|  |  |
| --- | --- |
| 学号 | 2015301500092 |
| 密级 |  |

**武汉大学本科毕业论文**

|  |
| --- |
| **移动套餐智能检测系统的设计与实现** |

|  |  |
| --- | --- |
| 院（系）名 称 ： | 计算机学院 |
| 专 业 名 称 ： | 计算机科学与技术 |
| 学 生 姓 名 ： | 孙加辉 |
| 指 导 教 师 ： | 喻丹丹 副教授 |

二〇一九年五月

**郑 重 声 明**

本人呈交的学位论文，是在导师的指导下，独立进行研究工作所取得的成果，所有数据、图片资料真实可靠。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确的方式标明。本学位论文的知识产权归属于培养单位。

本人签名： 日期：

摘 要

近年来，软件工程发展迅速，软件开始充斥我们生活的每一个角落。但是软件由人工设计实现，这一特性使得软件不可避免的存在着或多或少的缺陷。而随着近些年来IT行业的巨大发展，越来越多的领域与软件行业存在联系，对于软件的质量要求也越来越高。当前，中国移动（海南）创建套餐的方式是手动编辑上百行SQL语句之后插入到数据库中，一旦SQL语句中出现错误，将会导致重大损失，而人工检测SQL语句十分依赖经验丰富的员工，且很难发现SQL语句中的所有错误，本文针对该问题提出了一种基于数据挖掘的自动化检测SQL语句的解决方案。

本文主要通过协同过滤算法和关联规则挖掘算法分析创建套餐的SQL语句，通过对大量历史套餐的SQL语句的分析，对创建新套餐的SQL语句是否存在错误进行预测。其中利用协同过滤算法分析套餐与创建该套餐的SQL语句中涉及到的数据库表之间的关系，通过计算历史套餐的SQL语句与新创建的套餐的SQL语句之间的相似度找出与新创建的套餐相似的历史套餐，并根据相似的历史套餐所涉及到的数据库表对新创建的套餐进行“推荐”，即错误预警。协同过滤算法只能检测新创建的套餐是否遗漏或多添加了某些数据库表，而对于数据库表内部错误却无能为力，为了能够检测数据库表内具体数值是否出错，本文通过关联规则挖掘算法进行进一步的检测。对于创建套餐涉及到的数据库表，先找出其中的关键属性，将数据库表名作为事务名，关键属性的具体数值作为事务内容就可以得到一条事务，将所有历史套餐中同一个表的事务组成事务集，并利用关联规则挖掘算法对该事务集进行关联分析，就可以得到该事务集的数据库表的频繁项集和置信度关系，进一步可以对新创建套餐的SQL语句该数据库表的关键属性是否有错进行检测。

为了方便用户操作，提高用户体验，本文将通过Django框架搭建web应用程序，其中包括用户登录、上传文件、扫描检测、生成检测报告等功能。

本文旨在充分利用数据挖掘技术，对SQL语句进行自动化处理分析，解决中国移动（海南）人工检测SQL语句费时费力且容易出错的现状，并以此推进软件测试自动化技术的研究。

关键词：数据挖掘；关联规则挖掘算法；协同过滤算法；SQL检测

**ABSTRACT**

In recent years, with the rapid development of software engineering, software began to fill every corner of our lives. However, the software is designed and implemented by human beings, which makes the software have defects inevitably. With the great development of IT industry in recent years, more and more fields are related to the software industry, and the quality requirements of software are getting higher and higher. At present, China Mobile (Hainan) creates a package by editing hundreds of lines of SQL statements and inserting them into the database manually. Once an error occurs in the SQL statement, it will lead to great losses. However, manual detection of SQL statements is dependent on experienced employees heavily, and it is difficult for them to find all the errors in SQL statements. in order to solve this problem, this paper proposes an automatic solution of SQL statement detection based on data mining.

This paper mainly analyzes the SQL statements of creating packages through Collaborative Filtering Algorithm and Association Rules Mining Algorithm. Through the analysis of many SQL statements of historical packages, this paper predicts whether there are errors in the SQL statements of creating new packages. The Collaborative Filtering Algorithm is used to analyze the relationship between the package and the database tables involved in the SQL statement while creating the package. Find the historical packages which are like the newly created package by calculating the similarity between the SQL statement of the history package and the SQL statement of the newly created package. And according to the database tables in the similar historical package, “recommend” error warnings to the newly created package. The Collaborative Filtering Algorithm can only detect whether the newly created package omits or adds some database tables but can do nothing about the internal errors of the database tables, in order to detect whether the specific values in the database tables are wrong, the Association Rules Mining Algorithm is used for further detection in this paper. In order to check the database tables involved in creating the package, find out the key attributes firstly, take the database table name as the transaction name, and the specific value of the key attribute as the transaction content, you can get a transaction. The transaction set can be composed of transactions of the same table in all historical packages. The association analysis of the transaction set is carried out by using Association Rule Mining Algorithm, and the frequent itemset and confidence rules of the database table can be obtained, with which you can further detect the values of the key properties of the database table in the SQL statement for the newly created package.

In order to facilitate users’ operations and improve users’ experience, this paper will build a web application based on Django framework, including user login, file uploading, scanning detection, generation of detection reports and other functions.

The purpose of this paper is to make full use of data mining technology to process and analyze SQL statements automatically. Solving the problem that manual detection of SQL statements by China Mobile (Hainan) is time-consuming, laborious and error-prone and to promoting the research of software test automation technology are the purposes of this paper too.

**Key Words**: data mining; Association Rule Mining Algorithms; Collaborative Filtering Algorithm; SQL detection

目 录

[1 概述 1](#_Toc5796255)

[1.1 论文选题研究的背景及意义 1](#_Toc5796256)

[1.2 国内外研究现状 2](#_Toc5796257)

[1.2.1 国外状况 2](#_Toc5796258)

[1.2.2 国内状况 2](#_Toc5796259)

[1.3 本章小节及论文结构 3](#_Toc5796260)

[2 系统设计与实现 4](#_Toc5796261)

[2.1 相关技术简介 4](#_Toc5796262)

[2.1.1 Django 5](#_Toc5796263)

[2.1.2 Nginx 5](#_Toc5796264)

[2.1.3 uWSGI 5](#_Toc5796265)

[2.1.4 MySQL 5](#_Toc5796266)

[2.2 系统搭建步骤 6](#_Toc5796267)

[2.2.1 Django项目创建 6](#_Toc5796268)

[2.2.2 uWSGI环境配置 8](#_Toc5796269)

[2.2.3 Nginx环境配置 8](#_Toc5796270)

[2.2.4 数据库设计 8](#_Toc5796271)

[3 利用协同过滤算法检测错误 9](#_Toc5796272)

[3.1 协同过滤算法简介 9](#_Toc5796273)

[3.2 协同过滤算法基本步骤 9](#_Toc5796274)

[3.3 利用协同过滤算法检测数据库表错误 10](#_Toc5796275)

[4 利用关联规则挖掘算法检测错误 14](#_Toc5796276)

[4.1 关联规则挖掘算法简介 14](#_Toc5796277)

[4.2 相关概念介绍 14](#_Toc5796278)

[4.2.1 支持度 14](#_Toc5796279)

[4.2.2 最小支持度 14](#_Toc5796280)

[4.2.3 置信度 14](#_Toc5796281)

[4.2.4 最小置信度 15](#_Toc5796282)

[4.2.5 频繁项集 15](#_Toc5796283)

[4.3 关联规则挖掘算法基本步骤 15](#_Toc5796284)

[4.4 利用关联规则挖掘算法检测关键属性错误 17](#_Toc5796285)

[5 实验成果 20](#_Toc5796286)

[5.1 实现效果描述 20](#_Toc5796287)

[5.2 系统展示 20](#_Toc5796288)

[5.3 结果分析 23](#_Toc5796289)

[6 总结与展望 24](#_Toc5796290)

[参考文献 25](#_Toc5796291)

[致谢 27](#_Toc5796292)

