单位代码：10293 密　级：

南京郵電大學

专 业 学 位 硕 士 论 文

论文题目：基于微信小程序的智能推荐点餐系统的设计与实现

学　　号	1217012216
姓　　名	李昊
导　　师	李涛
专业学位类别	工程硕士
类　　型	全日制
专业（领域）	电子与通信工程
论文提交日期	二〇二〇年五月

Design and Implementation of Intelligent Recommendation Ordering System Based on WeChat Small Program

Thesis Submitted to Nanjing University of Posts and Telecommunications for the Degree of

Master of Engineering

By

Li Hao

Supervisor: Associate Prof. Li Tao

May 2020

摘要

近年来，互联网行业飞速发展，许多传统行业开始与互联网结合并通过数字化的改造、转型与升级创造出新的发展生态。尤其在国人最关注的“吃”的方面，餐饮行业通过将点餐、结算等一系列的服务流程从线下搬运到线上，使得顾客使用移动设备即可享受到更加方便快捷自由的服务。但是当前的很多点餐应用只是将原有的文本信息数字化，并没有让顾客得到个性化的服务。纵观其他电子商务行业的应用例如淘宝和京东已经能够通过推荐系统为用户提供个性化的推荐服务，为用户自动找出可能感兴趣的商品，而现在的点餐应用还不具备这种功能。为此设计出一个点餐系统，并在其中加入推荐系统来为在餐厅内点餐的用户提供个性化的推荐服务，该推荐系统使用了对基于协同过滤推荐算法改进的混合推荐算法。

传统的基于协同过滤的推荐算法存在着数据稀疏性高、推荐解释性差以及冷启动等问题。根据这些问题，可以使用基于内容和基于关联规则的推荐算法的特性对基于协同过滤的推荐算法进行改造。其核心思想是通过基于关联法则的算法对物品进行评分预测并使用此预测评分填充数据集降低数据的稀疏性，然后结合基于内容与基于协同过滤计算的物品相似度去预测用户评分并根据评分的高低给目标用户产生推荐物品。在为用户推荐菜品的时候，不能只靠推荐算法产生的结果，还需要结合一定的推荐策略来对物品进行组合排序筛选最后为用户推荐出合适数目且搭配合适的一系列菜品。因此设计出一种推荐策略，根据用餐人数和菜品的分类信息为用户推荐出数目合适且荤素搭配合理的一套菜品。

结合餐饮行业中消费者和商家的需求，设计并实现了一个包含点餐微信小程序、餐饮管理应用在内的智能推荐点餐系统。该系统不仅可以帮助消费者快速地找到自己喜欢吃的菜品和享受自由便捷的点餐服务，还可以帮助商家管理餐饮相关信息并且了解自己的运营情况。

关键词： 点餐，微信小程序，推荐系统 ，协同过滤，关联法则

Abstract

In recent years, with the rapid development of the Internet industry, many traditional industries begin to combine with the Internet and create a new development ecology through digital transformation and upgrading. Especially in the aspect of "eating" that Chinese people pay the most attention to, the catering industry transfers a series of service processes such as ordering and settlement from offline to online, so that customers can enjoy more convenient, fast and free services by using mobile devices. But many of today's ordering systems just digitize the original text information, instead of giving customers personalized service. Other applications in the e-commerce, such as Taobao APP and Jingdong APP, have been able to provide users with personalized recommendation services through the recommendation system which can automatically find out the commodities they may be interested in, while the current ordering systems do not have such functions. For this reason, an ordering system in which a recommendation system is added to provide personalized recommendation services for users ordering food in the restaurant has been designed. This recommendation system uses a hybrid recommendation algorithm based on collaborative filtering recommendation algorithm.

Traditional collaborative filtering-based recommendation algorithms have problems such as high data sparseness, poor recommendation interpretation, and cold start. According to these problems, the characteristics of recommendation algorithms based on content and association rules can be used to transform the recommendation algorithm based on collaborative filtering. The core idea is to predict the score of items by the algorithm based on association rules and use the predicted score to fill the data set to reduce the sparsity of the data, then predict the user score by combining the similarity of items calculated based on content and collaborative filtering, and generate recommended items for target users based on the score. When recommending dishes for users, we should not only rely on the results generated by the recommendation algorithm, but also need to combine certain recommendation strategies to sort and select the combination of items and finally recommend a suitable number of dishes for the user. Therefore, a recommendation strategy is designed to recommend to the user a set of dishes with a proper number and a reasonable combination of meat and vegetables according to the number of diners and the classification information of the dishes.

According to the needs of consumers and businesses in the catering industry, an intelligent recommended ordering system including a WeChat mini-program for ordering food and catering

management applications has been designed and implemented. The system can not only help consumers quickly find their favorite dishes and enjoy free and convenient ordering services, but also help merchants manage catering related information and understand their own operating conditions.
Keywords: Ordering, WeChat applet, Recommendation system, Collaborative filtering, Association rule

目录

专用术语注释表

缩略词说明：

APP	Application	（计算机或者手机）应用程序
AOP	Aspect Oriented Programming	面向切面编程
ANSI	American National Standards Institute	美国国家标准学会
JDBC	Java DataBase Connectivity	Java 数据库连接
IoC	Inversion of Control	控制反转
API	Application Program Interface	应用程序接口
HTTP	Hyper Text Transport Protocol	超文本传输协议
HTTPS	Hyper Text Transfer Protocol over SecureSocket Layer	超文本传输安全协议
JDK	Java Development Kit	Java 语言的软件开发工具包
ID	Identity	用户身份识别号

第一章 绪论

1.1 课题研究背景

随着互联网的高速发展，信息的传输速度越来越快，信息的种类和数量越来越多，随之带来了信息过载[1]的问题。每天各种消息的发布和传送使得人们需要通过花费大量的时间过滤消息才能获取到高价值且对自己有意义的信息。目前，搜索引擎[2]和推荐系统[3]都是能够帮助人们获取自己所需信息的重要工具。例如在电商领域，商品的数量越来越多，种类越来越丰富，消费者通常需要通过使用搜索引擎去寻找自己想要的商品，要不然根本无法快速在海量的商品信息中找到自己需要的那一个。而推荐系统的应用则可以减少消费者搜索寻找的步骤，并能够直接将其可能感兴趣的商品展示出来。使用搜索引擎这种被动的方式来检索信息需要花费相当的时间和精力，而且得到的结果并不一定是准确的。但是推荐系统能够发现人们关心的，感兴趣的信息，且自动将这些信息推送过来。目前，推荐系统已经被广泛应用在了电商、在线音视频、新闻阅读、在线交友等领域中[4]-[6]。

通常情况下，人们去餐饮店就餐首先需要进行点餐。传统的点餐方式是服务员递来菜单，顾客一边阅读菜单一边告诉服务员需要的菜品。这种点餐方式存在效率低的问题，特别是在客流量大时，由于服务员数量不够，导致部分顾客需要等待，如果等待时间较长，则会使得顾客的消费体验变差并给出不良评价，最终导致餐饮店的经营业绩下滑。如今，网络购物已经成为常态，人们的消费方式和消费观念也已发生了巨大改变，从传统的线下购物方式逐渐转移到线上购物方式。而现在餐饮行业也受到了电商发展的影响并随着智能手机的普及以及移动互联网的快速发展，不仅产生了像美团、饿了么这种外卖形式的点餐 APP，使用它们使人们不必出门便能享受到餐厅的美食，也出现了一种在餐饮店内使用在线点餐系统点餐的新方式。这种新型的点餐方式让人们只需操作手机便可以完成点餐、确认订单、支付等一系列服务流程，这相对于传统的需要服务员配合的点餐方式显得更加自由和方便，并且使用这种新型的点餐方式还可以减少餐厅的人力和运营成本，节约点餐时间，达到提高服务效率和提升顾客体验的目的。

虽然在线点餐系统已经开始使用，但目前也只是将原来纸质餐单上的内容通过数字化的形式呈现给顾客，顾客还是需要一边浏览菜单一边选择所需要的菜品，而在浏览菜单中各种各样的美食时，顾客往往又会花费大量的时间来思考菜肴的搭配，这种情况反而使得点餐的时间更长。基于此，可以考虑将推荐系统融合到在线点餐系统中，形成智能的点餐系统[7]。这

样，消费者可以获取到由系统生成的个性化菜单，从而使点餐变得简单快捷，也能获得良好的用餐体验；餐厅管理者也可以通过使用此系统来了解用户的需求和偏好并减少人力资源需求，从而能够及时调整运营决策和降低运营成本。近年来，随着微信用户量大幅增加以及微信小程序的快速发展[8]，将应用部署在微信小程序上成为一种趋势。并且微信小程序不需要用户单独安装，也不需要用户注册账户，即用即开非常方便，特别适合在点餐场景中使用，因此可以使用小程序作为点餐平台。

1.2 国内外研究现状

1.2.1 点餐系统研究现状

点餐系统通过利用现代化的设备并借助互联网技术来为餐饮店提供一种新型的点餐方式。国外在餐厅建设电子化点餐平台的起步较早，在 1970 年就出现了餐饮电子服务管理系统，德国在 2007 年出现了世界第一个无人餐厅，餐厅中的所有服务都没有人的参与，顾客可以通过店内的计算机设备完成菜品浏览，点餐与结算。日本的一家餐厅在此无人餐厅的基础上，结合移动互联网技术，将点餐系统移植到了智能终端上。顾客可以使用餐厅专属的 APP 预约点菜，这种方式节约了顾客的等待时间，并且展示了信息时代智能点餐系统给人们生活带来的便捷。从历史发展来看，国外点餐系统的发展主要有以下几个阶段：餐厅内有线点餐系统，餐厅内无线点餐系统，基于互联网的无线点餐系统。餐厅内的有线点餐系统主要是由服务员使用餐厅内的电脑完成点餐及其他服务。餐厅内的无线点餐系统则需要服务员通过部分手持终端替代固定的计算机设备完成点餐及其他服务。而目前使用最普遍的基于互联网的无线点餐系统，其只需要用户使用智能手机就可以完成点餐、结算等服务。例如，如今在肯德基内点餐，顾客就可以实现在手机上浏览菜品，下单并进行支付。

目前国内正在推动“互联网+”的建设，而与人们生活密切相关的餐饮行业自然也与互联网技术进行着深度融合[9]。不仅出现了如美团、饿了么这种以外卖为主的点餐软件，还诞生了在餐厅中使用手机自主点餐的新方式[10]-[11]。其中使用手机自主点餐还分为使用特定 APP 点餐[12]和扫描二维码点餐。而使用特定 APP 的方式略显繁琐，在不同的餐厅用餐需要下载不同的 APP 使用。相比之下，扫描二维码点餐则更加方便快捷，无需下载应用且也为应用开发者省去了适配不同手机系统的麻烦。近年来，随着微信的快速发展，基于微信平台[13]的应用层出不穷，许多商家也将点餐系统部署到了微信平台上[14]，顾客可以从微信直接登录到点餐系统中进行菜品浏览，下单与支付，这使得点餐变得更加方便与快捷[15]。

1.2.2 推荐系统研究现状

在上个世纪末，国外提出推荐系统的概念，由 GroupLens[16]研究小组提出的名为 MoviesLens[17]的电影推荐系统使得推荐系统成为了一个独立的研讨方向。在这之后的十几年中，推荐系统不断发展，并出现了许多基于不同算法的推荐系统，主要有基于协同过滤模型的推荐系统[18]、基于内容的推荐系统、基于知识的推荐系统以及混合多种算法集成的推荐系统。

由 Greg Linden[19]等人主持开发的 Amazon 购物系统应用了基于协同过滤的推荐算法。系统通过记录用户购买和评分的物品，然后选取与之相似度最高的物品集合并将其推荐给用户。Sarwar[20]等人提出的基于 SVD 的协同过滤算法，通过 SVD 矩阵分解，实现对高维稀疏矩阵的降维与填充，提高了推荐准确度。Rafailidis[21]等人提出了一种基于张量分解与标签聚类的推荐算法，优化了数据稀疏性和“冷启动”的问题。Nasiri[22]等人提出了一种新的基于矩阵分解的算法，针对协同过滤算法中出现的扩展性及数据稀疏性问题进行了优化。

目前在国外，推荐系统被广泛应用于电子商务、在线音乐、在线视频、新闻等平台中。其中非常著名的有大型电子商务平台 Amazon，搜索引擎 Google，社交网站 Facebook 及在线视频平台 YouTube。

反观国内，推荐系统的相关研究工作稍有滞后，与国际研究水平存在一定的差距，但是推荐系统的浪潮仍然对国内的众多互联网公司产生了较大影响。随着国内互联网以及电商的快速发展，推荐系统的研究也得到了较快的发展。贾冬艳[23]等人在基于协同过滤的推荐算法的基础上设计了一种多维信任模型，此模型通过相关参数对用户的信誉评级进行衡量，然后通过信任模型和协同过滤算法相结合来选择目标用户进行推荐。陈浩[24]等人提出了一种优化用户相似度的算法，用来解决在协同过滤算法中相似度衡量失真的问题。

如今国内许多大型公司也在大规模的使用推荐系统，例如同样作为广泛应用的搜索引擎，百度在它的多个业务中加入了个性化推荐服务；国内最大电子商务公司阿里巴巴旗下的电商网站（淘宝网、天猫等）[25]加入了数十项个性化推荐的应用场景，通过推荐交互和反馈优化等技术手段，为用户实时提供高质量的各类推荐信息。个性化音乐网络电台——豆瓣电台，通过分析用户一定时间的反馈，形成用户的兴趣模型，从而使电台的个性播放内容更加地契合用户的听歌爱好。

在推荐系统的研究中，总体分为对用户日志的研究、推荐引擎的研究以及推荐策略的研究。其中推荐引擎即推荐算法的研究是最为核心的研究方向，目前国内外学者的研究主要集中在数据稀疏性、算法扩展性以及相似度量准确性等问题上[26]-[28]。总的来说，国内外现有的

推荐系统在并发数据处理量以及推荐精准度等方面都有很大的改进空间。推荐系统所涉及的协同过滤算法优化、相关性计算问题、系统冷启动问题等多个方面，都需要继续进行深入的理论研究，并尽可能与实际应用相结合。

1.3 本文的研究目标和内容

为了让用户在餐饮店内点餐更加方便自主以及让商家对餐厅实现信息化管理，本文设计并实现了一个智能点餐系统，该系统包括手机端的微信点餐小程序、网络端的餐饮管理应用以及为用户推荐菜品的离线推荐系统。

本文主要研究内容包括以下几个方面：

（1）设计并实现微信点餐小程序。通过对用户点餐行为的分析，确定小程序的功能；学习微信小程序开发技术，设计小程序的用户界面，实现点餐流程的所有功能；结合离线推荐系统，为用户提供快速点餐功能。

（2）设计并实现餐饮管理系统。通过分析商家对餐饮管理的需求，设计并实现用户信息管理、菜品信息管理，订单信息管理，销售统计等功能，帮助商家更方便地了解店铺的运营情况。

（3）研究基于协同过滤的推荐算法，分析其存在的问题。并在研究其他推荐算法的基础上，利用它们的核心思想对基于协同过滤的推荐算法进行改进从而提升推荐质量。

（4）设计并实现离线推荐系统。在采集到用户的历史行为数据并对其进行预处理后，使用改进后的推荐算法分析挖掘出用户的喜好，并为不同的用户推荐其可能喜欢的菜品。

1.4 论文组织结构

第一章绪论。主要介绍本文的研究背景以及国内外在点餐系统及推荐系统上的研究现状，分析当前点餐系统和一些推荐算法的局限性，并对本文的研究目标与内容作出概述。

第二章系统相关技术概述。主要对本文中用到的一些应用技术和相关理论作出简要介绍，包括系统开发中用到的开发框架、推荐系统及常见的推荐算法。

第三章主要介绍了基于物品的协同过滤、基于内容和基于关联法则的推荐算法，对其主要推荐流程做了详细介绍并指出它们的优缺点。然后针对基于物品的协同过滤推荐算法存在的问题结合其他两种推荐算法的特性对它进行改造形成一种混合推荐算法。最后使用 movieLens 数据集对改造后的算法进行比较测试，结果表明该混合推荐算法的推荐性能要比原来的基于物品的协同过滤算法要好。

第四章为系统的设计与实现部分，首先根据系统所需的功能对其进行整体架构设计，并划分了功能模块。然后对各个功能模块进行分析设计并加以实现。

第五章为系统测试与总结。首先从功能上对各个主要应用端进行测试，检查是否能按照需求正常工作。然后对服务器进行性能测试，检查其是否能满足并发要求。最后对全文工作进行总结，主要对文中的研究内容与成果进行了总结并对其中存在的问题以及不足之处进行了说明。

第二章 系统相关技术概述

智能推荐点餐系统主要由四个部分组成：移动端的微信点餐小程序、网络端的餐饮管理应用、离线推荐系统以及服务器端组成。在系统实现上，面向消费者的微信小程序采用 uni-app 前端框架并且使用 HbuilderX 和微信开发者工具对其进行开发。网络端的餐饮管理应用使用 X-admin 前端框架来开发用户交互界面。离线推荐系统使用 Python 语言进行开发，使用改进后的推荐算法来为用户提供菜品推荐服务。服务器端，使用 Java 语言和 SpringBoot 框架以及数据库来为移动端和网络端的应用开发后台服务程序。本章下面将对餐饮平台开发中涉及到的框架技术、数据库以及离线推荐系统中所涉及到的推荐算法和工作流程进行简要介绍。

