良炤. 基于Mahout视频推荐系统的研究与实现[D]. 北京: 北京交通大学, 2016.
2. 徐璐. 基于Web挖掘的视频推荐系统分析与实现[D]. 南京: 南京邮电大学, 2016.
3. 谢发川. 基于数据挖掘的视频推荐系统建模研究[D]. 成都: 电子科技大学, 2012.
4. 李姗姗. 基于协同过滤的视频推荐系统设计[D]. 南京: 南京邮电大学, 2017.
5. 刘晓伟. 基于协同过滤视频推荐系统的设计与实现[D]. 黑龙江: 黑龙江大学, 2015.
6. 曹丹丹,乐嘉锦,夏小玲. Redis数据库在视频推荐服务系统中的应用[J]. 计算机与现代化, 2013, 卷缺失(10): 94-97.
7. 卜旭松. 基于物品协同过滤的个性化视频推荐算法改进研究[D]. 宁夏: 宁夏大学, 2015.
8. 李冰. 基于Hadoop的个性化视频推荐系统的设计与实现[D]. 北京: 北京工业大学, 2015.
9. 贺栋博. 视频推荐系统数据分析决策平台[D]. 广东: 华南理工大学, 2012.
10. 高睿. 基于深度神经网络的视频个性化推荐系统研究[D]. 深圳: 深圳大学, 2017.
11. 赵楠. 基于Mahout的推荐系统实践及算法改进[D]. 西安: 西安电子科技大学, 2015.
12. 夏林. 基于Hadoop的视频推荐系统的研究与应用[D]. 上海: 上海交通大学, 2015.
13. 钟前建. 基于Hadoop和Mahout的视频推荐技术研究及应用[D]. 广东: 广东技术师范学院, 2016.
14. 蹇易. 基于混合方法的个性化视频推荐技术研究与实现[D]. 武汉: 华中科技大学, 2012.
15. 丁阳. 基于web用户体验的推荐系统设计与实现[D]. 北京: 北京邮电大学, 2013.
16. 唐明. 基于用户偏好的视频推荐技术研究[D]. [出版地不详]: 电子科技大学, 2013.
17. 李默,梁永全,赵建立,等. 面向网络视频的组合推荐系统模型研究[J]. 计算机工程与设计, 2013, 卷缺失(12): 4379-4383.
18. 张永霞. 面向用户个性化推荐的日志分析系统设计与实现[D]. 北京: 北京邮电大学, 2013.

# 致 谢

四年的时间很快，我们马上毕业了，就连最后的毕业论文也写到结尾了。在这里，我向在这四年里帮助过我，激励过我的老师，同学们表达我最诚挚的感谢。

在推荐系统方面，以前了解的几乎为零，但是希望自己可以多学习，多了解，因此选择了个性化视频推荐系统，首先非常感谢我的导师肖迎元教授愿意把此课题给我。从论文的开题到最终的定稿，肖老师一直在给我们精心的指导。选题之后就开始教我们怎么查阅资料，阅读文献，为开题报告提前做好准备。在中期设计开发过程中，给我们提供思路，算法的讲解。以及后期的论文的书写，指导我们写作的规范，论文的书写结构等等。还有研究生的学长，给我们指导怎么获取数据。非常感谢在这段时间老师和学长的帮助，让我顺利完成毕业论文。

祝大家身体健康，工作顺利。