# 概述

## 论文选题研究的背景及意义

近年来，软件工程发展迅速，软件无处不在，充斥着我们生活的每个角落。但是由于软件是人的大脑高度智力化的体现，这一特性使软件工程不同于其他科技和生产领域，因此软件不可避免的存在着或多或少的缺陷。为了防止和减少这些缺陷，测试便成了软件生命周期中不可或缺的一个环节。到目前为止，软件测试仍然是排除和防止软件缺陷的最有效的方法。

软件测试是使用人工或者自动化的手段来运行或测试某个系统的过程，是软件开发的重要组成部分，其目的在于检验软件是否满足规定的需求或弄清预期结果与实际结果的差别。在软件测试的过程中，可以找出软件存在的缺陷以便修复。随着近些年来IT行业的巨大发展，各个领域都与软件行业存在或多或少的联系，对于软件的质量要求也越来越高。为了提高软件测试的精度和可信度、提升测试的可靠性和稳定性，软件测试自动化技术正在被引入到越来越多的软件开发过程中。相比于常规的软件测试，自动化测试将以人为中心的测试行为转化为机器执行，即模拟手工测试步骤通过执行程序语言编制的测试脚本自动的测试程序，减少人工的参与，提高了软件开发效率和准确性。自动化测试可以运行更加繁琐的测试，执行一些人工测试困难或者不可执行的测试，例如模拟大量用户场景的压力测试。由于在自动化测试时不存在执行过程中的疏忽和错误，一旦软件测试通过强有力的自动测试后，软件的信任度便可以得到保证。

当前，中国移动（海南）创建套餐的方式是手动编辑上百行SQL语句之后插入到数据库中，该方式不仅依赖于经验丰富的业务员，而且非常容易出错。为了检测SQL语句是否存在错误，需要花费大量的人力和时间。对于这样的大型企业而言，一旦套餐出现失误，便会导致极大的损失。在本系统设计以前，中国移动（海南）曾针对通过SQL创建套餐容易出错的问题开发过可视化界面，然而由于套餐种类繁多，需要员工配置的项较多，各套餐涉及到的业务也存在区别，整个可视化界面显得特别冗长，且当需要创建的套餐包含了之前没有的业务时，可视化界面便无法满足需求。为了解决这一问题，本论文设计了基于数据挖掘的SQL语句自动化测试系统。

本系统对SQL语句进行处理和分析，从中挖掘有效的数据，并利用协同过滤算法和关联规则挖掘算法对此进行分析，推断出SQL中可能存在的错误。本论文选题旨在充分利用数据挖掘技术，对SQL语句进行自动化处理分析，改变人工检测SQL语句费时费力的现状。

## 国内外研究现状

软件测试的产生是伴随着软件的产生的，在软件开发之初，软件的规模较小，开发过程杂乱无章，软件测试的定义几乎等同于程序调试，仅仅用于纠正软件开发时存在的已知问题。随着时间推移，软件测试的内涵也越来越丰富，人们对于软件测试的了解也越来越深入，软件测试在软件开发周期中开始扮演越来越重要的角色。

### 国外状况

在国际上软件测试是一件非常重要的工作，测试也成为了一个相对独立的职业。在很多软件开发公司中，测试人员的数量不低于开发人员的数量，在微软与IBM等这样的大型软件公司测试人员和开发人员的比例则达到了2比1甚至更高。在软件测试的技术方面，自动化测试被使用的越来越广泛，正在朝着通用化、标准化、智能化的方向迈进。近年来，自动化软件测试技术发生重大变革，以综合通用的自动化软件测试思想代替单一的自动化测试思想，采用通用的软硬件平台以实现资源共享的思想被大多数软件开发企业采纳。该思想认为，自动化测试应该采用共同的测试策略，从设计过程开始就通过“增值开发”的方式使当前阶段的开发成果能够被后一阶段的测试设备的研制所利用，测试程序集可以被移植，软件模块能够被复用，使用通用的标准，缩短测试的时间，且使软件模块方便扩展和升级，降低软件开发成本。

### 国内状况

由于国内软件行业起步较晚，所以目前软件测试在国内的地位并不高，大多数企业的软件测试仅停留在软件单元测试、集成测试和功能测试上。软件测试的标准化和规范化程度不够。测试人员的数量与软件开发过程中的实际需求存在差距，导致软件测试不够充分，软件中可能存在的问题无法得到解决。目前国内缺乏商业化的操作机构，软件测试产业化程度不高，与国外相比还存在差距。国内大多互联网公司缺乏对自动化测试的认识，自动化测试前期高额的开发成本让这些公司望而却步，而且相对于常规的手工测试，自动化测试的维护成本、工作量都会大很多，这些问题都大大阻碍了自动化测试在国内的发展。

## 本章小节及论文结构

本章首先介绍了软件测试和软件自动化测试的研究背景和意义，这是本文的技术前提。接着对本项目将要实现的系统进行简单介绍。最后分别介绍了软件测试在国内外的发展状况，以便于了解软件测试在国内的处境及发展前景。

第二章主要对本文所描述的系统实现需要使用到的技术进行介绍，并分析系统设计及实现详细过程。

第三章着重介绍本文应用的第一个关键算法--协同过滤算法，介绍了数据获取和数据处理方法以及在本系统中如何利用协同算法生成工具对数据进行分析。

第四章着重介绍本文应用的第二个关键算--关联规则挖掘算法，介绍了数据获取和数据处理方法以及在本系统如何利用关联规则挖掘算法生成工具对数据进行分析。

第五章对系统的实现效果进行简单介绍，展示系统的相关页面，并对系统的测试结果进行分析。

第六章对本文进行总结与展望，总结现有的不足并展望未来的工作。

# 系统设计与实现

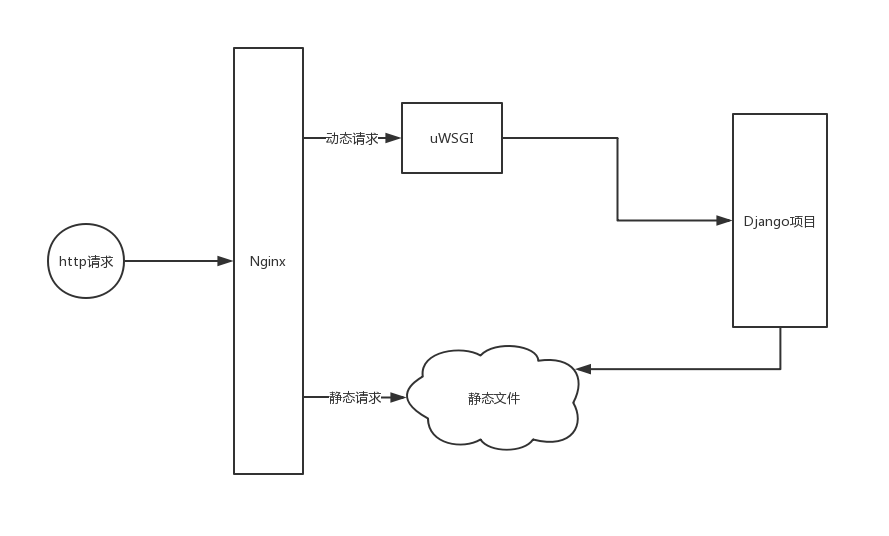
本系统旨在搭建一个移动套餐智能检测平台，帮助使用者提高工作效率，降低开发成本。本系统主要通过协同过滤算法和关联规则挖掘算法等数据挖掘方法对数据进行分析，使用者通过浏览器/服务器（B/S）模式与系统进行交互。本系统将通过Vue.js和Django搭建，用户通过交互界面输入SQL，服务器端接收数据后调用使用Python编写的服务对数据进行处理，再通过利用协同过滤算法和关联规则挖掘算法编写的服务进行分析，最后将分析结果生成报告，通过前端浏览器进行展示。本章将介绍系统框架的搭建所涉及的技术以及系统搭建的详细步骤。系统框架流程如图2.1所示，当有http请求访问服务器时，会被Nginx接收并处理，Nginx会判断该请求为静态请求还是动态请求。静态请求即访问静态文件的请求，动态请求即需要服务器对输入的数据进行处理并返回相应数据的请求。Nginx会将静态请求直接处理，而动态请求会被转发给uWSGI服务器进行处理，uWSGI服务器会对该请求进行相应的处理后再交给Django处理。