2.1 餐饮平台开发相关技术

2.1.1 微信小程序

微信小程序是一种可以运行在微信中的应用服务。在微信公众平台中，“小程序”是指一种新的开放能力，通过其提供的一个简单、高效的应用开发框架和配套的组件及 API，可以帮助开发人员开发出可以运行在微信中的具有原生 APP 体验的服务，这种服务通常被称为微信小程序。微信小程序不需要用户单独安装，用户需要使用时，可以通过在微信中搜索小程序名称找到其打开使用或者通过微信扫一扫功能扫描小程序二维码打开使用。除此之外，用户还可以通过点击其他微信好友分享的小程序链接打开使用。

与传统的手机 APP 相比，微信小程序无需用户安装，即用即开，不占用手机存储空间。并且当用户不再使用时直接关闭即可，不必卸载软件，实现了用完即走的理念。从开发者角度来看，微信小程序的开发成本比 APP 低，开发周期比 APP 短，可以更快地上线应用服务。

2.1.2 uni-app 框架

uni-app 是一个使用 Vue.js[29]和微信小程序 API 开发所有前端应用的框架，开发者编写一套代码，经过编译可以发布到不同的平台上，包括 iOS、Android、H5（移动端网页）以及各种小程序（微信、支付宝、百度等）。DCloud（数字天堂网络技术有限公司）最先于 2012 年开始研发小程序技术并推出了技术标准，但由于微信、支付宝等公司订制了自己的标准导致了小程序开发者想要在不同平台上上线具有同样功能的小程序就要学习各种不同的小程序技术标准，这增加了开发者的负担。因此，DCloud 推出了 uni-app 框架，通过这个框架能够为开发者抹平各平台的差异。uni-app 功能框架如图 2.1 所示。

从功能框架图中可以看到，uni-app 能够将应用编译发布到 APP 平台、小程序平台以及

H5 平台，即实现了跨平台应用发布。图中 uni 内置组件和 API 一层，将常用的组件和 API 进行了跨平台封装，可以覆盖大部分的业务需求。图中 APP 平台中的 HTML5Plus 表示的是一个跨 iOS 和 Android 的 JavaScript 增强引擎。

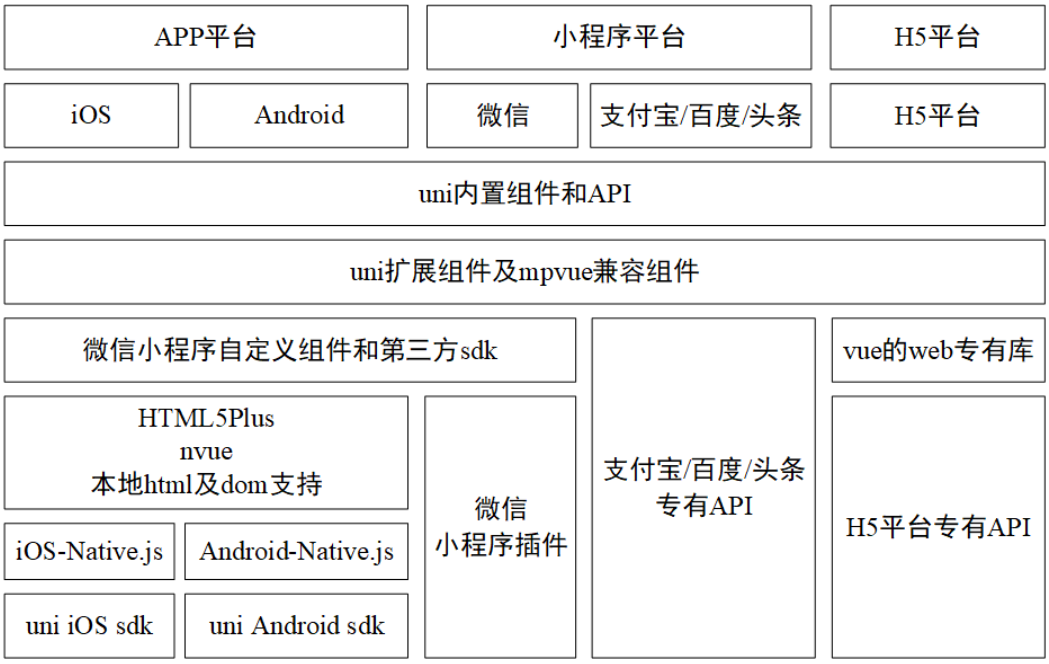

图 2.1 uni-app 功能框架图

相对于微信小程序自身的开发框架来说，使用 uni-app 框架更加简单。开发者使用 uni-app 框架，可以采用 Vue.js 的语法和微信小程序 api 来进行程序开发，这对于前端工程师来说并不需要额外的学习成本。另外，使用该框架之后，开发者写的一套代码不仅可以运行在微信小程序平台中还可以运行在其他小程序平台上，方便了日后项目运营的扩展。

2.1.3 SpringBoot 框架

SpringBoot 框架[30]是在 2013 年开始研发并在 2014 年发布第一个版本的一个开源的轻量级的 Java 开发框架。它是在 Spring 框架[31]的基础上设计研发的，其不仅继承了 Spring 框架原有的优秀特性，而且通过简化配置进一步简化 Spring 应用的整个搭建和开发过程。要想了解 SpringBoot 框架，就必须先了解 Spring 框架。

Spring 框架是于 2003 年兴起的一个用于 Java 开发的一站式框架，其特点是：分层、开源、轻量。此框架是为了解决企业应用开发的复杂性而创建出来的。它的分层架构允许使用者自由选择使用哪一种组件来进行开发，并且还提供了在展现层、持久层和业务层中能够使用到的企业级应用技术。Spring 体系结构如图 2.2 所示。

从图中可以明显看出，Spring 框架是一个分层架构，它包含一系列的功能要素并被分成大约 20 个模块。最底层只有一个测试模块，其主要作用是方便开发者对程序进行测试。往上的核心容器层（Core Container）主要包含了四个模块，它是 Spring 框架中最基础的部分，提

供了 IoC 和依赖注入功能。AOP 模块提供了面向切面编程的实现，允许自定义方法拦截器和切入点，并对一些方法进行增强。数据访问/集成层（Data Access/Integration）提供了与数据库有关的 JDBC 抽象层、流行的对象关系映射 API 以及与事务管理相关的功能。网络（Web）层包括一些与 web 开发相关的功能模块，可以提供面向 web 的基本功能和面向 web 的应用上下文，例如文件上传功能、使用 Servlet 监听器初始化 IoC 容器等。

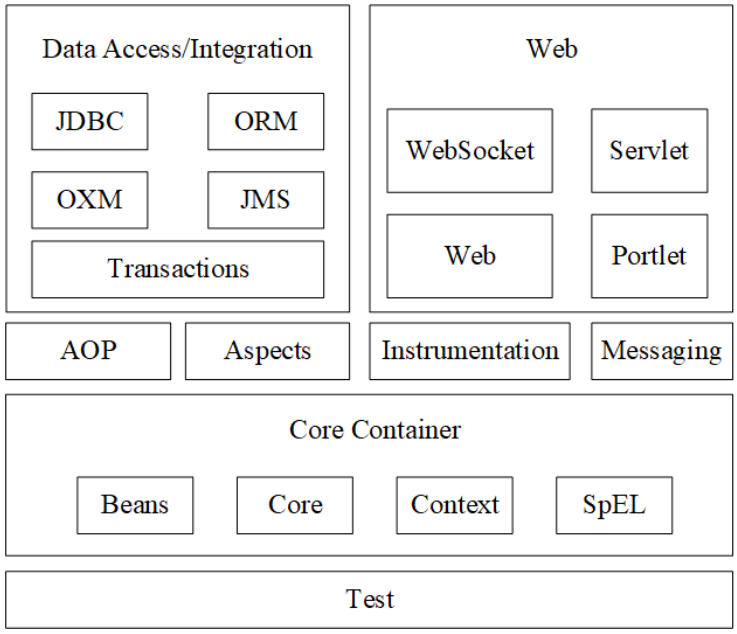

图 2.2 Spring 体系结构图

Spring 相比与传统开发方式的优势有：

（1）方便解耦，简化开发。

（2）AOP 编程的支持。

（3）声明式事务的支持。

（4）方便程序的测试。

（5）方便集成各种优秀框架。

（6）降低 JavaEE API 的使用难度。

但是 Spring 配置比较繁琐，而 SpringBoot 的出现则解决了这个问题。使用 SpringBoot 框架可以轻松创建出独立的 Spring 应用程序，它还内嵌了 Tomcat、Jetty 等 web 容器，不需要部署 war 文件和安装 web 容器软件，并且提供一系列的启动文件来简化配置，不需要添加很多依赖文件。

2.1.4 数据库

数据库是一种按照数据结构来组织、存储和管理数据的仓库。每个数据库都有一个或多个不同的 API 用于创建、访问、管理、搜索和复制所保存的数据。本文中设计的系统中用到

了 MySQL 数据库和 Redis 数据库，下面将对这两个数据库做简要的介绍。

MySQL[32]由瑞典 MySQL AB 公司开发，目前属于 Oracle 公司。它是目前最流行的关系型数据库之一，被经常使用在 web 应用项目中。所谓关系型数据库，是指一种建立在关系模型基础上的数据库，它可以借助于集合代数等数学概念和方法来处理数据库中的数据。关系型数据库不是将所有数据都放在一个大仓库中，而是将数据分散保存在不同的表中，这样将会增加数据查询的速度。MySQL 是开源的，免费使用，但其性能很好，可以处理拥有上千万条记录的大型数据表。MySQL 支持标准的 SQL 语言来对它进行控制管理，并且其能够运行在多种系统之上，同时还支持多种编程语言例如 Java、Python 等对它进行操作。

Redis 和 MySQL 不同，它是一个高性能的非关系型数据库[33]。NOSQL（Not Only SQL），意思为“不仅仅是 SQL”，是一项全新的数据库理念，泛指非关系型数据库。NOSQL 是为了应对大规模数据集合及多重数据种类带来的挑战而产生的。它与关系型数据库相比，安装部署更加简单，存储数据的格式更加丰富，扩展性更强，并且由于 NOSQL 将数据存储于缓存之中，所以查询速度更快。Redis 是使用 ANSI C 语言编写的一个键值对（key-value）型的数据库，它也是开源的。Redis 不仅可以将数据存储在缓存中，也支持数据的持久化即将缓存中的数据保存到磁盘中。在性能上，Redis 具有读取数据的速度最高可达 110000 次/s，写入数据的速度最高可达 81000 次/s 的极高性能。在使用上，Redis 经常被用来缓存应用的数据，并且辅助秒杀、抢购等活动。

关系型数据库和非关系型数据库通常是互补的关系而不是对立的关系。在本文设计的系统中，使用关系型数据库 MySQL 存储应用的基本数据，使用非关系型数据库 Redis 存储一些需要经常查询且没有太多变化的数据。

2.2 推荐系统相关技术

近年来，随着互联网行业的发展，人们发布信息的平台越来越多，发布信息的方式越来越丰富，发布信息的频率越来越高，从而产生了大量的信息，社会也因此进入了一个信息爆炸的时代。为了让人们在纷繁复杂的信息中快速得到他们感兴趣的内容，推荐系统应运而生。推荐系统，可以通过分析用户的历史行为记录，发现用户的个性化需求从而找到其感兴趣的内容。通常一个推荐系统按照功能主要分为数据采集模块、推荐引擎模块和推荐排名模块三个部分。

数据采集模块，主要作用是收集用户的历史行为记录数据并进行数据的预处理[34]。用户的历史行为记录数据主要包括用户的点击浏览、收藏、购买、打分等行为产生的数据。在采

集到用户的有关数据后，需要对一些无用的、错误的、异常的数据项进行预处理，这些预处理通常包括直接删除、使用其他值替代等方法。

推荐引擎模块，是推荐系统的核心，其主要功能是通过各种推荐算法对用户产生的相关数据进行分析与挖掘，发现用户的特点并为其推荐可能感兴趣的内容。

推荐排名模块即推荐策略[35]，此模块主要是将从推荐引擎模块中得到的初步推荐结果进行进一步的处理并产生最终的推荐内容列表。由于推荐引擎中可能包含多种推荐算法，每种算法产生的推荐结果不尽相同并且在不同应用场景中的推荐策略也不尽相同，所以需要经过一定的过滤、合并及排名操作产生最终的推荐结果。

推荐算法作为推荐系统中最核心的部分，下面将简单介绍本文设计的系统中所涉及到的几种算法。

2.2.1 基于内容的推荐算法

基于内容的推荐算法[36]的基本思想是先对一个物品的内容属性进行分析，建立特征，然后基于用户对某种特征的兴趣度以及一个物品具有什么特征来推荐相关物品。例如书籍可以分为人文、科幻、工具等类别，电影有爱情、搞笑、玄幻等分类。基于内容的推荐，就是根据这些物品的分类或内容属性并结合用户的购买或打分记录计算出该用户对不同分类或内容属性的喜好程度，然后再根据他的喜好匹配到其他类似的物品并将其推荐出来。

基于内容的推荐算法主要可以分为四个步骤：

（1）特征提取：首先分析并提取所有物品的特征，例如上面提到的电影分类标签。然后构建物品特征资料数据集。该数据集一般用来计算物品之间的相似度。

（2）用户偏好计算：利用用户对某一个物品的评分或者其他操作行为能够计算出用户对不同内容特征的喜好程度。举个例子，如表 2.1 所示，是两个用户对不同电影的评分情况，这些电影被分为喜剧和科幻类，“1”表示该电影属于该分类，“0”则表示不属于。“？”表示用户没有看过该电影，没有评分。用户 A 对于喜剧类电影的喜好程度可以通过其看过的所有喜剧类电影评分的平均分数来衡量，其喜好程度为 $(5+4+4)/3=4.3$。用户 A 对于科幻类电影的喜好程度为 $(3+2)/2=2.5$。由此可以看出，用户 A 更喜欢喜剧类型的电影。

（3）内容召回：在物品数据集中匹配到用户可能感兴趣的物品，将其取出放入预推荐物品池中。由上面的例子可以发现，用户 A 更喜欢喜剧类的电影，那么将会挑选出《泰囧》、《夏洛特烦恼》、《受益人》等喜剧类电影放入到预推荐物品池中。

（4）物品排序：该步骤是对预推荐物品池中的相关物品进行排序和再次挑选，排序的策略可以是根据物品的平均评分的高低也可以依据物品关注程度的高低。例如在上面推荐的电

影中，《泰囧》平均评分是 4.8 分，《夏洛特烦恼》是 4.5 分，《受益人》是 4.6 分，则《泰囧》会被优先推荐给用户 A。

表 2.1 用户评分矩阵

分类 / 电影名	喜剧	科幻	用户 A	用户 B
我不是药神	1	0	5	?
西虹市首富	1	0	4	2
港囧	1	0	4	3
流浪地球	0	1	?	5
盗梦空间	0	1	3	4
阿凡达	0	1	2	4

基于内容的推荐算法对推荐结果的解释性较强，用户易于理解并且其推荐方式简单有效，所以此算法通常会与其他推荐算法共同应用于推荐系统中。其主要优点有：

（1）由于物品的内容特征不依赖于用户数据，所以没有物品冷启动问题。而且由其推荐出的物品不会存在过于热门的问题。

（2）能为有不同兴趣爱好的用户进行精准推荐。

（3）原理简单，易于理解和定位问题。

2.2.2 基于近邻的协同过滤算法

协同过滤算法是通过对大量用户给出的评分进行协同处理后给出推荐。它的基本思想是由已知的评分去预估未知的评分然后根据预测评分的高低来推荐物品。基于近邻的协同过滤算法也被称为基于记忆的算法，这种算法基于一个事实，就是相似的用户会以相似的行为模式对物品进行评分，且相似的物品往往能获得相似的评分。基于近邻的协同过滤算法可以分为两类，一类是基于用户的协同过滤算法[37]，另一类是基于物品的协同过滤算法[38]。

基于用户的协同过滤算法其主要原理是先计算某一个用户与其他用户的相似度，然后选取与这个用户最相似的几个其他用户，把他们购买过或感兴趣的物品推荐给这个用户。举个例子，如表 2.2 所示，用户 A 购买了物品 a、b、c，用户 B 购买了物品 d，用户 C 购买了物品 b、c、e。通过比较用户 A、B、C 的购买记录，可以发现用户 A 与用户 C 都购买了物品 b 和 c，而用户 B 只购买了物品 d，因此可以判定用户 A 与 C 两个用户具有较高的相似性。所以在对用户 C 进行推荐的时候，可以将用户 A 购买过的物品 a 推荐给它。

表 2.2 用户购买记录

用户	购买记录
A	物品 a，物品 b，物品 c
B	物品 d
C	物品 b，物品 c，物品 e

基于物品的协同过滤算法与基于用户的协同过滤算法类似，不同的是基于物品的协同过滤算法利用的是物品之间的相似度而不是用户之间的相似度。它的推荐思想是：某个用户购买过物品 a 或对其感兴趣，则会为他推荐与物品 a 相似的其他物品。这个思想与基于内容的推荐算法类似，但是基于物品的协同过滤算法对物品间相似度的计算并不是依据物品的内容属性，而是通过分析用户的购买或其他行为记录去计算物品之间的相似度，因为该算法认为物品 a 和物品 b 相似是因为喜欢物品 a 的用户大都也喜欢物品 b。举个例子，分析表 2.2 的用户购买记录，统计出物品同时被购买的次数，如表 2.3 所示，“/”表示没有被同时购买。从表中可以看出物品 c 与物品 b 同时被购买的次数最多，所以物品 c 与物品 b 最相似。

表 2.3 物品同时购买次数

	物品 A	物品 B	物品 C	物品 D	物品 E
物品 A	/	1	1	/	/
物品 B	/	/	2	/	/
物品 C	1	2	/	/	1
物品 D	/	/	/	/	/
物品 E	/	1	1	/	/

2.2.3 基于关联规则的推荐算法

关联规则，是指不同物品之间的关联关系[39]，基于关联规则的算法是数据挖掘领域中的一项重要技术，它能够根据历史的统计数据发现物品之间的关联规则。在其应用中，最出名的当属“啤酒与尿布”的案例：据报道，美国的一家连锁超市发现，经常去超市为孩子买尿布的爸爸，通常会同时购买一些啤酒，啤酒与尿布本是两个不相关的物品，但在这种情况下却具有了一定的关联性。在推荐系统中，可以使用基于关联法则的算法在海量的用户行为记录中发现许多具有关联性的物品，从而可以将这些物品组成一个集合推荐给用户。如何找到物品之间的关联规则呢？这就需要先了解几个概念：支持度、频繁项集、置信度[40]。

支持度，指的是某一发生的事件在所有已发生的事件中所占的比例。例如在购物场景下，其表示一个物品被购买的次数在所有购买记录中所占的比例。如果一个物品的支持度大于一个预先设定好的阈值（最小支持度），则这个物品被称为频繁项。如果是多个物品组成的集

合，则它们被称为频繁项集。频繁项集可以为用户购买行为的关联性提供重要线索。

置信度，表示的是事件 X 发生的情况下事件 Y 发生的概率。拿“啤酒与尿布”的例子来说，就是在尿布被购买的情况下啤酒被购买的概率有多少。置信度可以用来衡量一个规则的强度。基于最小的支持度和最小的置信度可以定义一个关联规则。

下面举个例子来简要介绍基于关联法则的算法推荐物品的过程。表 2.4 是用户的购买记录，对用户购买过的物品分别计算支持度，结果如表 2.5 所示。

表 2.4 用户购买记录

用户	购买记录
A	豆浆，油条，鸡蛋
B	豆浆，包子，鸡蛋
C	豆浆，鸡蛋，煎饺
D	包子，小米粥，

表 2.5 商品的支持度

商品	支持度
豆浆	0.75
油条	0.25
鸡蛋	0.75
包子	0.5
煎饺	0.25
小米粥	0.25

假设预先设定的最小支持度为 0.5，那么支持度小于 0.5 的油条、煎饺、小米粥被剔除，然后计算豆浆、鸡蛋、包子两两组合后的支持度，结果如表 2.6 所示。将支持度小于 0.5 的组合剔除，然后计算剩下商品组合的置信度，结果如表 2.7 所示。假设预先设定的最小置信度为 0.75，则表 2.7 中的组合规则都是基于最小支持度为 0.5 且最小置信度为 0.75 的关联规则，且此规则为强关联规则，即购买过豆浆的用户很可能也会购买鸡蛋，购买鸡蛋的用户也很有可能购买豆浆。

表 2.6 商品组合的支持度

商品组合	支持度
豆浆，鸡蛋	0.75
豆浆，包子	0.25
鸡蛋，包子	0.25

表 2.7 商品组合规则的置信度

商品组合规则	置信度
豆浆 → 鸡蛋	1
鸡蛋 → 豆浆	1

2.3 本章小结

本章首先介绍了点餐系统的组成部分，然后对开发系统需要用到的框架及相关技术作了简要概述，其中包括对微信小程序的相关特性及所使用到的 uni-app 开发框架的功能结构进行了介绍，并且对服务器端所用到的 SpringBoot 框架的体系结构及其相关的功能做了说明。另外，还对系统中使用到的两种数据库以及推荐系统的组成部分及其涉及到的相关算法进行了介绍。

第三章 基于协同过滤的混合推荐算法

3.1 基于物品的协同过滤算法

基于物品的协同过滤算法是协同过滤算法中的一种，另外一种是基于用户的协同过滤算法。基于用户的协同过滤算法有一些缺点：因为它是比较用户之间的相似度，所以当用户量不断增多，那么计算用户相似度所用的时间也会越来越长而且其所需要的存储空间也会不断增加。其次，它对推荐结果的解释性相比较于基于物品的协同过滤算法要差。

基于物品的协同过滤算法通过分析用户对物品的行为记录信息，从海量的商品中找出与其发生过正反馈关系的物品相似度较高的商品集合，然后根据用户对这些商品集合中的物品的评价信息去预测用户对那些没有发生过关系的商品的兴趣度或者评分。最后对那些被预测出具有较高评分或兴趣度的商品进行筛选、重组、排序之后再把它们推荐给目标用户[41]。基于物品的协同过滤算法的主要流程如图 3.1 所示。

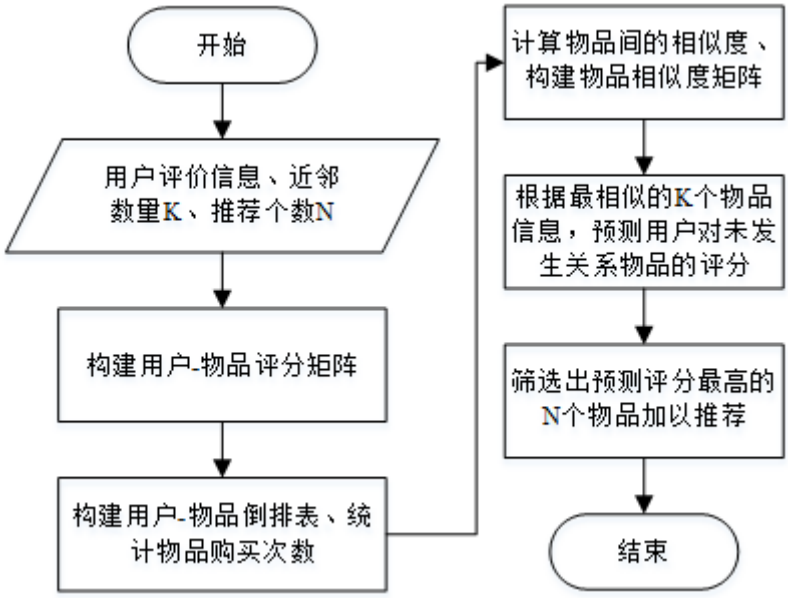

图 3.1 基于物品的协同过滤算法推荐流程

（1）构建用户-物品评分矩阵

根据用户的评价信息，构建一个矩阵，如公式 3.1 所示，内容是每个用户对所有物品的评分信息。

$$R_{m\times n}=\begin{bmatrix} r_{1,1} & \cdots & r_{1,n} \\ \vdots & \ddots & \vdots \\ r_{m,1} & \cdots & r_{m,n} \end{bmatrix} \tag{3.1}$$

矩阵$R_{m\times n}$包含了 m 个用户对 n 个物品的评分信息。$r_{i,j}$表示用户 i 对物品 j 的评分信息。

（2）计算物品之间的相似度

要想计算两个物品之间的相似度，首先需要统计出每个物品被购买的总次数以及这两个物品被同时购买的次数。 然后使用公式 3.2 来计算出这两个物品之间的相似度。这种公式计算出来的相似度叫做 Jaccard 相似度。

$$sim_{i,j} = \frac{|N(i) \cap N(j)|}{\sqrt{|N(i)||N(j)|}} \tag{3.2}$$

上式中，$|N(i)|$表示购买物品 i 的用户数，分子$|N(i) \cap N(j)|$表示的是同时购买物品 i 和物品 j 的用户数。这个式子可以理解为购买了物品 i 的用户中有多少比例的用户也购买了物品 j，同时购买这两个物品的人数越多则说明这两个物品越相似。如果某个物品很热门，则很多人会选择购买，那么有可能导致这个物品与其他物品的相似度都会很大，所以在公式 3.2 中，分母使用了物品总购买人数来做一个惩罚，用来降低该热门物品和其他物品的相似度。

上面计算物品相似度的方法直接使用的是同时购买两个物品的人数。但很有可能某个用户购买了一个商品但是其对这个商品并不满意，所以也可以利用物品的评分信息来计算两个物品间的相似度[42]，计算方法如公式 3.3 所示。

$$sim_{i,j} = \frac{R_i \cdot R_j}{||R_i||||R_j||} = \frac{\sum_k (r_{k,i} \times r_{k,j})}{\sqrt{\sum_k r_{k,i}^2} \times \sqrt{\sum_k r_{k,j}^2}} \tag{3.3}$$

上式中，$sim_{i,j}$表示用物品 i 和物品 j 的相似度，R_i表示不同用户对物品 i 的评分，$r_{k,i}$表示用户 k 对物品 i 的评分信息。

（3）计算目标用户对物品的预测评分

在计算出物品间的相似度后，就可以根据公式 3.4 去计算目标用户对于某个物品的预测评分。

$$p_{ui} = \frac{\sum_{j \in N(u) \cap S(i,K)} sim(i,j) r_{uj}}{\sum_{j \in N(u) \cap S(i,K)} sim(i,j)} \tag{3.4}$$

上式中$N(u)$表示用户 u 购买过的物品的集合，$S(i,K)$表示与物品 i 最相似的 K 个物品集合。$sim(i,j)$表示物品 i 与物品 j 之间的相似度，r_{uj}表示用户 u 对物品 j 的评分。

（4）为目标用户生成推荐列表

在计算出所有用户对其没有发生过关系的物品的预测评分后，找出目标用户对所有物品的评分以及对应的物品信息，剔除掉该用户购买过的物品，然后筛选出预测评分最高的 N 个物品，最后按照预测评分从高到低的顺序生成推荐物品列表。

基于物品的协同过滤算法适用于那些用户数量远远大于物品数量的应用场景，例如购物网站，因为物品的数据相对稳定，所以计算物品间的相似度所花费的时间要少，且不需要频

繁地去更新。基于用户的协同过滤算法更适合应用在新闻、博客等内容的推荐上，因为它们的内容更新频率非常高。基于物品的协同过滤算法的推荐结果侧重于维系用户的历史兴趣，它的推荐注重个性化，反映了用户自己的兴趣传承。而基于用户的协同过滤算法的推荐结果更侧重于反映用户所在的兴趣小组中的热点，它的推荐更加社会化。

基于协同过滤的推荐算法虽然应用得最为广泛，但是其自身还存在着一些问题。其中最主要的问题是稀疏的评分矩阵会导致预测评分的覆盖度不足[43]。例如，用户 A 对物品 b 进行预测评分时，用户 A 对物品 b 的最相似的 K 个物品都没有评分，那么用户 A 就无法对物品 b 的评分进行有效预测。所以评分矩阵的稀疏性会给物品的预测评分甚至物品的相似度计算的健壮性带来挑战。除此之外，协同过滤算法还有冷启动的问题，包括用户冷启动和物品冷启动[44]。用户冷启动是指如何给新用户做出个性化推荐，因为新用户没有历史行为记录，所以协同过滤算法不能为其找到可以推荐的物品。物品冷启动是指如何将新的物品推荐给可能对它跟兴趣的用户，协同过滤算法依然因为没有用户对它的历史行为记录而无法把它推荐给用户。

针对上述协同过滤算法中存在的问题，可以通过填充用户评分信息和在推荐系统中加入基于内容的推荐算法并结合基于流行度的推荐算法等方式加以解决。

3.2 基于关联规则的推荐算法

关联规则，在推荐领域指的是物品之间存在的一种关系，这种关系通常可以通过分析用户的行为记录而被发现，而这个发现物品间关系的过程被称为关联分析。关联分析，是一种简单、实用的数据挖掘技术，它可以发现存在于大量数据集中的关联性，从而描述出事物同时出现的规律和模式。事物间的关系可以有两种形式：频繁项集和关联规则。频繁项集是指经常出现在一起的物品的集合。关联规则就像上面说的那样，暗示着两种物品之间可能存在着某种很强的关系。基于关联规则的推荐算法，就是通过分析用户的购买或者其他行为记录，分析出物品之间隐藏的关联关系，而后利用这种关系来对目标用户进行物品推荐。该算法的基本推荐流程如图 3.2 所示。下面介绍基于关联法则推荐算法的主要流程：

（1）找出二阶频繁项集

根据用户的购物记录，先找出一阶频繁项集，即那些物品支持度大于预设最小支持度s_m的物品集合。计算单个物品支持度的公式如 3.5 所示。

$$\text{Support(i)} = \frac{N(i)}{n} \tag{3.5}$$

式中，$N(i)$表示购买物品 i 的记录条数，n表示所有的购买记录条数。所以支持度表示的

是某个事件在所有发生的事件中所占的比例。

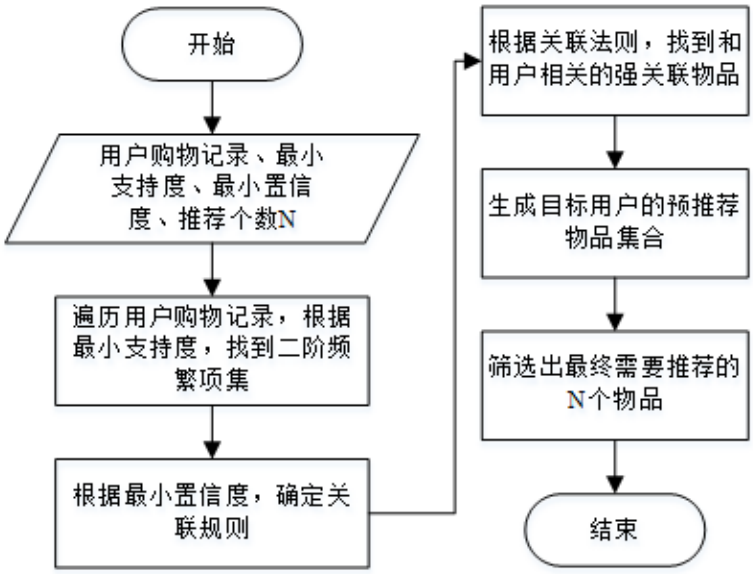

图 3.2 基于关联法则算法的推荐流程

在计算出单个物品的支持度后，将那些支持度小于预设最小支持度阈值s_m的物品剔除，而那些剩下的物品集合就是一阶频繁项集。然后在剩下的物品中找出二阶频繁项集，使用公式 3.6 去计算两两组合的物品的支持度。

$$\text{Support(i,j)} = \frac{N(i,j)}{n} \tag{3.6}$$

该公式中，$N(i,j)$表示同时购买物品 i 和 j 的记录条数，Support(i,j)表示同时购买物品 i 和 j 发生的概率。根据预设的最小支持度s_m，将小于该值的物品组合剔除，剩下的物品组合即为二阶频繁项集。

（2）确定关联规则

关联规则的强度可以用置信度来衡量。置信度的定义如公式 3.7 所示，表示的是事件 X 发生的情况下事件 Y 发生的概率。

$$\text{Confidence(X} \Rightarrow \text{Y)} = \frac{Support(X,Y)}{Support(X)} \tag{3.7}$$

$$\text{Confidence(i} \Rightarrow \text{j)} = \frac{N(i,j)}{N(i)} = \frac{Support(i,j)}{Support(i)} \tag{3.8}$$

公式 3.8 表示关联规则i ⇒ j的置信度，表示购买物品 i 还会购买物品 j 的可能性。如果该置信度大于预设的最小置信度c_m，则该规则i ⇒ j被称为是最小支持度s_m和最小置信度c_m下的关联规则。一条规则的置信度的值域为(0,1)，置信度值越大则关联规则越强。

最后，判断关联规则是否有效还需要借助提升度。提升度表示事件 X 发生对事件 Y 发生概率的提升作用，用来判断关联规则是否有实际价值，如果提升度的值大于 1 则表示该规则有效，否则表示无效。公式 3.9 为计算提升度的方法。

$$\text{Lift}(\text{X} \Rightarrow \text{Y}) = \frac{Support(X,Y)}{Support(X) \times Support(Y)} \tag{3.9}$$

（3）产生推荐列表

对于目标用户，如果一条关联规则的前见表示的物品集包含在其喜欢的或购买过的物品集合中，则该规则是可以被此用户触发的[45]。所有能被触发的规则按照置信度从高到低排序，排好序后的规则的前 N 个后见物品即是要推荐给该用户的。

基于关联法则的推荐算法在确定频繁项集的时候需要进行大量的计算，尤其是当数据量庞大的时候计算时间更长，并且当频繁项集的阶数越大计算的量就越大。

3.3 基于内容的推荐算法

基于内容的推荐算法，利用物品的描述性属性来对用户进行推荐。它尝试为用户推荐那些与其喜欢的物品相似的物品，这种相似性基于用户喜欢的对象的属性而不是基于与其他用户之间的评分相关性。不同于基于用户的协同过滤推荐算法，基于内容的推荐算法更关注目标用户自己的评分以及他喜欢的物品的属性。下面将对基于内容的推荐算法的流程进行详细介绍，其整体推荐流程如图 3.3 所示。