图2.1 系统框架流程

## 相关技术简介

本系统采用B/S模式进行搭建，通过Django、Nginx、uWSGI等实现，数据库则采用较为常用的MySQL。本系统涉及的相关技术大多数为开源项目，这些技术的稳定性和安全性可以得到保障，因此也可以提高了本系统的安全性和可信度。

### Django

Django是一个开源的web应用框架，使用Python语言进行开发 。采用MVC框架模式，即模型Model，视图View和控制器Controller。作为一个开源的框架，Django最显著的优点就是有强大的社区和文档支持。Django还有强大的数据库功能，使用Python的类对应数据库中的表，实现了数据库和数据模型的解耦，即数据模型不依赖于特定的数据库。Django的设计目的是可以简便快捷的开发数据库驱动的网站，重视代码重用，组件可以以插件的形式服务于整个系统。同时，Django还有许多功能强大的第三方插件，Django的可扩展性也支持用户开发自己的工具包。Django可以使用Redis等缓存系统，网页的加载速度得到了保证，因此使用Django开发的web系统可以一定程度上保证用户体验。

### Nginx

Nginx是一的款开源的轻量级的Web服务器和反向代理服务器以及电子邮件服务器。反向代理的定义与正向代理相反，正向是通过代理服务器去访问已知的地址，而反向代理是指只知道代理服务器的地址，而实际上访问哪个服务器对于访问者来说是透明的，它代理的是服务端，主要适用于服务器集群分布式部署的情况，反向代理可以隐藏服务器的信息，保证内网的安全。Nginx基于REST架构风格，以URI和URL作为依据，通过http协议提供服务。与Apache不同的是，Nginx体积小，占用内存小，支持高并发的应用场景。由于Nginx使用基于事件驱动架构，所以其可以支持数以百万级别的TCP连接，Nginx还是一个跨平台的服务器，可以运行在Windows、Linux及Mac等操作系统上。

### uWSGI

uWSGI是一款实现了WSGI、uwsgi、http等协议的服务器，其中WSGI是一种通信协议，uwsgi是一种用来定义传输信息类型的线路协议。之所以要使用uWSGI服务器，是因为Python并不可以快速的处理所有的请求，Nginx接收到动态请求时会将请求转发给uWSGI服务器，uWSGI收到后对请求进行处理，处理成web应用可以接受的形式，再调用对应的方法。

### MySQL

MySQL是一款开源的高性能关系型数据库管理系统，是一个多用户、多线程的SQL数据库服务器。MySQL以客户机/服务器的形式实现，由服务器守护进程和不同的客户程序和库组成，能够快捷、有效、安全的处理大量的数据。MySQL支持多平台，也支持多种语言开发，是当前软件开发使用的主流数据库服务器之一。

## 系统实现步骤

为了保证系统的运行效率及稳定性，本系统选择Linux作为开发环境，操作系统的版本为Ubuntu16.04。

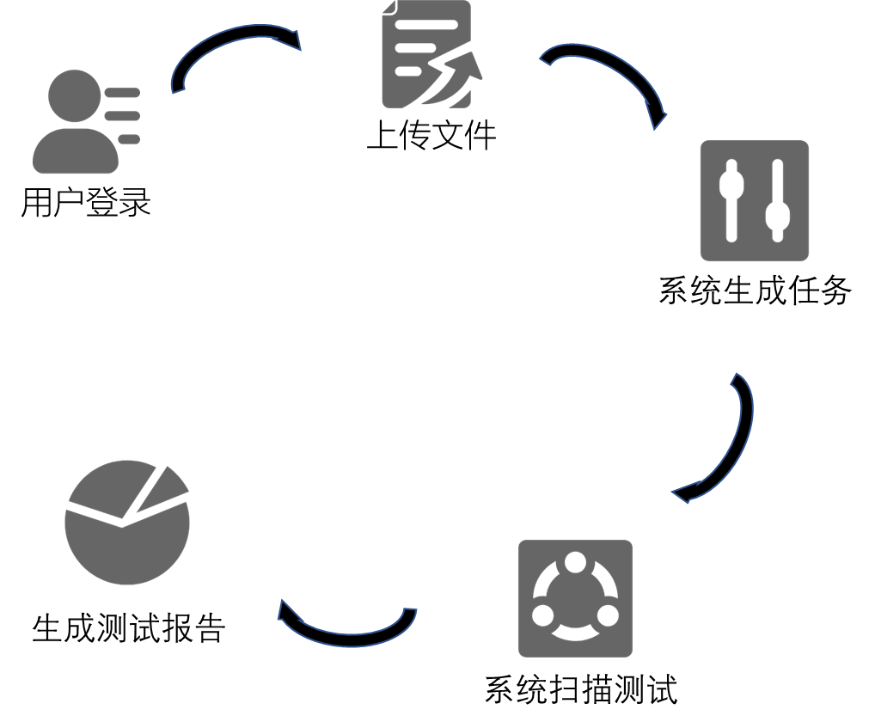
本系统虽然没有将前端和后端分成两个项目实现，但是前端所有的数据处理工作都是由JavaScript实现，极大的减少了服务器的压力，现了动态资源和静态资源分离，提高了性能和扩展性。

图2.2 系统操作流程

在搭建系统之前需要考虑系统的操作流程，本系统的操作流程如图2.2所示，首先需要验证用户的身份，用户的密码通过MD5加密方式进行加密，提高系统的安全性，用户登录之后在主页中上传需要检测的文件，系统接收用户上传的文件并对该文件创建任务，通过任务信息调用系统的扫描检测服务，最后将扫描结果保存到数据库并生成测试报告，用户可以通过浏览器在线查看测试报告。

### Django项目创建

在创建Django项目之前需要先安装Python环境，本系统所使用到的Python版本为Python 3.6，安装完Python后将安装路径配置到系统环境变量中。创建Django项目可以从其官网上下载对应版本的项目压缩包，或者使用pip命令进行安装，两种途径创建的结果并无太大区别。项目创建完成后通过pip命令安装mysqlclient与numpy等模块，所有模块导入之后新建一个应用。Django的一个项目中可以创建多个App，每个App都是相互独立的，可以移植的业务。

App创建完成后需要进行配置，首先需要配置静态文件存放的位置，其次是指定所使用的数据库为MySQL，并配置数据库名称和密码等参数，最后对MySQL数据库进行初始化操作，将需要使用到的数据表在MySQL中生成。本项目采用的MVC架构，将控制层、服务层、模型层分开，在应用根目录下新建service和model等Python包用来存放相应的python文件，当接收到前端发送的请求时会先通过urls.py中的接口配置访问指定的Controller函数，Controller函数对请求进行处理后调用service层中的方法，关联规则挖掘方法、协同过滤方法、数据处理方法等服务的函数都是在这一层实现的，当需要对数据库进行操作的时候，service层会通过model层相应的类对数据库进行操作。

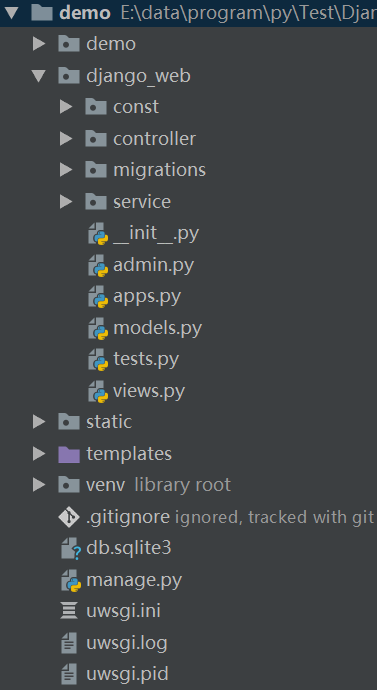
虽然前后端分离有利于系统的并行开发和降低系统的耦合度，但是由于本系统的业务较少，前后端分离开发的必要性不是很高，因此没有选择前后端完全分离的设计模式。本系统的html页面放置在系统的template路径下，css及js等静态文件存放在系统是static路径下。Django项目结构如图2.3所示。

图2.3 Django项目结构

### uWSGI环境配置

uWSGI的配置较为简单，安装完uWSGI之后，修改相应配置文件，使uWSGI服务器监听指定端口，当该端口接收到请求时，对该请求进行处理之后把请求转发到Django所监听的端口。设置服务类型为socket而非http，因为请求不是直接通过http方式访问uWSGI服务器的，而是通过Nginx服务器来访问的。

### Nginx环境配置

安装完Nginx服务器之后通过修改Nginx自带的配置文件nginx.conf对Nginx服务器进行配置。由于使用者是通过浏览器访问服务器端的，所以Nginx需要监听服务器的80端口，并将所有发送到uWSGI服务器所监听的端口，此处配置的端口号应与uWSGI配置的端口号保持一致。Nginx服务器中还应该配置静态文件的路径，当使用者发送静态请求，只访问静态文件时，Nginx服务器对该请求不做转发，而是直接处理，只有当Nginx服务器接收到动态请求时才会转发到uWSGI中进行处理。

### 数据库设计

本系统的数据库设计将从以下几个方面考虑：首先需要设计用户表(user)，用户需要登录以后才可以进行相关操作。另外，用户提交SQL文件进行检测在系统中是以“任务”的形式存在的，即每次提交都会在系统中新建一个“任务”，该任务记录本次SQL检测的用户和文件信息等内容，所以需要设计任务表(task)。检测结果将会以测试报告的形成呈现给用户，报告中包含SQL文件中可能存在的错误，因此本系统需要设计错误表(mistake)，包含错误位置和错误类型等信息，数据库表的详细设计如表2.1-表2.6所示。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性名 | 类型 | 是否为空 | 属性 | 说明 |
| id | int | Not null | Primary\_key | 用户id |
| name | varchar | Not null |  | 用户名 |
| password | varchar | Not null |  | 用户密码 |
| create\_time | varchar | Not null |  | 账号创建时间 |
| modify\_time | varchar | Not null |  | 账号最后修改时间 |

表2.1 user表设计

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性名 | 类型 | 是否为空 | 属性 | 说明 |
| id | int | Not null | Primary\_key | 任务id |
| user\_id | int | Not null | Foreign\_key | 用户id |
| file\_name | varchar | Not null |  | 文件名 |
| create\_time | varchar | Not null |  | 任务创建时间 |
| state | int | Not null |  | 任务状态 |

2.2 task表设计

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性名 | 类型 | 是否为空 | 属性 | 说明 |
| id | int | Not null | Primary\_key | 错误id |
| task\_id | int | Not null | Foreign\_key | 任务id |
| mistake\_type | varchar | Not null |  | 错误类型 |
| mistake\_grade | varchar | Not null |  | 错误预警等级 |
| mistake\_detail | varchar | Not null |  | 错误详情 |
| find\_time | varchar | Not null |  | 错误检测时间 |
| method | varchar | Not null |  | 错误处理方法 |
| extends | varchar | Not null |  | 其他 |

2.3 mistake表设计

# 利用协同过滤算法检测错误

## 协同过滤算法简介

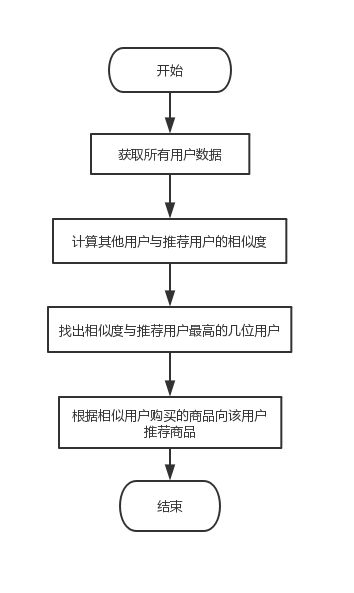
由于待检测的SQL会涉及到多表操作，所以检测的第一步应该是定位到哪些表的操作可能存在问题，之后再对可能存在错误的表进行进一步的分析。尝试了多种方法之后，本系统最终还是选择了协同过滤方法。

图3.1 协同过滤推荐商品的步骤

协同过滤的主要功能是预测和推荐，是一个常见的推荐算法，推荐商品的步骤如图3.1所示。协同过滤主要基于用户的历史操作或者其他用户的相似操作，用以挖掘用户的喜好偏向，并预测该用户可能喜好的商品进行推荐。预测推荐过程主要有两种推荐思路，一种思路是根据与该用户喜好偏向类似的其他用户的喜好来进行推荐，例如，当某用户购买了部分商品，而另一个其他用户购买的商品与这个用户的商品非常类似，大多数都是相同的，就可以向这个用户推荐他没有购买但是与他相似的用户已经购买的商品；另一种是向用户推荐与其喜好的商品类似的其他商品，这两种方式分别被称为基于用户的协同过滤算法和基于物品的协同过滤算法。本系统中使用的是第一种方式，即基于用户的协同过滤算法。由于本系统中不存在用户和产品的关系，所以本系统转换了一种思路，将一个套餐看作是一个用户，而套餐中涉及到的数据库表看作商品。基于用户的协同过滤算法面临的两个主要的问题是相似的用户信息如何获取和如何计算用户之间的相似度，第一个问题对于本系统来说比较容易解决，由于本系统是将套餐看作为用户，不同的历史套餐就可以当作其他用户。用户之间相似度的度量标准有多种，其中常用的有杰卡德相似系数与夹角余弦等，本系统将套餐间包含的不同的表的数量作为两个套餐的距离，对距离进行归一化处理便可以得到相似度。

## 协同过滤算法基本步骤

协同过滤算法的基本主要分为四步：

第一步是收集数据。协同过滤算法除了需要制定用户的信息之外，还需要大量其他用户的信息。这些信息根据应用场景的变化而变化，例如当应用场景是影院向用户推荐电影时，就应该收集用户看过的电影或者评论过的电影等信息；而当应用场景是向用户推荐商品时，就应该收集用户购买过的商品等信息。本节中以下的步骤将以向用户推荐商品为应用场景展开。

第二步是初始化数据。在计算用户相似度之前，需要对第一步中获取的数据进行处理。本算法需要从大量的数据中挖掘出有用的信息，首先通过Python编写的处理函数进行数据处理，整理出用户和商品的Python字典对象，其中用户作为key值，该用户已购买的商品数组作为value值，遍历所有相关用户的信息，得到Python字典对象。

第三步是计算指定用户与其他用户之间的相似度。在第二步处理完数据之后，对指定的用户进行与第一步相同的处理，得到一组用户—商品的key—value数据。取出需要计算的两个用户的信息，遍历两个用户对应的商品数组，当某个商品出现在其中一个用户的已购买商品数组中，却没有出现在另一个用户的已购买数组中时，这两个用户之间的距离加一，遍历所有的商品，计算出这两个用户的距离。遍历所有的其他用户，对他们和指定的某个用户重复执行以上的操作，就可以得到该指定用户与其他所有用户之间的距离。从定义中可以看出，用户之间的相似度与距离是成负相关的，取距离的倒数作为相似度，对所有的距离取倒数，就可以得到该指定用户与其他所有用户之间的相似度。

第四步是进行推荐。第三步中已经得到了该用户与其他用户之间的相似度，将其他用户按照相似度的大小进行排序。取出与用户相似度较高几个用户的信息，并将这几位用户的已购商品与指定用户的已购商品进行比较，并向该用户推荐未出现在其已购商品数组中却出现在其他几位用户的已购商品数组中的商品。