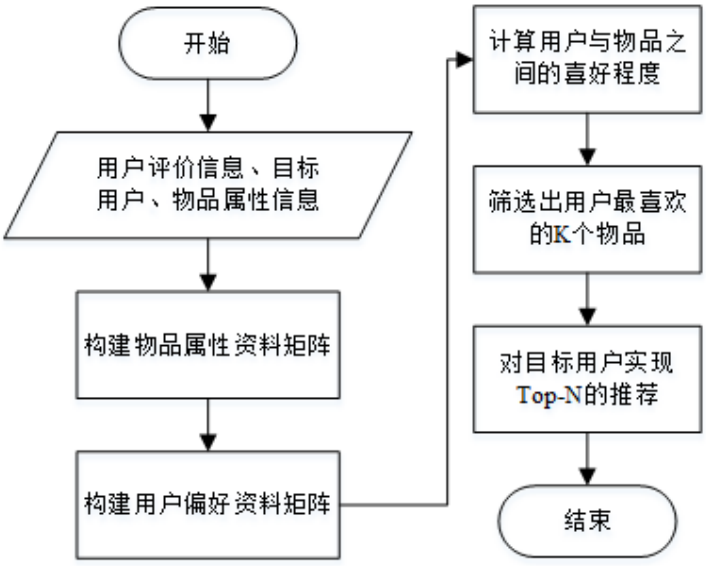

图 3.3 基于内容的推荐算法流程图

（1）构建物品属性资料矩阵

基于内容的推荐算法首先要获取物品的属性信息，并建立物品属性资料矩阵，如公式 3.10 所示。物品属性资料矩阵有 m 行 n 列，代表有 m 个物品和 n 个属性特征。矩阵的每一行表示物品拥有的属性信息，如果该物品有某个属性信息，则矩阵中的值为 1，如果没有则为 0。

$$I_{m\times n} = \begin{bmatrix} i_{1,1} & \cdots & i_{1,n} \\ \vdots & \ddots & \vdots \\ i_{m,1} & \cdots & i_{m,n} \end{bmatrix} \tag{3.10}$$

（2）构建用户偏好资料矩阵

用户偏好资料矩阵表示用户对于物品的某个属性特征的喜爱程度。矩阵如公式 3.11 所示，有 m 个用户和 n 个物品属性特征，矩阵中的值也就是用户对物品的某个属性特征的喜好程度需要借助用户评分信息来计算。

$$U_{m\times n}=\begin{bmatrix} u_{1,1} & \cdots & u_{1,n} \\ \vdots & \ddots & \vdots \\ u_{m,1} & \cdots & u_{m,n} \end{bmatrix} \tag{3.11}$$

首先，需要计算出用户对所有物品的评分的平均分，然后找到所有涉及到某个物品属性特征且是该用户评过分的电影，按照公式 3.12 计算出该用户对这个物品属性特征的偏好值。

$$p_{u,i}=\frac{\sum_{j\in S(u,i)}(x_j-Avg_u)}{n} \tag{3.12}$$

公式 3.12 中，$p_{u,i}$表示用户 u 对物品属性特征 i 的偏好程度，$S(u,i)$表示所有用户 u 评过分的且包含属性 i 的物品，x_j表示用户 u 对某个包含属性 i 物品的评分，Avg_u表示用户 u 对所有物品评分的平均分，n表示包含在$S(u,i)$中的物品数量。

（3）计算用户与物品之间的喜好程度

使用用户偏好资料与物品属性资料信息可以计算出用户对某一个物品的喜好程度。计算方法如公式 3.13 所示，利用余弦相似度来衡量一个用户对某个物品的喜好程度。

$$\cos(U,\mathrm{I})=\frac{\sum_j u_j i_j}{\sqrt{\sum_j u_j^2}\sqrt{\sum_j i_j^2}} \tag{3.13}$$

在上面的公式中，u_j表示用户 u 对属性特征 j 的偏好程度，i_j表示物品 i 的属性特征 j 在物品属性资料矩阵中对应的值。

（4）完成推荐

在计算出某个用户对所有物品的喜好度后，筛选出喜好度最高的几个物品推荐给用户。

基于内容的推荐算法倾向于推荐用户迄今为止见过的类似的物品，推荐的结果缺乏一定的新颖性和偶然性。在冷启动问题上，此算法有助于解决新物品的推荐问题，但是无法解决用户冷启动问题，因为新用户没有历史行为记录，所以无法确定新用户对哪些物品感兴趣。

3.4 混合推荐算法

以上几小节中主要介绍了三个基本的推荐算法的工作流程及它们各自的优缺点。所以针对以上算法的特点，本文设计出一个基于协同过滤的混合推荐算法，主要流程如图 3.4 所示。下面将对该混合推荐算法的具体工作流程做出详细介绍。

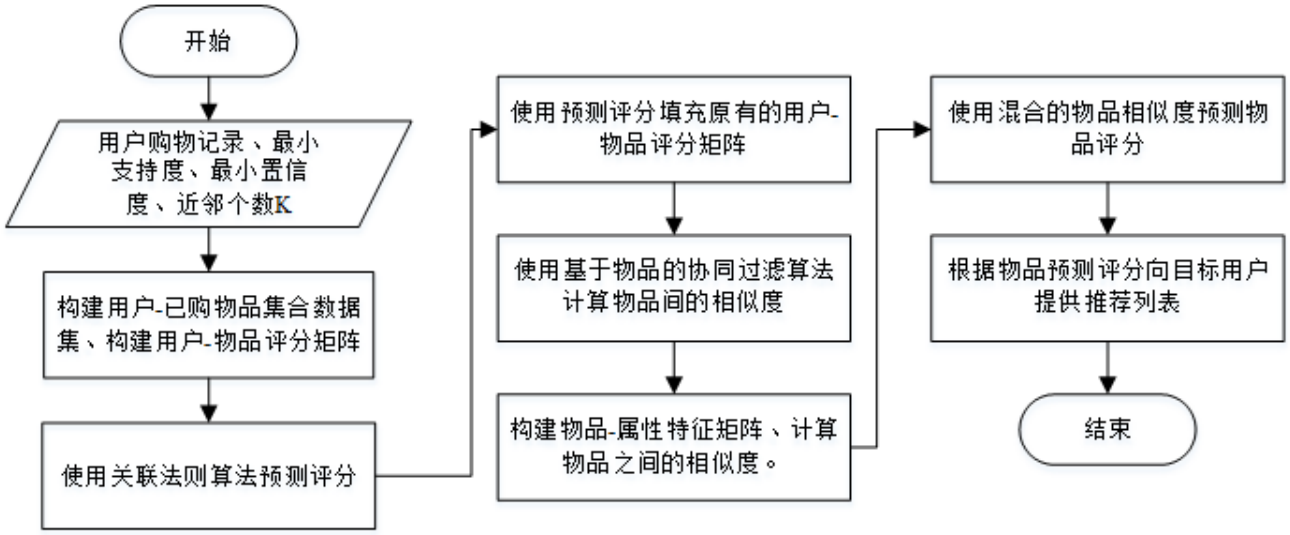

图 3.4 混合推荐算法主要流程

步骤一：准备数据。

获取用户购物记录信息，构建用户购买物品记录数据集，该数据集每行数据为用户发生正反馈的物品列表：[ItemID1，ItemID2，……，ItemIDn]。同时根据用户购物记录构建用户-物品评分矩阵 R，如式 3-1 所示。给出计算物品相似度时的最近邻个数 K、最小支持度阈值s_e和最小置信度阈值c_e。

步骤二：使用关联法则预测物品评分。

通过 Apriori 关联规则算法找出满足最小支持度s_e且最小置信度为c_e的二阶频繁项集关联法则。Apriori 关联规则算法主要包含两个步骤：首先找出数据集中所有的二阶频繁项集，这些项集出现的频繁度要大于等于最小支持度，然后根据这些二阶频繁项集找到满足最小置信度的关联规则。

遍历历史记录数据集，首先寻找一阶频繁项集，然后在满足最小支持度s_e的一阶频繁项集中寻找二阶频繁项集，并根据满足最小支持度s_e的二阶频繁项集找出置信度大于等于最小置信度c_e的关联规则。最后找到的关联法则是由 (ItemIDx, ItemIDy, Confidence)构成的数据集，ItemIDx 表示物品 x 的 ID，Confidence 表示此组关联规则的置信度。

步骤三：填充用户-物品评分矩阵。

使用步骤二中的关联法则数据集预测评分。遍历用户-物品评分矩阵中的每一行数据，对于每个用户，如果关联法则数据集中有其发生过正反馈的物品且该物品所对应的关联物品没有和用户发生过正反馈的话，则预测此用户对这个关联物品的评分。使用公式 3.来预测用户对物品的评分。

$$P_{ui} = \frac{\sum_{j \in S(u)} conf_{ji} \times r_{uj}}{\sum_{j \in S(u)} conf_{ji}} \quad (3.14)$$

P_{ui} 为用户 u 对物品 i 的预测评分，$S(u)$表示用户 u 已经评分的物品集合，$conf_{ji}$ 表示物

品 j 对物品 i 的置信度，r_{uj} 表示用户 u 对物品 j 的评分。使用预测的评分填充用户-物品评分矩阵。对每个用户还未评分的物品使用上面的预测评分来作为该用户对此物品的评分。

步骤四：使用新的用户-物品评分矩阵计算物品间的相似度。

使用填充后的用户-物品评分矩阵计算物品的相似度。使用 3.1 小节介绍的公式 3.3 来计算物品间的相似度。为每个物品计算其与其他物品的相似度，最终获得物品相似度矩阵 S：

$$\mathrm{S}=\begin{bmatrix} s_{11} & \cdots & s_{1n} \\ \vdots & \ddots & \vdots \\ s_{n1} & \cdots & s_{nn} \end{bmatrix} \tag{3.15}$$

$s_{ij}(1 \le i,j \le n)$表示物品 i 与物品 j 的相似度。

步骤五：使用基于内容的推荐算法计算物品相似度。

对每个物品进行分析，构建物品-内容属性矩阵 W：

$$\mathrm{W}=\begin{bmatrix} w_{11} & \cdots & w_{1l} \\ \vdots & \ddots & \vdots \\ w_{n1} & \cdots & w_{nl} \end{bmatrix} \tag{3.16}$$

n 表示物品的数量，l 表示物品属性的数量。$w_{ij}(1 \le i \le n, 1 \le j \le l)$表示物品 i 的第 j 个属性，如果物品 i 有此属性，则$w_{ij}=1$，否则$w_{ij}=0$。

通过比较每个物品的属性来获得物品之间的相似度。相似度计算公式为：

$$sim_c(\mathrm{i,j})=\frac{V(i)\cdot V(j)}{\sqrt{\|V(i)\|\|V(j)\|}} \tag{3.17}$$

$sim_c(\mathrm{i,j})$表示物品 i 和物品 j 通过内容属性计算得到的相似度，值域为[0,1]。$V(i)$表示物品 i 的属性向量。为每个物品计算其与其他物品的相似度，最终获得物品相似度矩阵S_c：

$$S_c=\begin{bmatrix} s_{c11} & \cdots & s_{c1n} \\ \vdots & \ddots & \vdots \\ s_{cn1} & \cdots & s_{cnn} \end{bmatrix} \tag{3.18}$$

$s_{cij}(1 \le i,j \le n)$表示物品 i 与物品 j 基于内容属性计算得到的相似度。

步骤六：使用混合物品相似度预测物品评分。

结合协同过滤算法计算出的相似度$\mathrm{sim(i,j)}$和基于内容属性计算出的相似度$sim_c(\mathrm{i,j})$得到改进后的物品相似度，公式为：

$$sim_{new}(\mathrm{i,j})=\alpha \mathrm{sim(i,j)}+(1-\alpha)sim_c(\mathrm{i,j}) \tag{3.19}$$

上式中α表示权值，根据不同的权值可以得到不同的物品相似度$sim_{new}(\mathrm{i,j})$。

使用改进后的物品相似度对用户 u 的目标物品 i 的进行预测评分，预测评分公式为：

$$P_{ui}=\frac{\sum_{j\in S(i,K)} sim_{new}(i,j)r_{uj}}{\sum_{j\in S(i,K)} sim_{new}(i,j)} \tag{3.20}$$

P_{ui}表示用户 u 对物品 i 的预测评分。$S(i,K)$是与物品 i 最相似的 K 个物品集合。r_{uj}表示用户 u 对物品 j 的真实评分。

步骤七：生成推荐列表。

对于目标用户，遍历所有物品集，挑选出还未发生过正反馈的物品集合，并按照物品的预测评分从大到小排序，生成待推荐列表。最后选取待推荐列表中前 N 个物品生成推荐列表。

3.5 测试与分析

本小节使用由 GroupLens 研究组提供的 MovieLens 数据集来对本文设计的混合推荐算法进行实验，测试并比较其与基于物品的协同过滤算法的性能。本次实验使用的 MovieLens 数据集包含 600 个用户对 9000 部电影的 100000 条评分记录以及包含电影分类属性的 9000 部电影数据。用户评分取值范围为 1-5，评分越高说明用户对此电影越满意，每位用户的评分信息不少于 20 条。

实验中，首先需要对用户的评分记录数据进行处理，并将其分成训练数据集与测试数据集两个部分，训练数据集与测试数据集数据量的比值为 7:1。使用训练数据集构建用户-物品评分矩阵，通过计算可以求得该矩阵的稀疏度。实验前先设置算法中涉及到的初始值，最小支持度阈值s_e为 0.2，最小置信度阈值c_e为 0.6，计算混合相似度时的权值α为 0.9。推荐算法效果的衡量指标多种多样，本次实验采用计算物品预测评分与实际评分的误差值来判断推荐质量的好坏，计算公式为 3.21。

$$MAE_u = \frac{\sum_{i=1}^{N_u}|P_{ui}-r_{ui}|}{N_u}, \mathrm{MAE}_a = \frac{\sum_{u=1}^{M} MAE_u}{M} \tag{3.21}$$

MAE 表示平均绝对误差，该值越小则表明算法对物品的预测评分越准确，即算法的推荐质量越好[46]。上式中，MAE_u 表示用户 u 对N_u 个物品预测评分的 MAE，N_u为要预测评分的物品数量，P_{ui}为用户 u 对物品 i 的预测评分，r_{ui} 为用户 u 对物品 i 的实际评分。MAE_a表示全体用户预测评分的 MAE，M 为所有用户的数量。

本次实验，将主要比较基于物品的协同过滤算法和 3.4 小节介绍的混合推荐算法的推荐效果，实验结果如图 3.5 所示。图中纵坐标是预测评分的 MAE 值，横坐标是计算物品相似度时的最近邻个数 K，从 5 开始增加到 30，增加间隔为 5。图中标记为 itemCF 的曲线表示基于物品协同过滤算法预测评分的 MAE 值。标记为 itemCF-content 的曲线表示的是基于物品协同过滤与物品内容的混合推荐算法对物品预测评分的 MAE 值，该种混合方式是将基于物品协同过滤计算出的物品相似度与基于内容算法计算出的物品相似度进行线性加权叠加，并使用叠加后的新的物品相似度去计算物品预测评分。曲线 rule-itemCF 表示使用基于关联法则的算

法预测物品评分填充到用户-物品评分矩阵，然后使用新的矩阵结合基于物品的协同过滤算法预测物品评分的 MAE 值。曲线 rule-itemCF-content 表示的是使用 3.4 小节介绍的混合推荐算法计算出的物品预测评分的 MAE 值。

从实验结果图中，可以看出，3.4 节介绍的混合推荐算法的推荐效果是最好的。观察最近邻个数 K 对 MAE 值的影响可以发现，随着最近邻个数 K 的缓慢增长，MAE 值慢慢减小，并且在 K 到达 10 的时候 MAE 值最小。在 K 的个数大于 10 的情况下，随着 K 的个数的增加，MAE 值在不断变大。由此可以发现最近邻个数为 10 左右时，物品预测评分的 MAE 值最小，这表示在计算物品相似度时，选取最相似的 10 个左右的物品可以让推荐的质量达到最好。

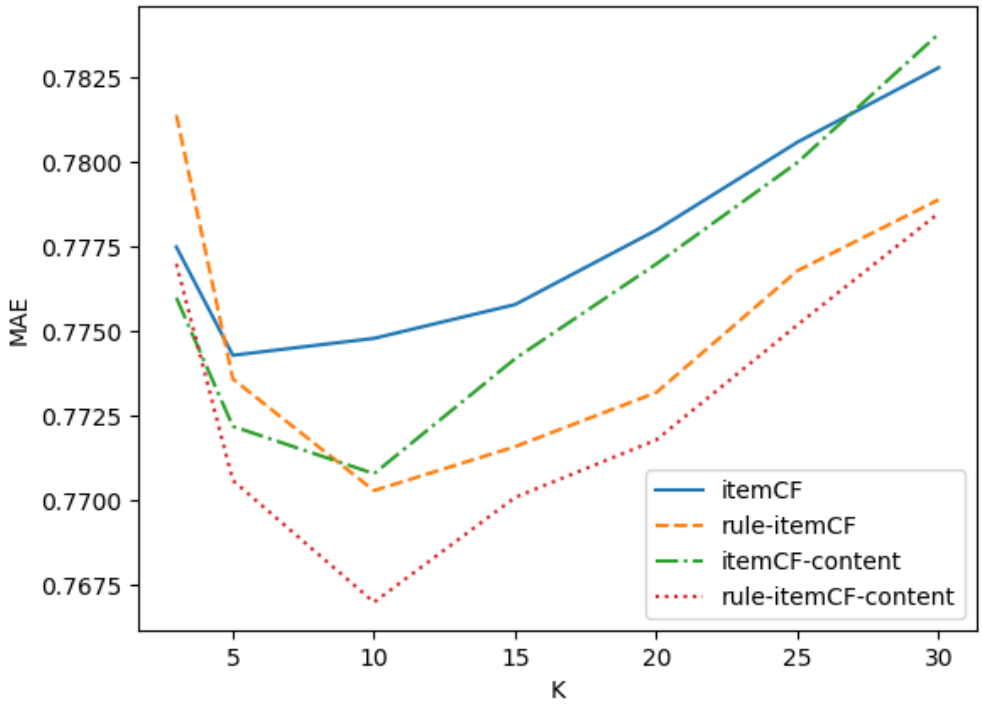

图 3.5 不同推荐算法预测评分的 MAE 对比

3.6 本章小结

本章首先介绍了基于物品的协同过滤、基于内容、基于关联法则这三种基本推荐算法的主要工作流程以及它们存在的问题。从而引出包含以上三种算法思想的一种混合推荐算法。该算法使用基于关联法则的算法预测用户对物品的评分并将它们填充到初识的用户-评分矩阵中，降低了矩阵的稀疏度，然后结合基于内容算法与基于物品的协同过滤算法得到的物品相似度去预测物品评分实现物品推荐。最后测试结果表明，本章给出的混合推荐算法能够达到更好的推荐效果。

第四章 系统设计与实现

4.1 系统架构设计

整个系统以点餐功能作为基础进行设计，且为了方便消费者点餐，需要设计一个移动端的点餐应用。消费者点餐过程中浏览到的菜品信息是由商家给出的，所以需要给商家设计一个餐饮管理应用，为商家提供菜品信息管理等功能。另外，消费者与商家对各自应用的操作请求需要通过服务器来处理，同时在操作过程中产生的数据需要存储到数据库中，方便日后进行查看与分析。在点餐的过程中，为了让消费者能够快速地找到自己喜欢的菜品，需要在系统中加入一个推荐功能模块来为消费者提供个性化的推荐服务。

经过上面的分析，本文的智能推荐点餐系统主要由四个部分组成：移动端点餐功能模块、网络端餐饮管理功能模块、服务器端模块以及推荐功能模块。系统的整体架构如图 4.1 所示。

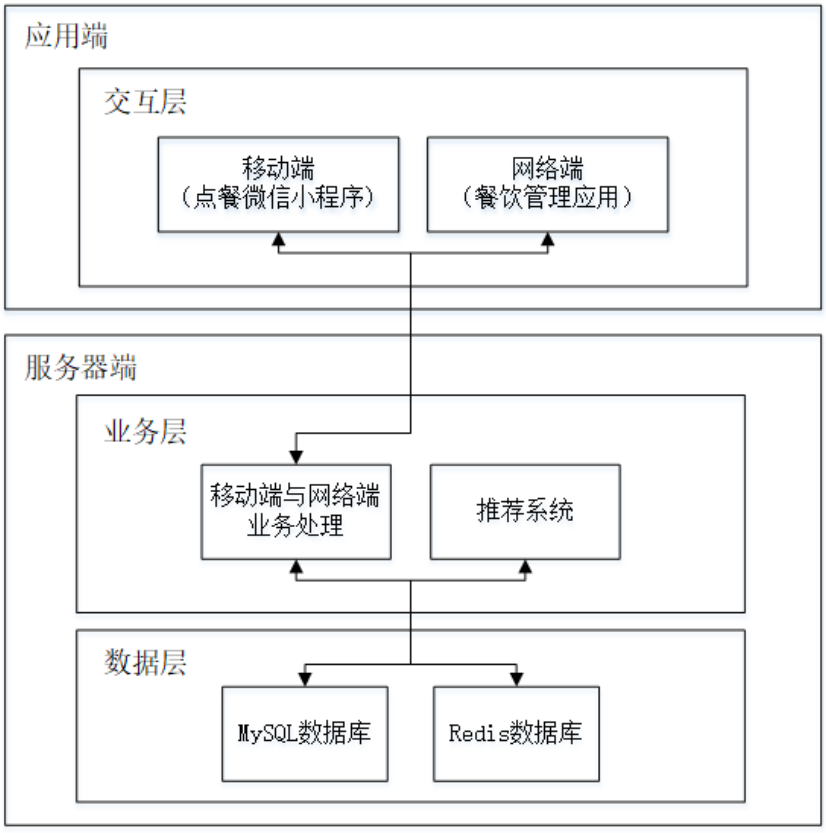

图 4.1 系统整体架构

本文的智能推荐点餐系统是基于 B/S 模式设计的，总体分为三层。交互层构建在应用端，主要负责显示用户操作界面、响应用户的操作。服务器端包含业务层和数据层，业务层主要负责处理交互层中用户传来的操作命令信息并接收传来的各种数据，根据不同种类的操作命

令及数据执行不同的业务流程，并将需要的信息返回给交互层对界面进行刷新。有的业务需要去数据层对数据库中的数据做出增删改查等操作。应用端与服务器端通过 HTTPS 协议进行通信，HTTPS 协议由于在 HTTP 协议的基础上对传输进行加密并加入身份认证机制，从而使得数据的传输更加安全。

4.2 功能模块设计

4.2.1 移动端功能模块设计

移动端功能模块即微信点餐小程序。通常消费者在餐厅内的点餐流程如图 4.2 所示。根据点餐流程去设计微信点餐小程序的主体功能，可以将其分为四个部分：菜品显示、菜品选择、订单信息显示、付款与评价。

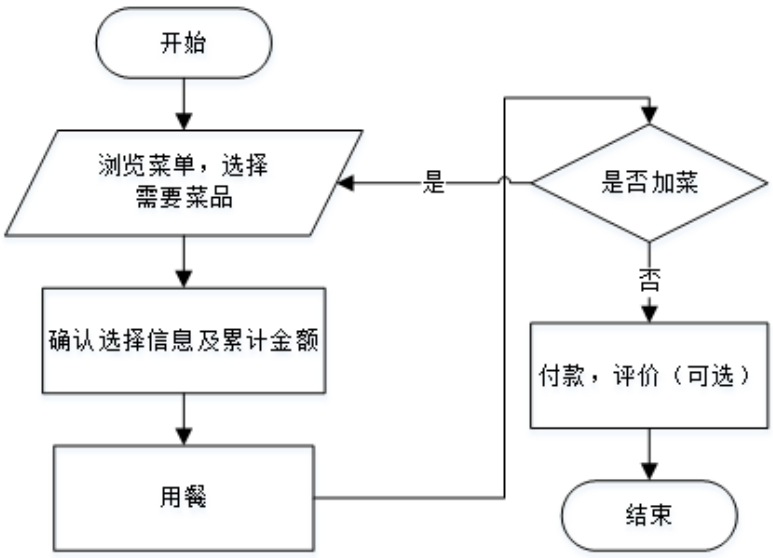

图 4.2 点餐流程

消费者在使用微信点餐小程序时，不需要像使用其他 APP 那样先去输入用户的账号和密码进行登录操作。由于微信小程序是运行在微信 APP 之中，所以只需用户授权便可使用。在用户授权后，小程序可以自动获取到用户的登录信息。

在用户进入到小程序时，通过向服务器端发起菜品信息查询请求并获取返回结果后，小程序将向用户展示出所有的菜品信息，包括菜品名称、图片及价格等。所有菜品信息通过分类以列表的形式展示出来，用户可以根据不同的分类查看相对应的菜品信息，这种方式有助于用户快速找到自己想要类型的菜品。同时，推荐菜品也可以作为一种特殊的类型，用户可以通过点击该类型标签查看。另外在用户点击该特殊分类查看推荐菜品时，可以通过用户选择的用餐人数给其推荐合适数目的菜品。

用户浏览菜单时，通过点击菜品信息栏上的添加按钮可以将该菜品加入购物车，也可以点击删除按钮将该菜品从购物车中去除。同时购物车中可以列出用户选择的所有菜品名称及其价格信息，并且用户可以直接在购物车中删除菜品或增加菜品数量。

选择完需要的菜品后，用户点击确认按钮进入到确认订单界面，此界面为用户展示所有已选择的菜品相关信息并给出总金额，方便用户确认是否多点或少点某个菜品。确认后用户可以直接下单。

用户在用完餐后，可以直接在小程序中进行付款并在付款后对每个菜品进行评价。

4.2.2 网络端功能模块设计

网络端即餐饮管理应用，主要是帮助商家对一些与餐饮相关的信息进行管理，例如对菜品信息的增加、删除、修改。其主要功能如图 4.3 所示。

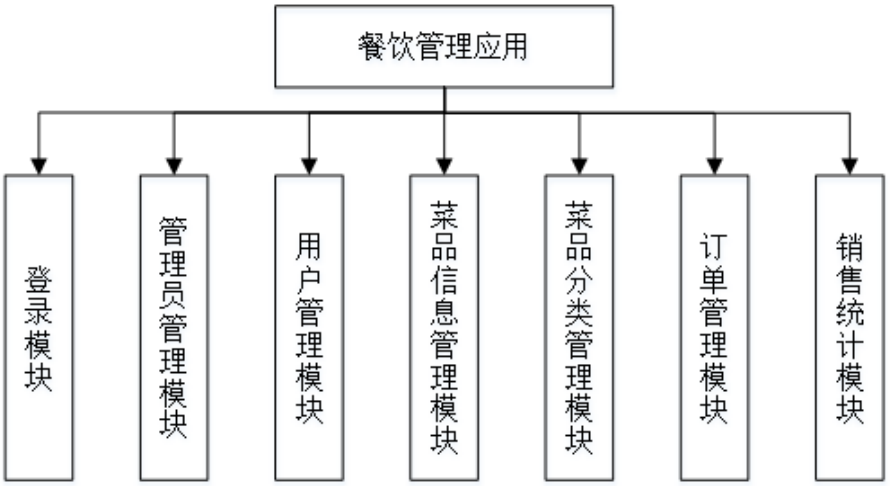

图 4.3 餐饮管理应用主要功能

管理员管理模块，主要对能够登录餐饮管理应用的工作人员的帐户信息进行管理，包括帐户的增加、删除、修改、查询等功能。用户管理模块，主要是对使用过微信点餐小程序的用户信息进行管理。菜品信息管理与菜品分类管理模块，能够帮助商家管理菜品的分类信息以及菜品相关属性的详细信息，包括菜品的名称、图片、价格、库存、类别、销量、评分等。订单管理模块可以让商家查看到每个用户的购买记录。销售统计模块，通过分析用户的购物记录，可以按照一定的时间间隔统计出累计销量、累计金额以及最受欢迎的菜品信息，让商家对餐厅的运营情况有一个直观的了解。

4.2.3 推荐系统功能模块设计

推荐系统通过采集消费者历史行为记录信息并结合推荐算法对其进行分析挖掘以实现为

消费者提供菜品推荐的功能。本文中的推荐系统为离线推荐系统，该系统会在一个指定的时间点开始工作，对用户数据进行分析然后产生推荐结果，一般用时较长。其整体流程如图 4.4 所示。观察离线推荐系统的工作流程，可以将其分为三个功能模块：数据采集模块、数据分析模块、物品推荐模块。下面将对这三个模块分别进行介绍。

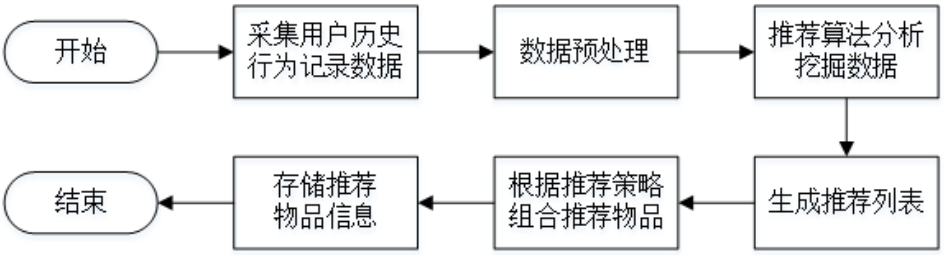

图 4.4 离线推荐系统工作流程

（1）数据采集模块

该模块主要功能是采集用户历史行为记录信息，例如用户的购物记录信息、用户的评价信息等。同时还将采集其他需要的重要信息，例如用户购买的菜品信息。用户的订单信息和用户评价信息都存放在 MySQL 数据库中，所以该模块需要编写程序访问数据库获取这些必要数据以为之后的数据分析做准备。在获取这些数据后，还需要进行数据的预处理工作，将一些异常数据或之后数据分析中用不到的信息剔除然后建立用户-菜品评分矩阵以及菜品信息数据集。

（2）数据分析模块

该模块主要是对上个模块输出的数据集进行分析，使用本文中给出的混合推荐算法预测不同用户对各个菜品的评分，然后根据这些评分的高低按一定顺序为各个用户生成推荐列表。大体步骤为：先使用基于关联规则的算法预测用户对菜品的评分，然后使用这些预测评分填充用户-菜品评分矩阵，降低原有矩阵的数据稀疏度，然后使用基于物品协同过滤的算法和基于内容的算法分别计算出物品间的相似度，并对这两个相似度进行线性加权得到新的相似度，最后使用新的相似度预测用户对物品的评分。

（3）物品推荐模块

该模块使用物品的预测评分按照从高到低的顺序为每个用户生成菜品推荐列表，如果某个菜品已经被用户购买过则不再进行推荐。另外，该模块还需遍历菜品信息数据集，统计出评分最高的前 8 个分类分别为素菜和非素菜的菜品，生成候补推荐列表。最后，将这些生成的推荐菜品信息存储到 MySQL 数据库中。

4.2.4 服务器端功能模块设计

服务器端的主要功能是通过 HTTPS 协议与应用端进行数据交互，为消费者和商家使用的应用程序提供服务，并且使用推荐系统作用于数据库中的历史记录来生成推荐物品，实现对消费者的个性化推荐服务。服务器端主要分为两层，本小节主要介绍业务层。服务器端的业务层主要分为两个模块：移动端与网络端业务处理模块、推荐系统模块。本小节主要介绍第一个模块。

移动端与网络端业务处理模块，是本系统的核心功能模块，主要负责处理移动端或网络端应用发出的业务请求，其在后台通过各种不同的业务方法实现各种核心功能，包括为移动端提供菜品信息、接收并存储用户订单以及为网络端应用提供菜品信息管理、销售情况统计等重要功能。同时该模块还需要与数据库进行交互，根据相关的业务对数据库执行增删改查等操作。该模块中的一个特色业务功能是为指定用户生成菜品推荐列表。其主要流程如图 4.5 所示。

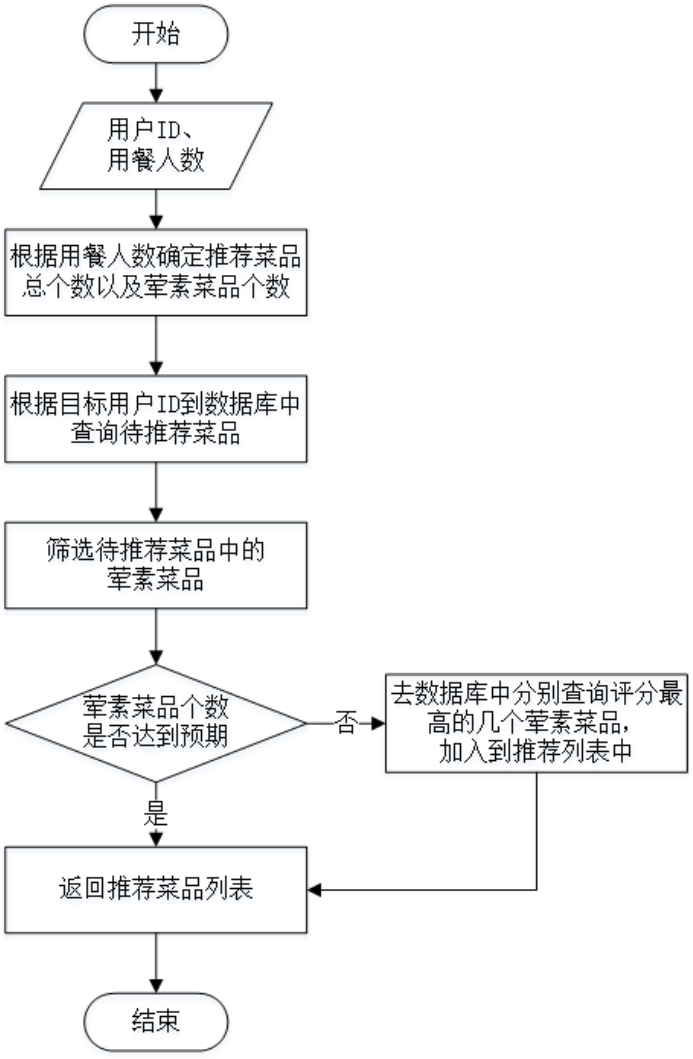

图 4.5 推荐策略流程

首先需要获取用户的 ID 和用餐人数，并通过用餐人数确定推荐菜品总个数。之后，根据推荐菜品总个数确定素菜个数和非素菜个数，原则上，素菜个数不少于总个数的三分之一。确定完菜品个数后，先去数据库查询推荐系统给此用户生成的所有推荐菜品，然后根据荤素类别筛选菜品，并统计相应个数。如果素菜或非素菜的个数小于之前确定的数目，就去数据库中查询最受欢迎的素菜或非素菜加入到推荐菜品列表中。最后将这些菜品发送到应用端的微信小程序上展示给用户。