通俗的来说协同过滤算法就是利用了“物以类聚，人以群分”这个特性，利用具有相同兴趣、相似经历的群体的喜好来向用户推荐他们感兴趣的东西。协同过滤算法使用了其他人的经验，很大程度上解决了内容分析不够完全和不够准确的问题，而且能够通过一些难以表述的或复杂的概念对数据进行过滤。另外，协同过滤还可以根据用户对于之前的推荐的反馈信息来改变算法的推荐策略，进一步提高推荐的精度。当用户数据比较少的时候，推荐信息可能会不太准确，但是在推荐的过程中，推荐策略可以得到不断调整，用户库也得到不断丰富，之后的推荐准确度会越来越高。

## 利用协同过滤算法检测数据库表错误

在本系统中，不存在用户和商品的关系，也不存在用户群体这一概念，所以无法将协同过滤算法直接应用到本系统中。本系统换了一种思路，将套餐视为用户，将套餐中设计到的数据库表视为商品，这样就可以将用户—商品的模式应用到套餐—数据库表的应用场景中，向用户推荐商品对应于给套餐找出有问题的数据库表，其他用户对应于历史套餐。本小节中将协同过滤算法通过python语言实现，并将其服务化与工具化，方便服务器的调用。

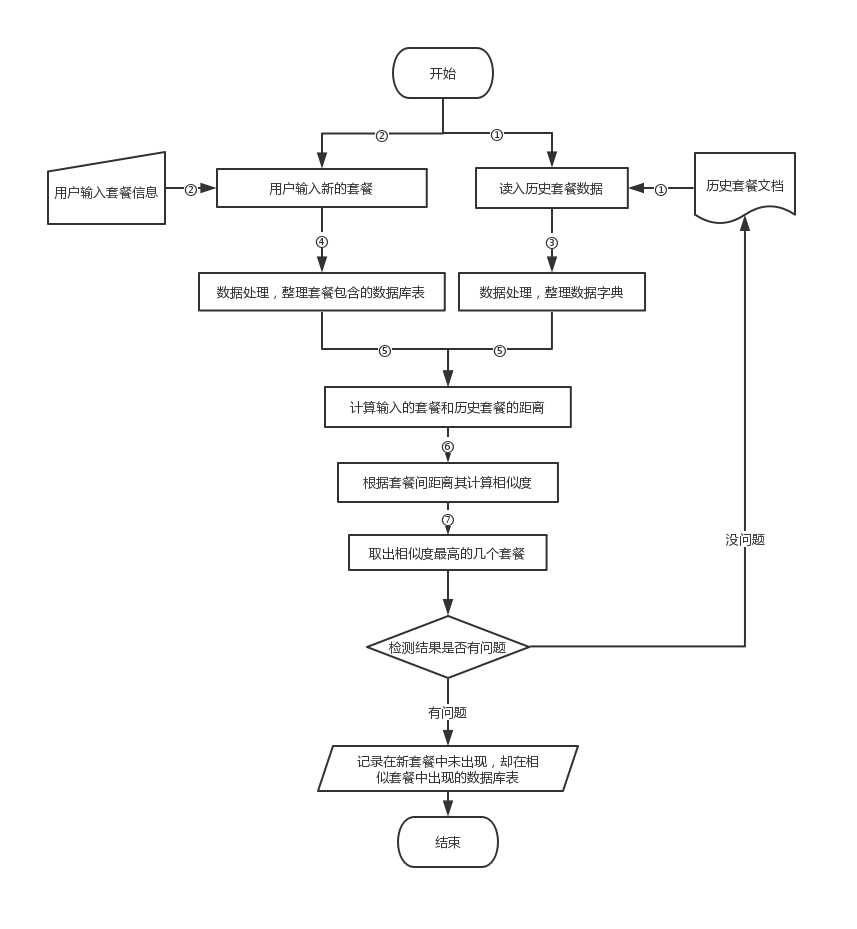


图3.2 协同过滤算法应用过程

协同过滤算法在本系统中的应用过程如图3.2所示。首先从历史套餐中读取数据到系统中，通过Python编写的处理函数进行数据处理，整理、格式化大量的历史套餐，将每个套餐的ID作为Python字典对象的key值，套餐中所涉及到的数据库表名列表作为value值。遍历所有的历史套餐，从历史套餐中整理出一份Python字典对象。由于本系统中通过协同过滤算法实现定位错误是以MVC架构中服务的形式实现的，所以用户的输入是通过http请求的方式进行，相应的Controller对用户的请求进行处理分析，并将新的套餐作为参数传到本节中所介绍的算法中。

通过协过滤算法实现的服务对传入的数据进行处理，与处理历史套餐的过程类似，得到新建套餐的ID和包含的数据库表的数组。

处理完新套餐数据和历史套餐数据之后，将历史套餐的数据逐一与新套餐的数据比较，当一个数据库表出现在其中一个套餐中却没有出现在另一个套餐中时，这两个套餐的距离便加一，遍历两个套餐包含的所有数据库表后，就可以得到这两个套餐间的距离。考虑到套餐间距离与它们之间的相似度成反比，本系统将距离取倒数作为量化两个套餐间的相似度的标准。重复以上操作，得到新套餐与所有历史套餐间的相似度。

通过上面的计算得到了所有历史套餐与新建套餐间的相似度关系，将这些历史套餐按照相似度的大小进行排序。按照需求取出相似度最高的几个套餐的信息，如果需要推荐精度高一些，就多取几个套餐的信息，但是这样运算的复杂度就会高一点；反之则可以少取几个套餐的信息，减少运算的复杂度。如果只取一个历史套餐的信息，则可以比较取出的这个历史套餐的信息与新创建套餐的信息，则需要记录下新创建的套餐没有包含，而这个历史套餐却包含的数据库表。如果取了多个历史套餐的信息，则需要按照不同的需求使用不同的策略，可以取这几个历史套餐所包含的数据库表的交集或并集，然后再跟新建套餐所包含的数据库表比较，记录下新创建的套餐没有包含，而这些历史套餐的交集或并集却包含的数据库表，如果采用的策略是取并集，则SQL的检测结果覆盖面会比较广，会检测出更多的错误，自然也会包含更多的无效结果；如果采用的是交集策略，则SQL的检测结果会减少很多，无效结果也会减少很多，可是交集策略也会导致该检测出来的错误未被检测出来，其中无效结果是指某个数据库表实际上未发生错误，但是扫描结果却显示该数据库表出错。另外，还可以根据在相似套餐中出现的次数来记录预警级别，本系统中所使用的分类标准如下：当某个数据库表在9个或9个以上相似的套餐中出现却没有在待检测套餐中出现或者在新套餐中出现，或者在新套餐中出现却没有在2个以上的相似历史套餐中出现，则将该数据库表标记为高危错误；当某个数据库表在7个至9个相似的套餐中出现却没有在待检测的套餐中出现，或者在新套餐中出现却没有在5个以上的相似套餐中出现，则将该数据库表标记为中危错误；当某个数据库表在5个至7个相似的套餐中出现却没有在待检测的套餐中出现，或者在新套餐中出现，却没有在7个以上的相似套餐中出现，则将该数据库表标记为低危错误。

如图3.2所示，新创建的套餐的检测结果如果没有问题则会被加入到历史套餐中，用以丰富历史套餐库。当检测结果有问题时，则会将本次扫描检测发现的可能存在的错误存进数据库中，提交测试的用户点击查看报告时，本系统中生成报告的服务就会获取数据库中的错误信息，将其生成报告，以网页的形式展现给用户。