该功能模块的设计基于 B/S 架构形式，分为表现层、业务层和持久层。表现层，负责接收客户端的请求并向客户端发送响应结果，通常客户端使用 HTTP 协议与服务端进行通信。表现层包括展示层和控制层，展示层负责响应结果的展示，控制层负责接收客户端的请求。表现层依赖于业务层，接收到客户端的请求一般会传递给业务层进行处理，然后获取处理结果。业务层，主要负责业务逻辑处理，可能会依赖于持久层。持久层，负责数据的持久化即与数据库打交道，对数据库进行增删改查操作。

4.3 数据库结构设计

数据库中需要存储用户、菜品等基本信息以及用户的历史行为信息，这些信息将会被展示在应用端的用户界面上并且会被用于进行数据分析。数据库中实体表如下：

（1）管理员信息表，包含管理员编号、姓名、密码、创建和修改时间。

（2）菜品信息表，包含菜品编号、名称、价格、库存、图片、是否热门、描述、是否下架、烹饪类别、荤素类别、食材信息、销量、评分、创建及修改时间。

（3）烹饪分类表，包含烹饪方式编号、名称、创建及修改时间。

（4）荤素分类表，包含分类编号、荤素种类命名、创建及修改时间。

（5）食材信息表，包括食材编号、名称、创建及修改时间。

（6）订单信息表，包括订单编号、订单总金额、订单支付状态、用餐桌号、用餐人数、下单用户编号、创建及修改时间。

（7）订单详情信息表，包括订单详情编号、菜品编号、所属订单编号、所购菜品份数、菜品单价、菜品评分、创建及修改时间。

（8）用户信息表，包含用户微信 ID、用户微信昵称、用户所在城市、用户性别、创建及修改时间。

（9）推荐菜品信息表，包含用户微信 ID、菜品编号、创建及修改时间。

4.4 系统实现

4.4.1 微信点餐小程序实现

微信点餐小程序采用基于 Vue.js 和微信小程序 API 的 uni-app 前端框架进行开发实现。其主要功能包括菜单分类展示、菜品推荐、点餐下单、确认订单、付款评价，这些功能被分布在四个用户页面中完成。

（1）菜单显示页面

进入微信小程序后，首先进入的是菜单显示页面，如图 4.6 所示。在该页面加载之前，小程序的 onLaunch() 函数会为用户实现自动登录，不需要用户执行注册或登录操作。进入页面后，通过 onLoad() 函数与服务器端通信获取所有菜品的信息，并将其以分类的形式展现给用户，用户通过点击左侧类别按钮，可以在右侧查看相应分类的菜品信息，包括菜品图片、名称与价格。如果用户需要选择菜品，直接点击右侧菜品信息栏中的加号，此时会显示出购买该菜品的份数同时将该菜品加入到购物车中。点击页面左下角的购物车图标可以进入购物车，购物车中包含所有用户选择的菜品，用户可以在购物车中增减菜品个数或者清除所有菜品，如图 4.67 所示。

图 4.6 菜单页面

图 4.7 购物车

点击左侧分类栏上的“为您点餐”，即可为用户进行菜品推荐。点击该分类后，用户需要先选择用餐人数，如图 4.8 所示。确认后小程序将用户微信编号与用餐人数发送给服务器端，服务器端根据这些信息获取推荐菜品并将其返回给小程序并在右侧菜品信息栏中显示出来，如图 4.89 所示。

图 4.8 用餐人数选择

图 4.9 推荐菜品展示

（2）订单确认页面

用户选择完菜品后点击“选好了”按钮，小程序将跳转到确认订单页面，如图 4.10 所示。该页面把用户选择的所有菜品信息罗列出来供用户确认，并请用户选择用餐人数。如果用户有其他要求，可以填写备注信息。点击确认下单后，用户的订单信息将会被传输到服务器端并存储到数据库中。

（3）账单详情页面

确认下单后，小程序将跳转到账单详情页面，如图 4.11 所示。该页面为用户展示出订单详情以及需要支付的金额。如果用户还需加菜，则可以点击“加菜”按钮，此时页面将跳转到菜单页面。如果用户没有加菜需求，用完餐后可以直接点击“买单”按钮完成在线支付。

（4）用餐评价页面

支付完后，小程序跳转到评价页面，如图 4.12 所示，用户可以对本次消费的所有菜品进行评价，点击“完成”按钮即可提交评价信息。小程序将评价信息发送到服务器端，服务器

端操作数据库将评价信息加入到用户的订单详情中。

图 4.10 确认订单页面图

4.11 账单详情页面

图 4.12 评价页面

4.4.2 网络端应用实现

网络端，即餐饮管理应用，使用 X-admin 前端框架进行开发。主要功能有管理员与用户信息管理、菜品及其分类信息管理、订单信息管理以及销售情况统计。下面将对这几种功能的实现做出介绍。

（1）管理员与用户信息管理

管理员是指可以登录餐饮管理应用的工作人员，其可以进行查看用户订单信息，维护菜品信息等操作。用户信息管理界面如图 4.13 所示。界面中使用表格的形式分页列出了包括用户昵称、性别在内的一些用户信息。管理员可以通过选择日期或者输入用户昵称来查询符合条件的用户信息，也可以对某些用户信息进行删除操作。

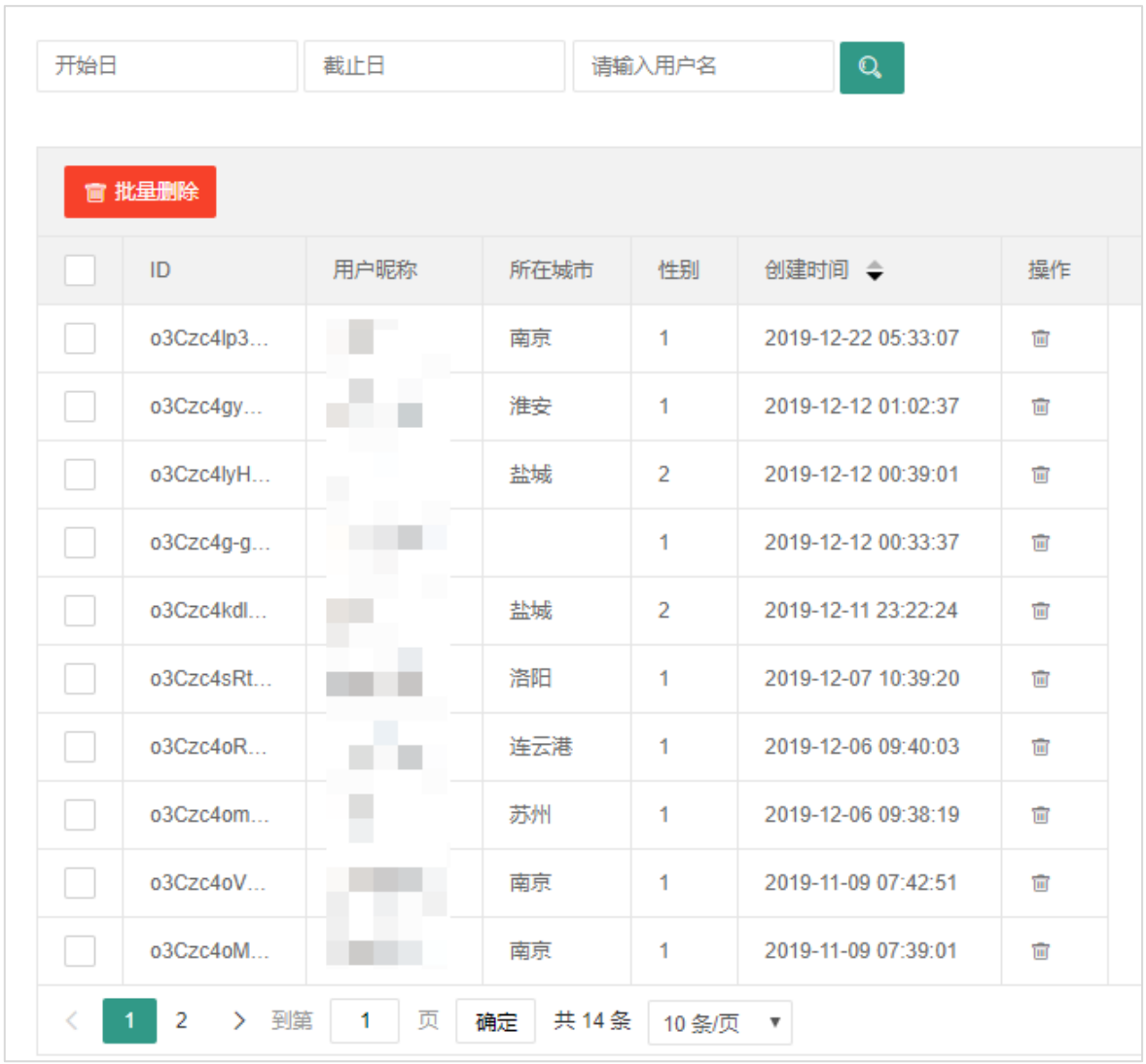

图 4.13 用户信息管理界面

（2）菜品及其分类信息管理

通过与服务器端交互查询出数据库中菜品信息，然后在菜品管理界面中（图 4.14 所示）

以表格的形式分页展示菜品数据，可以直观地看到菜品名称、图片、菜价等重要信息。表格的最后一列包含了三个操作按钮，分别执行菜品上下架、菜品信息修改、菜品信息删除功能。该界面也包含按照条件查询的功能，另外通过点击表格上方的添加按钮，管理员可以新增一条菜品信息，如图 4.15 所示。管理员可以在输入框中填入菜品的名称、价格等信息，菜品的类别通过下拉框来选择，菜品的属性也即食材信息通过复选框进行选择。其中，标注红色星号的项目为必填项，如果没有填写或填写格式错误则会弹出提示窗告知管理员需要按照要求来填写相关信息。

图 4.14 菜品管理界面

（3）订单信息管理

订单信息管理界面和菜品信息管理界面类似，都是以表格的形式分页展示信息并且可以按照相应条件查询对应的订单信息，但是此界面没有新增和修改功能，因为新增和修改操作是通过小程序完成的。该界面在最后的操作栏中包含改变支付状态、查看订单详细信息以及删除此条订单记录的功能，如图 4.16 所示。

（4）销售情况统计

该功能主要为商家提供餐厅销售情况的统计，其中包括销量、销售额以及最受欢迎菜品的统计，统计周期为每日、每周、每月。统计信息会以图表的形式直观地显示出来，商家通过查看这些统计信息，能够及时掌握餐厅的运营情况，然后根据当前的运营情况调整自己的运营策略。例如，通过查看最受欢迎的菜品统计信息，调整自己的菜品推荐策略，为新用户推荐那些比较受欢迎的菜品。如图 4.17 所示。

图 4.15 新增菜品界面

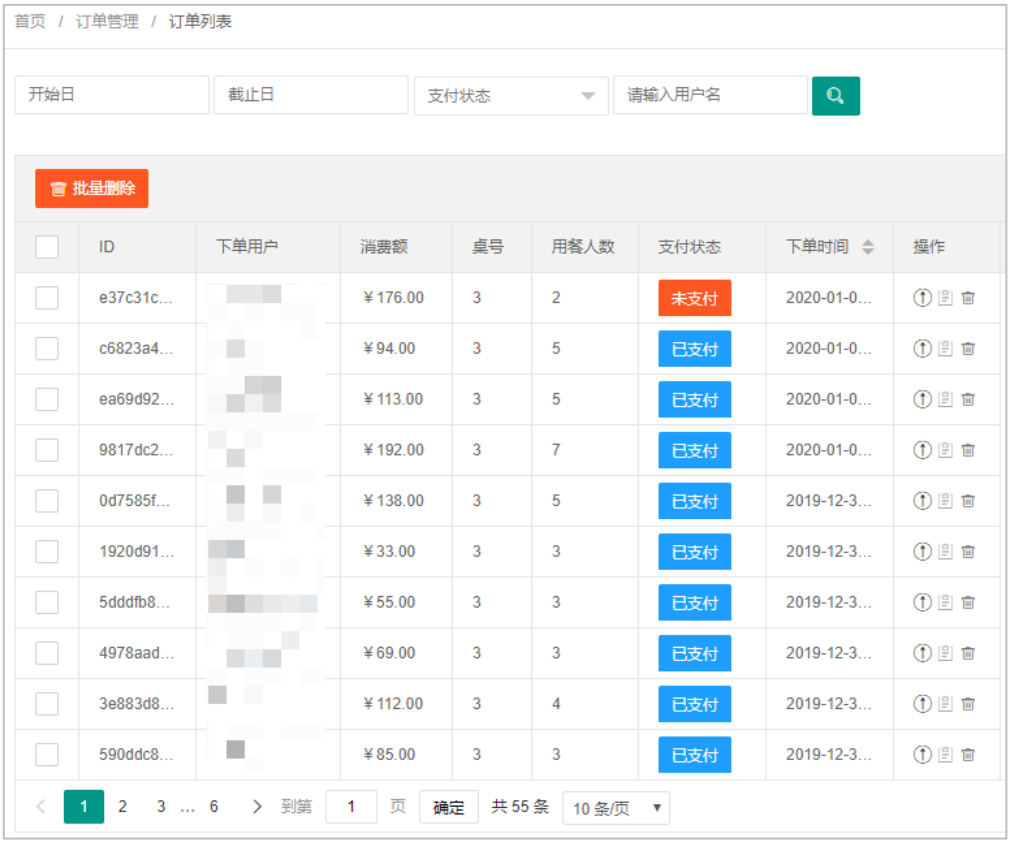

图 4.16 订单管理界面

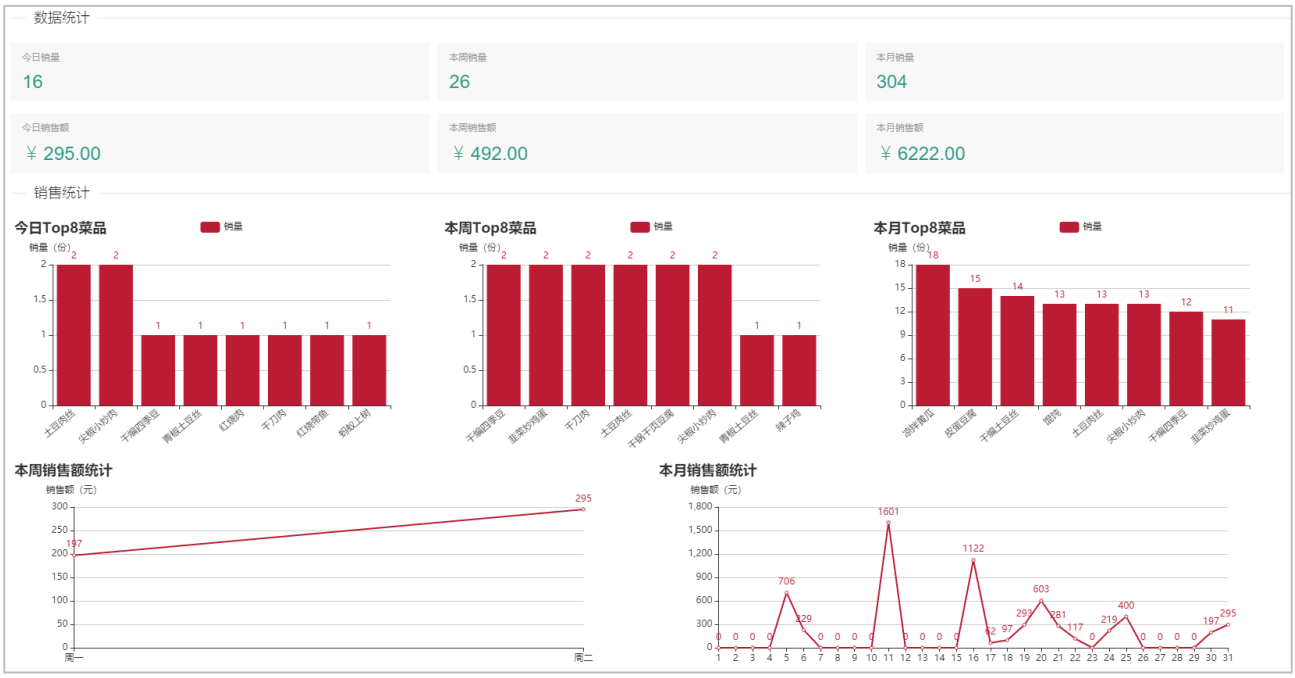

图 4.17 销售情况统计界面

4.4.3 推荐系统实现

推荐系统模块是整个系统的重要模块，是系统为用户提供推荐服务的核心。该模块使用 Python 语言进行开发[47]，并借助 MySQL 数据库存储推荐信息。其主要包括用户数据采集与预处理、推荐算法分析用户数据以及推荐信息生成三个部分。

首先，推荐系统模块访问数据库获取用户订单详细信息，以及菜品的相关信息。所有的用户订单详细信息和菜品信息分别被存放在一个列表 List 中，列表中的每一个元素都是用户订单信息或菜品信息组成的一个元组 Tuple，其格式如表 4.1 和表 4.2 所示

表 4.1 用户订单信息格式

名称	描述
userId	用户的编号
itemId	菜品的编号
score	用户对菜品的评分

表 4.2 菜品的信息格式

名称	描述
itemId	菜品编号
itemScore	菜品的平均评分
itemMaterials	菜品的属性（烹饪方式、荤素类别、食材）

在获取上述的数据后，需要对其进行预处理并构建用户-菜品评分矩阵。用户-菜品评分矩阵的创建在程序中使用字典（dict）这种数据类型来完成，dict 是一种以键值对（key-value）的方式存储元素的数据类型。每个用户编号（userId）对应键值对里面的 key，而菜品编号

（itemId）与用户对菜品评分（score）作为 value，与 userId 相对应。在 itemId 和 score 组成的 value 中，itemId 作为 key，score 作为它对应的 value。这样就形成了一个嵌套的 dict，其具体结构如图 4.18 所示。

```
{
  userId-1:
        {
            itemId-1: score1,
            itemId-2: score2,
            ......
        },
  userId-2:
        {
            itemId-1: score1,
            itemId-2: score2,
            ......
        },
  ......
}
```

图 4.18 用户-物品评分矩阵数据结构

数据预处理结束后，首先要使用基于关联法则的算法预测用户对物品的评分，并将这些预测评分填充到用户评分数据集中。然后使用基于协同过滤的算法结合用户-菜品评分矩阵计算物品间的相似度sim_i，其核心代码如图 4.19 所示。

```
def __item_similarity_cos(self):
    C = dict()
    N = dict()
    for u, items in self.train.items():
        for item in items.keys():
            if item not in N.keys():
                N[item] = 0
            N[item] += items[item] * items[item]
            for j in items.keys():
                if item == j:
                    continue
                if item not in C.keys():
                    C[item] = dict()
                if j not in C[item].keys():
                    C[item][j] = 0
                C[item][j] += items[item] * items[j]
    self.W = dict()
    for i, related_items in C.items():
        if i not in self.W.keys():
            self.W[i] = dict()
        for j, cij in related_items.items():
            try:
                self.W[i][j] = cij / (math.sqrt(N[i]) * math.sqrt(N[j]))
            except ZeroDivisionError:
                self.W[i][j] = 0
```

图 4.19 协同过滤算法计算相似度

另外，还需要使用物品属性信息通过基于内容的推荐算法计算物品间的相似度sim_c，并通过混合相似度sim_i和sim_c去预测用户对物品的评分。

最后一步为生成推荐信息，需要使用到用户对物品的预测评分，并根据预测评分的高低顺序为每位用户生成推荐菜品列表并将其保存到数据库中。其核心实现代码如图 4.20 所示。

```
def recom(k, train, user_id, sim_data, conn):
    rank = dict()
    seen_items = train[user_id]
    for item, pi in seen_items.items():
        for j, sim in sorted(sim_data[item].items(), key=lambda x: x[1], reverse=True)[:k]:
            if j in seen_items:
                continue
            if j not in rank:
                rank[j] = 0
            rank[j] += round(sim * pi, 3)
    recom_value = list()
    for item, score in sorted(rank.items(), key=lambda x: x[1], reverse=True):
        recom_value.append((user_id, item))
    count = dbUtil.add_recom(conn, recom_value)
    if count > 0:
        print("记录插入成功", count, '条')
```

图 4.20 生成推荐信息

4.4.4 服务器端实现

服务器端，包含业务层与数据层，主要为应用端提供各种业务逻辑服务，使用 Java 语言和 SpringBoot 框架进行开发。服务器端主要是对业务层进行实现，并将其分为三层进行开发，分别为表现层、业务层与持久层。表现层使用 SpringBoot 中的 SpringMVC 模块进行开发，持久层使用 MyBatis 框架辅助开发[48]。

表现层主要是和应用端进行数据交互，其工作流程如图 4.21 所示。应用端发送网络请求给 SpringMVC 的前端控制器，前端控制器收到请求调用处理器映射器，处理器映射器根据请求的 URL 找到具体的处理器，生成处理器对象后和处理器拦截器一并返回给前端控制器。然后前端控制器通过处理器适配器调用处理器。处理器执行操作，完成后返回 ModelAndView 对象，处理器适配器将其返回给前端控制器，前端控制器再将它转发给视图解析器，经过解析后返回具体的 View 对象。前端控制器对 View 对象进行视图渲染，最后将结果返回给应用端。图 4.22 中展示的是表现层中的查询菜品信息代码。

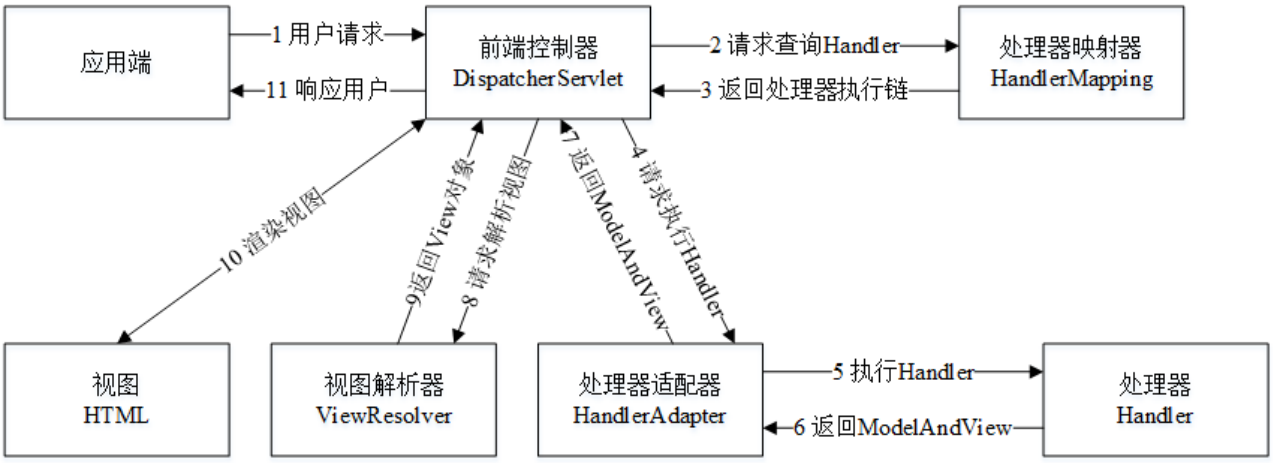

图 4.21 SpringMVC 工作流程

```
@Controller
@Api(tags = "商品数据接口")
public class ProductController {
    @Autowired
    ProductService productService;
    @GetMapping(value = "/showAllFood")
    @ResponseBody
    private ListVo showAllFood(String page, String limit) {
        List<FoodVo> foodList = productService.getAllFoodForPage(page, limit);
        ListVo listVo = new ListVo();
        listVo.setCode(0);
        listVo.setMsg("success");
        listVo.setCount(productService.getFoodRowsCount());
        listVo.setData(foodList);
        return listVo;
    }
}
```

图 4.22 表现层查询菜品信息代码

业务层主要是处理具体的业务逻辑，被表现层所依赖，通常从表现层的处理器中获取需要的参数，然后执行具体操作，并将结果返回给表现层。下图 4.23 展示的是业务层查询菜品信息代码。

```
@Service
public class ProductServiceImpl implements ProductService {
    @Autowired
    FoodMapper foodMapper;
    @Override
    public List<FoodVo> getAllFoodForPage(String page, String size) {
        Map<String, Integer> pageMap = new HashMap<>();
        int pageOne = Integer.valueOf(page);
        int pageSize = Integer.valueOf(size);
        int start = (pageOne - 1) * pageSize;
        pageMap.put("page", start);
        pageMap.put("size", pageSize);
        List<FoodVo> foodList = foodMapper.queryFoodForPage(pageMap);
        return foodList;
    }
}
```

图 4.23 业务层查询菜品信息代码

持久层主要和数据库进行交互，根据业务层传来的指令对数据库中的表记录做增删改查操作。持久层查询菜品信息代码如图 4.24 所示。

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
      PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
      "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.lihaogn.sell.mapper.FoodMapper">
   <select id="queryFoodForPage" parameterType="Map" resultType="com.lihaogn.sell.vo.FoodVo">
      SELECT f.food_id,f.food_name,f.food_price,f.food_stock,
      f.food_image,f.food_is_hot,f.food_desc,f.food_status,f.fcc_id,f.ftc_id,
      f.food_create_time,f.food_sale_count,f.food_score,
      CONCAT_WS(',',fcc.fcc_name,ftc.ftc_name,f.food_material) AS category_name
      FROM food f,food_cook_category fcc,food_type_category ftc
      WHERE f.fcc_id = fcc.fcc_id AND f.ftc_id = ftc.ftc_id
      ORDER BY f.food_create_time DESC
      LIMIT #{page},#{size };
   </select>
</mapper>
```

图 4.24 持久层查询菜品信息代码

4.5 本章小结

本章主要介绍智能推荐点餐系统的设计与实现。首先对系统的整体架构进行介绍，然后对各个功能模块的设计进行说明并对数据库中实体表的字段进行了介绍。最后对各个功能模块的实现进行了具体地阐述。

第五章 系统测试与总结

系统测试是软件开发过程中的一个重要组成部分，通过测试可以发现系统设计时的问题以及系统运行时的错误，从而保证系统在使用的时候是可靠的。

5.1 测试环境搭建

5.1.1 测试平台环境说明

微信小程序运行在智能手机上；餐饮管理应用、服务端应用以及推荐系统运行在阿里云的服务器上，商家可以通过浏览器访问餐饮管理应用。服务器上安装着 MySQL 数据库与 Redis 数据以及 Java、Python 语言运行所需的环境。下面将简要介绍系统测试的硬件环境与软件环境。

（1）硬件环境

系统所用的服务器为租用的阿里云服务器。微信点餐小程序运行在小米 9SE 手机上，餐饮管理应用通过个人电脑上的浏览器进行访问。所用硬件参数如表 5.1 所示。

表 5.1 系统测试环境硬件参数

类型	基本参数
智能手机	CPU:高通骁龙 712 @2.3GHz； 运行内存：6GB；
个人电脑	CPU:AMD Ryzen5 2600 @3.40Hz; 运行内存：16GB；
服务器	CPU：Intel Xeon E5-2682v4 @2.5GHz 运行内存：2GB； 网络带宽：1Mbps

（2）软件环境

微信小程序、餐饮管理应用及系统服务程序所需的的软件环境如表 5.2 所示。微信小程序需要运行在微信 APP 中，所以手机上需要安装该软件。餐饮管理应用需要使用到浏览器来对其进行访问，所以通过在个人电脑上安装的 Google Chrome 浏览器对其进行访问并测试。JDK 是 Java 语言的软件开发工具包，其中包含 Java 的运行环境和 Java 开发工具，Java 应用程序的运行需要依赖于它。Python 软件选择 3.0 以上的版本，之前的版本会存在兼容问题导致程序运行出错。MySQL 数据库使用 5.6.0 以上版本，之前版本有些功能不支持。

表 5.2 系统测试环境软件参数

类型	操作系统	软件名称	软件版本
智能手机	MIUI 11.0.5 (Android 9)	微信	7.0.9
个人电脑	Windows 10 专业版	Google Chrome	75.0.3770.90
服务器	Linux-Centos 6.8	JDK	1.8.0_201
		Python	3.6.7
		MySQL	5.6.22
		Redis	3.0.0

5.1.2 应用运行环境安装

（1）JDK 安装

服务器端应用程序的运行需要在 Java 运行环境中，所以需要首先在服务器上安装 JDK。其安装过程如下：

步骤 1，下载 JDK 的安装包“jdk-8u201-linux-x64.tar.gz”到服务器上，注意要选择 Linux 版本的。

步骤 2，使用解压缩命令解压 JDK 安装包到指定目录，该命令如下所示：

```
tar -zxf jdk-8u201-linux-x64.tar.gz -C ../app/
```

步骤 3，配置环境变量，使得 Java 命令可以全局使用。首先打开用户家目录中的“.bash_profile”隐藏文件，并在文件末尾添加如下配置内容：

```
JAVA_HOME=/home/webapp/app/jdk1.8.0_201
PATH=$JAVA_HOME/bin:$TOMCAT_HOME/bin:$REDIS_HOME/bin:$PATH
export PATH
```

“JAVA_HOME”表示 JDK 的安装目录即 JDK 安装包解压后的目录位置。然后保存并关闭文件，最后使用“source .bash_profile”命令使得配置生效。

至此，JDK 安装完毕。

（2）Python 安装

Python 的安装与 JDK 的安装类似，这里不再赘述。

5.1.3 数据库安装

（1）MySQL数据库的安装

步骤1，下载MySQL安装包“MySQL-5.6.25-1.el6.x86_64.rpm-bundle.tar”到服务器上，注意选择Linux版本的。

步骤2，将MySQL安装包解压到指定目录中，解压命令如下：

```
tar –xf MySQL-5.6.25-1.el6.x86_64.rpm-bundle.tar –C /usr/local/mysql/
```

步骤3，解压完安装包后，开始安装MySQL。进入到解压目录中，执行下面的的命令安装MySQL服务端与客户端：