# 利用关联规则挖掘算法检测错误

## 关联规则挖掘算法简介

关联规则反映了一个事物与其他事物之间的关联性和依存性，是数据挖掘中的一个重要技术，可以从大量数据中挖掘出有价值的数据项之间的相互关系，常用于从数据库中关联分析出“某个事件发生而导致其他事件发生”这样的规则。反映这一规则的经典例子是“尿布与啤酒”：当超市把尿布和啤酒放在一起时，这两样商品的销量都大幅增加。因为超市发现了这两样商品之间的联系，很多男性在为孩子买尿布时都会买瓶啤酒来犒劳自己。关联规则挖掘算法的目的是为了生成频繁项集和发现关联规则，常用的频繁项集的评估主要通过支持度去实现，而关联规则的计算则需要利用频繁项集，并以置信度作为评估标准。之前一章所描述的错误检测算法利用协同过滤算法实现的，而协同过滤算法本质上是通过分析与待处理的数据相似的一组数据，并对待处理数据的结果进行预测，其准确度无法达到百分之百。为了提高检测的准确度，本系统采用关联规则挖掘算法配合协同过滤算法共同检测SQL中的错误。

## 相关概念介绍

### 支持度

支持度是指几个关联的数据在数据集中同时出现的次数占总数据集的比例，即项集{X,Y}在总项集里出现的概率，X和Y同时在总数据集I 中发生的概率，公式为：

Support(X→Y) = P(X,Y) / P(I) = P(X∩Y) / P(I) = Num(X∩Y) / Num(I)

其中，I是总数据集，Num(X)表示X在总数据集中出现的次数。

### 最小支持度

最小支持度是算法开始前手动指定的一个阈值，一般情况下会根据算法的结果进行相应的调整。在算法执行过程中会计算各项的支持度，判断各项是否为频繁项的标准就是是否大于或等于最小支持度。

### 置信度

置信度是指一个数据出现后另一个数据出现的概率，或者说是数据的条件概率，即在先决条件X发生的情况下，由关联规则“X→Y”推出Y的概率。表示在发生X的项集中，同时会发生Y的可能性，即X和Y同时发生的个数占仅仅X发生个数的比例，公式为：

Confidence(X→Y) = P(Y|X) = P(X,Y) / P(X) = P(X∩Y) / P(X)= Num(X∩Y) / Num(X)

其中，Num(X)表示X在总数据集中出现的次数。

### 最小置信度

最小置信度类似于最小支持度，是一个需要手动指定的阈值，并需要在算法执行过程中不断调整优化。在得到频繁项集后需要对频繁项集进行处理得到置信度关系，当置信度低于最小置信度时，将不会将该条记录存放到结果中。

### 频繁项集

关联规则挖掘的过程中需要量化事件间的关联程度，常用的衡量标准是这些事件在数据集出现的次数。频繁项集是指支持度大于或等于最小支持度的所有项的集合，其中最小支持度为指定的常数，取值在0到1之间。

## 关联规则挖掘算法基本步骤

图4.1 频繁集生成算法示例

使用图4.1中所示的共9条数据进行介绍，首先需要指定最小支持度为0.2，计算每个数据在总数据集所占的比例作为该数据的支持度，可以看出图4.1中的数据共涉及P1、P2、P3、P4和P5共五种数据，这五种数据出现的次数如图4.1中C1表所示，将它们出现的次数除以总数得到对应项的支持度，保留支持度不小于最小支持度的项，如图4.1中通过C1得到计算并比较复杂度之后可以得到L1中的结果，并对L1中的结果进行连接得到如图4.1中C2的新的项集，并分别计算各项在原始数据中出现的次数。并将它们出现的次数除以总数，可以得到各项的支持度，保留支持度不小于最小支持度的项，从图4.1中的L2可以看出，删除了C2中的{P1，P4}等支持度小于最小支持度的项。对满足最小支持度的项进行连接操作，并计算连接操作之后所有项的支持度，保留所有满足最小支持度的项，并重复以上操作，直至所有的项都不满足最小支持度为止。其中连接操作是指两个包含n个元素，且前n-1个元素都相同的项，将其中一个项的第n个元素添加到另一个项中，得到一个新的包含n+1个元素的项。

记录下整个操作过程中所有满足最小支持度的项，分别计算各项的各种组合的置信度，例如项{P1，P2}需要计算P1=>P2和P2=>P1的置信度。得到了各项的置信度，也就得到了各项中元素之间的关联关系。

## 利用关联规则挖掘算法检测关键属性错误

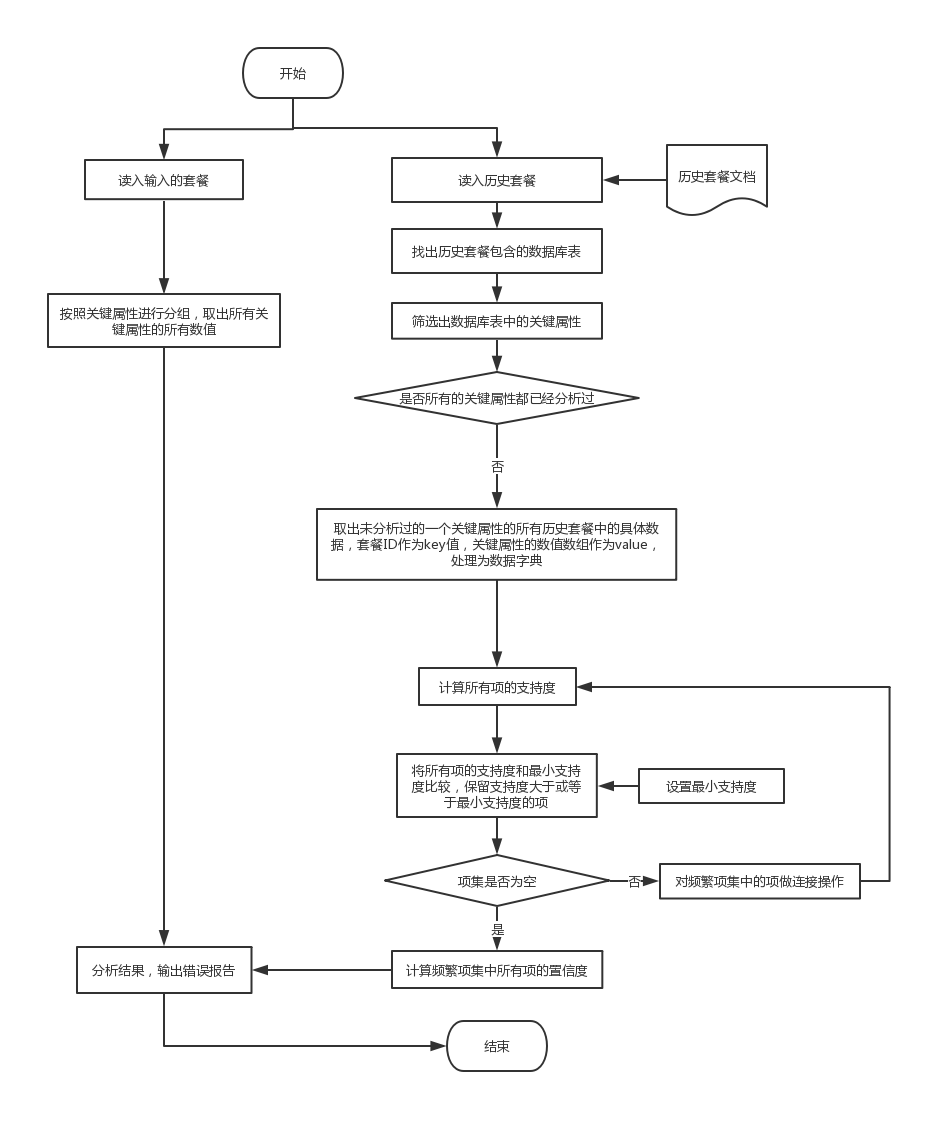
本文中的关键属性是指套餐创建过程中可能设计到的某个数据库表中的属性。一般来说，关键属性可以是每个数据库表中的外键，也可以是一些与数据库表有某些约束关系的属性。例如某个数据库表的功能是记录套餐间的关系，则其中的套餐ID和关系类型都可以看作关键属性。检测关键属性错误的过程如图4.2所示。