```
rpm -ivh MySQL-server-5.6.25-1.el6.x86_64.rpm
rpm -ivh MySQL-client-5.6.25-1.el6.x86_64.rpm
```

步骤4，启动MySQL并进行初始配置。初始配置包括更改用户初始密码、将MySQL服务加入到系统服务中并开启自动启动、设置字符编码。在安装MySQL服务端时会告知用户初始密码所在文件位置。

至此，MySQL安装完毕。

（2）Redis数据库的安装

Redis数据的安装比较简单，直接将安装包下载到服务器上并进行解压，然后进入解压目录中的bin目录下，执行“./redis-server”命令即可使用Redis。

5.2 系统功能测试

系统功能测试主要是对系统中实现的各种功能进行测试，检查系统中的各个功能是否满足业务需求，是否能够正常运行并达到预期的效果。下面几小节将对系统的主要功能模块进行测试。

5.2.1 移动端应用功能测试

移动端即微信点餐小程序，其服务主体为消费者，主要功能是为消费者提供一种便捷的点餐方式。按照功能可以分为菜单浏览与选择、推荐信息展示、订单展示、以及用餐评价功能。

菜单浏览与选择功能主要为为消费者提供所有菜品信息展示，允许消费者选择想要的菜品并加入购物车中，查看购物车并在购物车中增减菜品。该功能详细测试情况如表 5.3 所示。

推荐信息展示功能为用户提供快速点餐服务，用户通过选择用餐人数获取系统为其生成的一套推荐菜品并展示在页面中。详细测试情况如表 5.4 所示。

订单展示功能，为消费者列出其已经下单的所有菜品信息，方便用户查看某个菜品是否已经送达。同时为消费者提供加菜和付款功能按钮。详细测试情况如表 5.5 所示。

表 5.3 菜单浏览与选择功能

测试功能	操作步骤	预期目标	测试结果
菜单浏览	进入微信小程序，浏览菜品信息；点击分类按钮查看对应菜品信息；	小程序页面正常显示出菜品的分类、图片、名称、价格信息；正常显示出操作按钮图标等页面组件信息；	通过
选择菜品	点击菜品信息栏中的加号或减号；	点击加号按钮后显示出所点菜品份数；点击减号菜品份数减 1，若份数为 0 则不显示；点击加减号后底部信息栏动态显示出总金额与选择的所有菜品份数；	通过
查看购物车	点击底部购物车图标进入购物车；点击购物车中每个菜品信息后的操作按钮；点击清空购物车按钮；	进入购物车后能够看到所有选择的菜品信息；点击加减按钮增加或减少菜品份数；能够清空购物车中所有项目；	通过

表 5.4 推荐信息展示功能测试

测试功能	操作步骤	预期目标	测试结果
选择用餐人数	点击“为您点餐”分类按钮，选择推荐人数。	页面显示用餐人数信息弹窗；选择用餐人数显示选中状态；点击取消按钮，弹窗消失；点击确认按钮，获取推荐信息；	通过
查看推荐信息	浏览推荐菜品信息，点击增加或减少按钮；	页面正常显示推荐菜品信息，包括菜品名称、图片、价格以及操作按钮；点击增减按钮，显示菜品所选份数并加入到购物车中；	通过

表 5.5 订单展示功能测试

测试功能	操作步骤	预期目标	测试结果
订单信息显示	点击确认下单按钮进入订单信息显示页面；浏览订单信息；	下单后小程序跳转到订单显示页面；订单信息显示正确，功能按钮图标显示正常；	通过
加菜	点击加菜按钮，进入菜单界面；点击导航栏返回按钮从菜单界面返回到订单显示页面；	页面正常跳转；	通过

用餐评价功能，为消费者提供评价服务，消费者可以对自己品尝的每一道菜品进行评价，

表示自己喜欢或者不喜欢。详细测试情况如表 5.6 所示。

表 5.6 用户评价功能测试

测试功能	操作步骤	预期目标	测试结果
展示消费的菜品信息	付款后进入评价界面，查看评价页面信息；	页面跳转正常；评价页面显示所有消费的菜品即评价按钮图标。	通过
菜品打分并提交	对每个菜品进行评价，点击喜欢或不喜欢按钮并显示选中状态；点击完成按钮提交结果；	点击按钮正常并显示出选中状态；点击完成按钮，如果没有进行评价，则给出提示信息，如果已有评价，则提交结果并跳转到菜单页面。	通过

5.2.2 网络端应用功能测试

网络端即餐饮管理应用，其服务主体为商家，主要为商家提供餐饮服务中各类信息的管理功能。按照功能可以分为管理员与用户信息管理、菜品信息管理、订单信息管理以及销售情况统计功能。

管理员与用户信息管理功能，主要是对商家工作人员与消费者账号进行管理。详细测试情况如表 5.7 所示。

表 5.7 管理员与用户信息管理功能测试

测试功能	操作步骤	预期目标	测试结果
查看管理员与用户信息	打开管理员信息页面；打开用户信息页面；	页面显示管理员和用户的具体信息；页面操作按钮图标显示正常；	通过
添加、编辑、删除单个管理员信息	在管理员信息页面，点击添加、编辑、删除按钮执行响应操作。	弹出添加、编辑窗口，填写相应内容后提交正常；删除单个管理员信息正常；	通过
删除单个用户信息	在用户信息页面，点击删除按钮。	正常删除一条用户信息，并给出删除成功提醒。删除用户信息同时删除其对应的订单信息	通过
批量删除管理员与用户信息	在管理员和用户信息页面分别选择要删除的信息行并点击批量删除按钮。	能够删除多个管理员或用户信息，并给出删除成功提示。	通过
根据条件搜索管理员与用户信息	在管理员和用户信息页面分别使用搜索功能搜索出对应的信息。	能够按照指定条件查询出相应的管理员或用户信息。	通过

菜品信息管理功能，主要为商家提供菜品相关信息的添加、查询、修改、删除功能。菜品相关信息包括菜品的基本信息、分类信息以及食材信息。详细测试情况如表 5.8 所示。

表 5.8 菜品信息管理功能测试

测试功能	操作步骤	预期目标	测试结果
查看菜品相关信息	进入菜品信息及其分类与食材信息页面	页面正常显示出菜品相关信息	通过
添加、编辑、删除单个菜品信息	在菜品信息相关页面中执行添加、编辑、删除操作	能够正常添加、编辑、删除单个菜品相关信息	通过
批量删除菜品信息	在菜品信息相关页面中选中多个要删除的条目，点击批量删除按钮	能够批量删除所选菜品信息并给出删除成功提示	通过
根据条件搜索菜品信息	根据条件在页面中搜索对应的菜品信息	能够根据日期、菜品名条件查询出对应的菜品信息	通过

订单信息管理功能，主要负责消费者订单信息的维护工作，包括订单信息的查询、订单支付状态的修改以及订单信息的删除。其详细测试过程如表 5.9 所示。

表 5.9 订单信息管理功能测试

测试功能	操作步骤	预期目标	测试结果
订单信息的查询	进入订单信息页面，浏览订单信息；根据条件查询相应订单信息	页面正常显示订单信息及操作按钮图标；能够根据日期、订单状态及用户名称查询到对应订单信息	通过
修改订单支付状态	点击支付状态修改按钮	未支付订单转为已支付状态	通过
查看订单详细信息	点击订单详情按钮查看订单详情	页面显示订单详情弹窗，列出订单中包含评分在内的菜品信息	通过
删除订单信息	点击删除按钮删除单个订单信息；选中多个订单信息，点击批量删除按钮删除	能够正常删除和批量删除订单信息，并给出删除成功提示	通过

销售情况统计功能，为商家统计出餐厅的销售情况，包括销量、销售额及最受欢迎的菜品。详细测试过程如表 5.10 所示。

表 5.10 销售统计功能测试

测试功能	操作步骤	预期目标	测试结果
销售情况的查询	进入销售统计页面，浏览销售统计图与表	页面正常显示出各种统计数据，正常显示统计柱状图与折线图。	通过

5.3 系统性能测试

性能测试结果与服务器的硬件配置有关，高配置的服务器其性能更好，用户每秒的访问并发量更高。本系统应用服务部署在 Linux 服务器上，使用 JMeter 工具[49]对服务器进行并发访问测试。通过在 JMeter 上设置并发访问数量、并发访问时间以及需要访问的 URL 模拟多名用户对服务器进行同时访问的情况。详细测试参数如表 5.11 所示。表中线程数表示的是用户数，一个线程代表一个用户。执行时长表示所有线程在多长时间内全部启动完。测试接口表示的是要进行压力测试的服务接口，表中包含的三个接口的功能分别是查询所有菜品信息，接收用户的订单信息和接收用户的评价信息。针对这三个服务接口，使用表中的参数进行测试，1 秒内 100 个用户的访问成功率为 100%，此结果满足餐厅内点餐的需求。

表 5.11 并发测试参数

参数名称	参数值
线程数	100
执行时长（秒）	1
测试接口	http://www.lihaogn.top/sell/wGetAllFood
	http://www.lihaogn.top/sell/wAddOrder
	http://www.lihaogn.top/sell/wUpdateOrderItemScore

5.4 全文工作总结

近年来，随着互联网的飞速发展，网络数据呈现爆炸式的增长，在这个背景下，数据分析与挖掘工作被重视起来，其中个性化推荐服务也得到了空前的发展。同时互联网的发展促进了其他传统行业的数字化发展进程，本文选择餐饮业作为推荐服务的应用场景，根据餐饮业的服务流程设计开发了一个基于微信小程序的智能推荐点餐系统，通过改进传统的协同过滤推荐算法以及使用流行的微信小程序为用餐的客户提供方便快捷自由的点餐方式。

本文的主要工作包括以下几个方面：

（1）分析了本课题的研究背景，总结了推荐系统和点餐系统的国内外发展状况及各自尚存在的问题。

（2）研究了基于协同过滤的推荐算法、基于内容的推荐算法以及基于关联法则的推荐算法。通过分析用户的历史行为记录信息和物品属性信息预测出物品评分，然后根据这些预测评分的高低为目标用户推荐商品。针对基于协同过滤的推荐算法中存在的数据稀疏性高、物品冷启动以及物品相似度计算解释性较差的问题，通过混合另外两种推荐算法对其缺点进行改进：对于数据稀疏性问题，使用基于关联法则的推荐算法对物品进行评分并将这些评分填

充到用户-物品评分矩阵中从而降低矩阵的数据稀疏度，提高了推荐的准确度。针对物品冷启动和推荐解释性较差的问题，使用基于内容的推荐算法计算出物品间的相似度并将其与基于协同过滤推荐算法得到的物品相似度结合，并使用结合后的相似度预测物品评分，降低了预测误差，提高了推荐精度。为了能够让用餐的顾客实现快速点餐，在为其推荐菜品时根据其所选的用餐人数确定合适的菜品数目并结合考虑到荤素种类的一个特定推荐策略生成一套菜品供其选择。同时为了解决用户冷启动问题，系统会优先选择最受欢迎的菜品推荐给顾客。

（3）通过分析顾客在餐厅内的点餐流程以及商家对餐饮信息的管理需求，结合多种开发技术并根据系统的架构与功能模块设计对基于微信小程序的智能推荐点餐系统进行了实现。

5.5 未来展望

本文给出了一种混合推荐算法，解决了基于协同过滤的推荐算法中存在的缺点并将其应用在一个基于微信小程序的智能推荐点餐系统中，但还有些不足之处：

（1）在基于协同过滤算法的改进中，还可以使用基于模型的方法来降低评分矩阵的稀疏度[50]。在计算物品相似度的时候，可以考虑时间对于用户兴趣度的影响，从而修正物品相似度计算的公式。

（2）在推荐系统中，只使用到了一个推荐引擎来为用户提供推荐服务，在后序的研究中，可以加入一些基于机器学习或深度学习的推荐算法[51]-[52]。另外，随着用户规模的增加，数据量将越来越大，日后可以采用分布式计算框架来加速数据的分析从而更快地产生推荐结果。

（3）在推荐策略上，还应该考虑到顾客的消费预算以及顾客的口味等条件。

（4）本系统中，还应该为商家提供一个后厨显示功能模块，将顾客的点餐信息直接反馈给厨房人员。

参考文献

[1] Li C T , Shan M K , Jheng S H , et al. Exploiting concept drift to predict popularity of social multimedia in microblogs[J]. Information Sciences, 2016, 339:310-331.

[2] 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, 31(14):54-56.

[3] 李川. 实时个性化推荐系统的设计与实现[D]. 北京: 北京邮电大学, 2015.

[4] Xueying D . Study on A Recommendation Algorithm of Crossing Ranking in E-commerce[J]. International Journal of u- and e- Service, Science and Technology, 2014, 7:53-62.

[5] Jiwei, Qin, Qinghua, et al. An Emotion-oriented Music Recommendation Algorithm Fusing Rating and Trust[J]. International Journal of Computational Intelligence Systems, 2014.

[6] 揭正梅. 基于协同过滤的高校个性化就业推荐系统研究[D]. 2015.

[7] 李媛. 电子商务个性化推荐关键技术研究[D]. 2012.

[8] 微信小程序[J]. 上海信息化, 2017(01):9-10.

[9] 宁家骏. “互联网+”行动计划的实施背景、内涵及主要内容[J]. 电子政务, 2015(6):32-38.

[10] 阮玉洁. 外卖软件的使用特点分析——以美团外卖为例[J]. 商场现代化, 2016(25):59-60.

[11] 段海梦. 基于外卖订餐软件的 O2O 电商模式现状分析及优化[J]. 科技致富向导(24):252-252.

[12] 王超斌. 基于 Android 的移动点菜系统的研究与实现[D]. 昆明理工大学, 2013.

[13] 赵敬, 李贝. 微信公众平台发展现状初探[J]. 新闻实践, 2013(8):8-10.

[14] 霍婉晖. 基于微信公众平台的智能点餐系统的设计与实现[D].吉林大学,2016.

[15] 宋丹丹. 基于微信小程序的美食点餐推荐系统的设计与实现[J]. 信息通信, No.180(12):94-95.

[16] Bradley Miller John, John T. Riedl, Joseph A. Konstan. Experiences with GroupLens: Making Usenet Useful Again[J]. 1997.

[17] Bradley N. Miller, Istvan Albert, Shyong K. Lam, etc. MovieLens Unplugged: Experiences with a Recommender System on Four Mobile Devices[C]// Proc of the Human-computer Interaction Conference. 2004.

[18] Zhang J , Lin Y , Lin M , et al. An effective collaborative filtering algorithm based on user preference clustering[J]. Applied Intelligence, 2016, 45(2):230-240.

[19] Greg Linden, Brent Smith, Jeremy York. Amazon. com recommendations: Item-to-item collaborative filtering[J]. IEEE Internet Computing, 2003, 7(1):76-80.

[20] Sarwar B M. Application of Dimensionality Reduction in a Recommender System : A Case Study[C]// ACM WebKDD 2000 Web Mining for E-Commerce Workshop. 2000.

[21] Rafailidis D, Daras P. The TFC Model: Tensor Factorization and Tag Clustering for Item Recommendation in Social Tagging Systems[J]. IEEE Transactions on Systems Man & Cybernetics Part B, 2013, 43(3):673-688.

[22] Nasiri M, Minaei B. Increasing prediction accuracy in collaborative filtering with initialized factor matrices[J]. Journal of Supercomputing, 2016, 72(6):1-13.

[23] Jia D, Zhang F, Liu S. A Robust Collaborative Filtering Recommendation Algorithm Based on Multidimensional Trust Model[J]. Journal of Software, 2013, 8(1):806-809.

[24] Chen H, Li Z, Hu W. An improved collaborative recommendation algorithm based on optimized user similarity[J]. Journal of Supercomputing, 2016, 72(7):1-14.

[25] 黄逸珺, 杜梦甜, 傅玉婷. 基于用户感知的个性化推荐系统效果研究——以淘宝电商平台为例[J]. 北京邮电大学学报(社会科学版), 20(05):28-35.

[26] 周军锋, 汤显, 郭景峰. 一种优化的协同过滤推荐算法[J]. 计算机研究与发展, 2004, 41(10):1842-1847.

[27] 邓爱林, 左子叶, 朱扬勇. 基于项目聚类的协同过滤推荐算法[J]. 小型微型计算机系统, 2004, 25(9):1665-1670.

[28] 邓爱林, 朱扬勇, 施伯乐. 基于项目评分预测的协同过滤推荐算法[J]. 软件学报, 2003, 14(9):1621-1628.
[29] 朱二华. 基于 Vue.js 的 Web 前端应用研究[J]. 科技与创新, 2017(20):119-121.
[30] 杨家炜. 基于 Spring Boot 的 web 设计与实现[J]. 轻工科技, 2016(7):86-89.
[31] 胡启敏, 薛锦云, 钟林辉. 基于 Spring 框架的轻量级 J2EE 架构与应用[J]. 计算机工程与应用, 2008, 44(5):115-118.
[32] 唐汉明, 翟振兴， 关宝军. 深入浅出 MySQL[M]. 人民邮电出版社, 2014.
[33] 李子骅. Redis 入门指南[M]. 人民邮电出版社, 2013.
[34] 王春才, 邢晖, 李英韬. 推荐系统中的数据预处理方法研究[J]. 信息技术, 2016(9):38-39.
[35] 熊馨, 王卫平, 叶跃祥. 电子商务个性化产品推荐策略研究[J]. 科技进步与对策(7):164-166.
[36] Iván Cantador, Ro Bellogín, David Vallet. Content-based Recommendation in Social Tagging Systems[C]// Acm Conference on Recommender Systems. ACM, 2010.
[37] Dong J , Li J , Li G , et al. Research on User-based Normalization Collaborative Filtering Recommendation Algorithm[C]// 2nd International Conference on Advances in Mechanical Engineering and Industrial Informatics (AMEII 2016). 2016.
[38] Suyun Wei, Ning Ye, Shuo Zhang,等. Collaborative Filtering Recommendation Algorithm Based on Item Clustering and Global Similarity[C]// Business Intelligence and Financial Engineering (BIFE), 2012 Fifth International Conference on. IEEE Computer Society, 2012.
[39] 王竹婷. 关联规则评分预测的协同过滤推荐算法[J]. 合肥学院学报(综合版), 26(1):70-74.
[40] Peter Harrington. 机器学习实战[M]. 人民邮电出版社, 2013.
[41] Salahli, Mehmet Ali, Gasimzade, Tokay, Alasgarova, Flora, etc. The Use of Predictive Models in Intelligent Recommendation Systems[J]. Procedia Computer Science, 102:515-519.
[42] 张小红. 协同过滤中的相似性度量方法的研究[J]. 无线电通信技术, 2013, 39(1):94-96.
[43] 项亮. 推荐系统实践[M]. 人民邮电出版社, 2012.
[44] 刘畅, 王玉龙. 推荐系统冷启动问题分析[J]. 电信网技术, 2017(1):65-68.
[45] Aggarwal C C . Frequent Pattern Mining[M]. Springer New York, 2014.
[46] Zibin Zheng, Hao Ma, Michael R. Lyu, 等 . WSRec: A Collaborative Filtering Based Web Service Recommender System[C]// 2009 IEEE International Conference on Web Services. IEEE, 2009.
[47] Oliphant, Travis E. Python for Scientific Computing[J]. Computing in Science & Engineering, 9(3):10-20.
[48] 阳小兰, 罗明. 基于 Spring+SpringMVC+MyBatis 网上论坛的设计与实现[J]. 科学技术创新, 2016(36):279-280.
[49] 秦露. Web 性能测试—Apache JMeter[J]. 时代报告：学术版(3):145-146.
[50] 朱冲奘, 蔡其锑, 张登辉. 基于 SVD 协同过滤的餐饮推荐系统设计[J]. 浙江树人大学学报(自然科学版), 2018, 18(02):5-9.
[51] 周齐. 基于机器学习的推荐系统[J]. 电子技术与软件工程, 2016(24):173-173.
[52] 黄立威, 江碧涛, 吕守业, et al. 基于深度学习的推荐系统研究综述[J]. 计算机学报, v.41；No.427(7):191-219.

附录 1 攻读硕士学位期间申请的专利

[1] 李涛、李昊，一种基于物品协同过滤与关联规则的混合算法的推荐方法，201910667120.X，2019.7.23；

致谢

时光如梭，转眼间研究生生涯即将结束。在南邮的这两年多的时间里，我不仅学到了很多知识，在生活上和工作上也积累了很多经验，这些知识和经验将对我以后的生涯产生深远的影响。从刚毕业的本科生一路走来并不是一帆风顺的，途中有些许困难，但还好有周围同学、老师以及家人的帮助与鼓励让我不畏险阻，一路前行，顺利毕业。在此向他们表示由衷的感谢！

首先，向我的导师李涛教授表达最深的敬意与真诚的感谢。在学术上，导师严谨负责的治学态度，认真的科研精神感染着我，在研究自己的专业方向之余还一直关注着行业的最新技术动向，保持着学习的热情。在学习上，导师不仅给我指明了研究的方向，推荐相关的资料，还对我的课题研究给予了诸多宝贵的建议与指导。在生活上，十分感谢导师的关心与宽容。

其次，我要感谢自己的师兄师姐以及师弟，他们在学习和生活上给予了我许多无私的帮助，将自己的学习资料、学习经验以及工作生活经验无私地分享了给我。同时，我还要感谢我的舍友和其他同学，感谢他们在这两年多的时间里一直陪伴着我，给予我鼓励与帮助，在科研的道路上并肩前行。

最后，要感谢学校和家人。感谢学校给予了良好的学习和生活环境。感谢我的家人在这段时间内的大力支持与鼓励，他们是我最坚强的依靠，让我一路奋勇前行。