图4.2 检测关键属性错误过程

与协同过滤算法在本系统中的应用类似，本系统中并没有用户和商品的概念，所以要在思路上进行转变。与协同过滤算法不同，协同过滤算法检测的层次是套餐和数据库表之间，而通过关联规则挖掘算法检测的层次是套餐涉及的数据库表和关键属性之间，基于关键属进行关联规则挖掘，将套餐内涉及到的一个数据库表视为一个事件，将每个数据库表中的关键属性视为交易的商品。与利用协同过滤算法定位错误的数据库表相同，本节利用关联规则挖掘算法检测关键属性错误的功能也会封装成服务，方便服务器的调用。

本系统并非是在新套餐创建时才进行历史套餐分析的，因为利用关联规则基于关键属性检测错误运算量很大，运算时间也很长，所以本系统设计将在闲时运算关联规则挖掘服务，并将分析得到的支持度和置信度保存在数据库中，当检测新套餐时可以直接从数据库中取数据进行比对，并将处理结果生成错误报告反馈给前端，改善用户体验的同时，也可以提高系统效率。

检测关键属性错误的过程如图4.2所示，首先会读取历史套餐的数据，分析历史套餐会涉及的数据库表中的关键属性，其中包括“商品间关系表”中的“关联商品ID”属性和“商品发布地市表”中的“城市ID”属性等。将每个套餐中的每个关键属性的具体的数据都整理出来，以关键属性为基础作为分类标准将这些数据分成不同的Python字典对象，Python字典对象中的每个元素都是套餐ID与该套餐中这个关键属性的所有具体数值组成的数组。

获得了Python字典对象以后，对这些对象进行遍历分析，取出当前分析的Python字典对象所有的value值所包含的元素并去重，各自作为一个数据项，这些数据项类似于图4.1中的原始数据。计算这些项在数据集中出现的次数，并计算其支持度，将这些项的支持度与设定的最小支持度进行比较，保留支持度大于或等于最小支持度的项，删除支持度小于最小支持度的项，因为包含该项的其他项的支持度必然会小于最小支持度。将支持度大于或等于最小支持度的项添加到频繁项集中，并将这些项进行连接操作后进行下一轮的运算。对连接操作后的项重新计算其支持度，并与最小支持度比较，小于最小支持度的项将被删除，满足最小支持度的项会被添加到频繁项集中并进行连接操作，进入新的一轮运算。重复以上操作，直至没有项的支持度大于或等于最小支持度。

经过以上的操作可以得到套餐和关键属性的Python字典对的频繁项集。对频繁项集中各项进行处理，将频繁项集中各项按照蕴含式的形式进行调整（即p->q的形式），每项都对应着不同的蕴含式，例如项{A,B}可以生成A->B和B->A两个蕴含式，项{A,B,C}可以生成A->BC、AB->C、AC->B等六个不同的蕴含式，需要对这些不同的蕴含式分别计算置信度，将这些蕴含式的置信度与最小置信度进行比较，满足最小置信度蕴含式将会被加入到最后结果集合中，不满足最小置信度的蕴含式则会被弃置。通过这一步的处理可以得到满足最小置信度所有的蕴含式，这些蕴含式即是关联规则。

需要利用关联规则挖掘算法检测套餐时，则根据数据库中的置信度信息，进行检测。例如“商品间关系表”中的“关联商品ID”属性生成的置信度表保存了Confidence(“130000000653”=>“130000000654”)=1的信息，而当待检测套餐“商品间关系表”中有“关联商品ID”属性值有“130000000653”的记录却没有“关联商品ID”属性值有“130000000654”的记录，则将该情况记录为关键属性错误，可以根据该蕴含式的置信度大小进行记录，本系统中使用的分类标准如下：当置信度大于0.9时记录为高危错误，置信度处于0.7至0.9之间时记录为中危错误，置信度低于0.7则记录为低危错误，并生成错误报告反馈给前端。

# 实验成果

## 实现效果描述

本文主要针对中国移动（海南）创建套餐容易错误的问题实现了SQL扫描检测的功能。其中系统主要通过Django框架搭建，扫描检测和预测SQL中的错误主要通过协同过滤算法和关联规则挖掘算法等实现。本文实现的系统可以利用历史套餐创建模型，对用户上传的套餐的SQL文件进行扫描分析，对可能出错的地方进行预警，并生成错误报告，通过网页的形式展示给用户。

## 系统展示

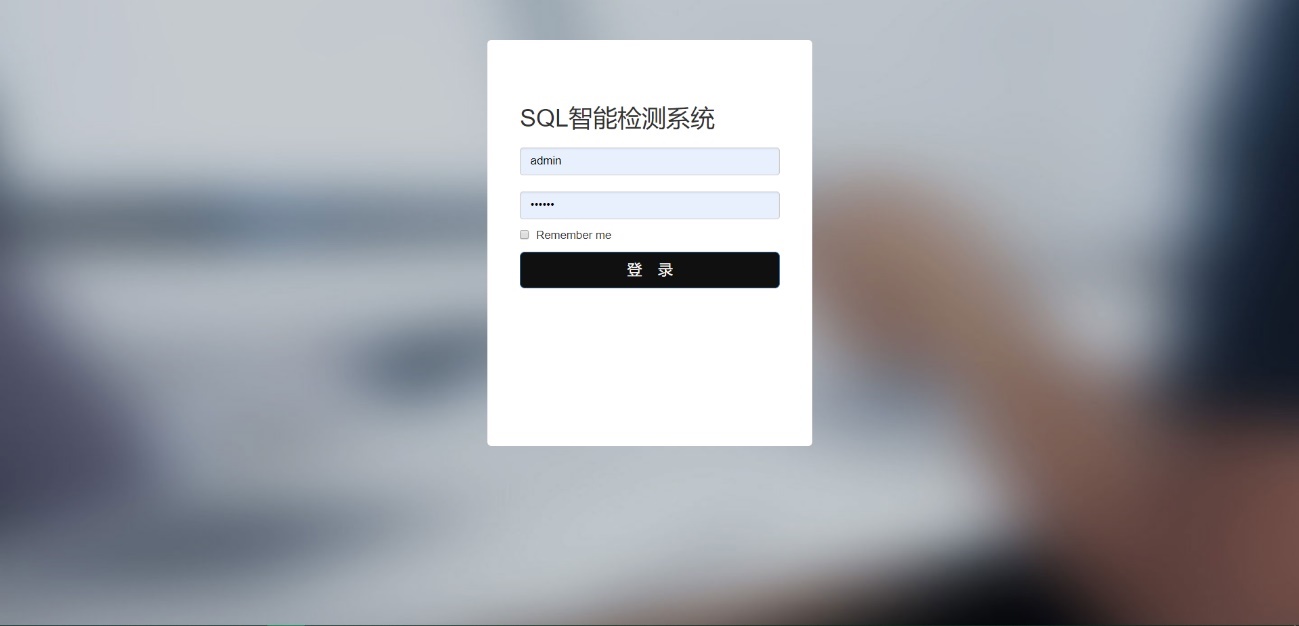
登录页面如图5.1所示。

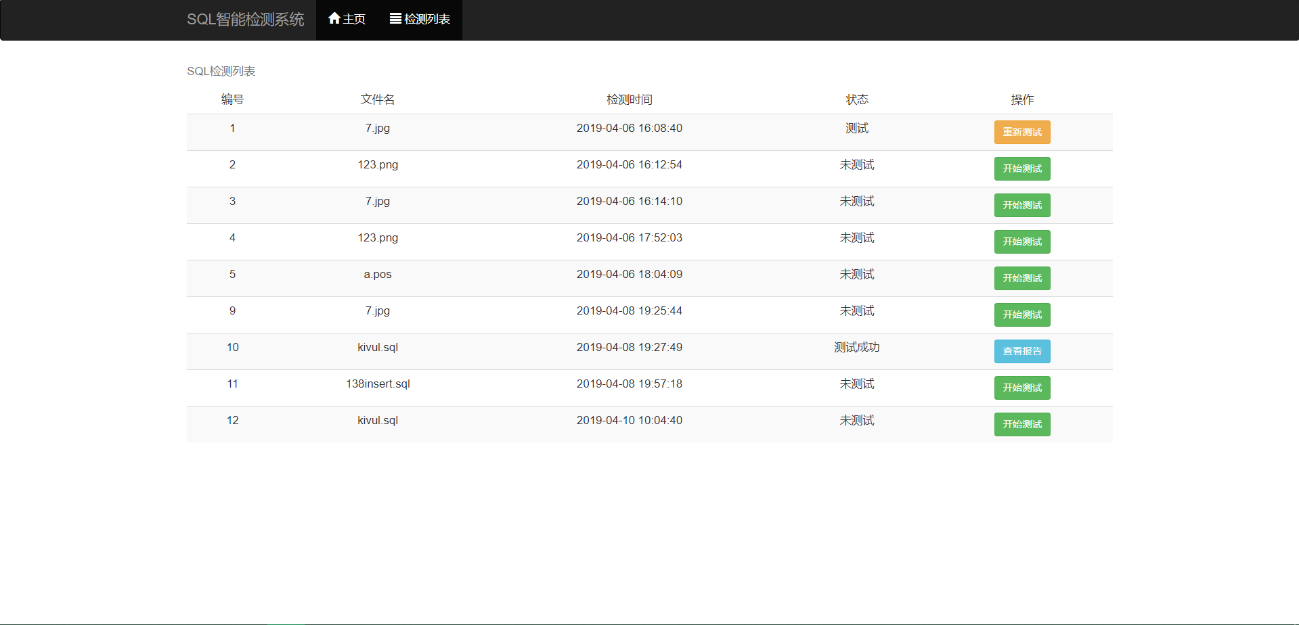
图5.1 登录页面

登录成功后可在主页上传需要检测的文件，上传文件前后主页如图5.2和图5.3所示。

图5.2 登录后显示的主页

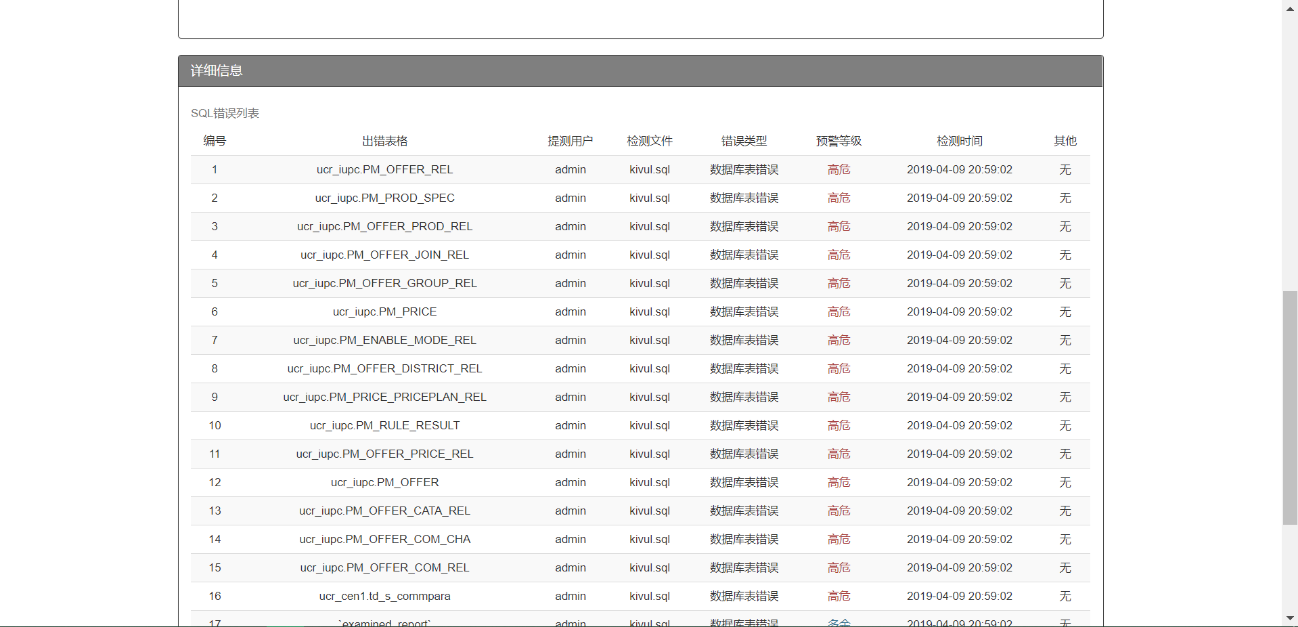
图5.3 文件上传成功后的主页

文件上传成功后，系统会根据当前上传的文件建立一个检测任务，用户可以在检测列表页面查看根据自己上传的文件所创建的任务，并可以对任务进行错误重测、开始测试、查看报告等操作，检测列表页面如图5.4所示。

图5.4 检测列表页面

点击开始测试后系统会对该任务对应的文件进行检测，并生成测试报告，如图5.5和图5.6所示。

图5.5 测试报告页面的基本信息和统计信息

图5.6 测试报告页面的详细信息

## 结果分析

检测结果如上节中成果图所示，从检测结果来看，当历史套餐数量较多时，且与新创建的套餐较为类似的历史套餐比较多时，检测结果较为准确；当历史套餐数量较低时，会有较多的正常数据被当成错误数据进行预警，也会有部分错误未被系统发现，准确率较低。

# 总结与展望

本文通过协同过滤算法和关联规则挖掘算法来实现对中国移动（海南）套餐创建过程中的错误检测，即对SQL语句的错误检测。为了提高错误检测的准确率，本文实现的系统中同时利用协同过滤算法和关联规则算法进行错误预测。

本文所实现的系统通过Django框架搭建，系统使用B/S 模式设计，提高了系统的易用性，简化了系统的开发、维护和使用。本文使用以上两种数据挖掘算法进行预测，利用协同过滤算法预测创建套餐的SQL语句中是否遗漏或多余某些数据库表，利用关联规则挖掘算法预测创建套餐的SQL语句中的关键属性值是否存在错误。两个算法都是通过Python语言实现。

虽然两个算法可以通过调整最小支持度和最小置信度等参数提高预测的准确性，但是两个算法本质上都是通过对历史数据的分析，进而对结果进行预测，因此准确性无法达到百分之百。利用关联规则挖掘算法分析SQL语句中的关键属性时，以及使用协同过滤算法分析数据库表时，由于关键属性和套餐包含的数据库表较多，导致程序执行时间较长，所以下一步工作是对这两个算法进行优化，使用多线程或者分布式的方法并行处理加快处理速度。

参考文献

[1]黄永祥.玩转Django 2.0[M]:清华大学出版社，2018.

[2]Jeff Forcier,Paul Bissex,Wesley Chun.Django Web开发指南[M].机械工业出版社，2009.

[3]王友钊，黄静.Django开发宝典[M].清华大学出版社，2017.

[4]齐伟.跟老齐学Python,Django实战[M].电子工业出版社，2017.

[5]齐伟.跟老齐学Python,数据分析[M].电子工业出版社，2018.

[6]张云飞,杨明光.基于Django的RESTful通用程序接口研究与实践[J].电脑知识与技术,2018,14(28):30-32.

[7]周志华.机器学习[M].清华大学出版社，2016.

[8]Peter Harrington.机器学习实战[M].人民邮电出版社，2013.

[9]邓亚文,罗可.一种基于用户和物品相似度的融合协同过滤推荐算法[J].电脑与信息技术,2019,27(01):6-10+33.

[10]陶志勇,崔新新.一种融合用户与项目属性的协同过滤算法的设计与实现[J].计算机应用与软件,2019(02):12-18+102.

[11]陈垲冰,黄荣,吴明芬,刘兴林.一种基于电影评分预测的协同过滤[J].哈尔滨师范大学自然科学学报,2018,34(06):1-5+11.

[12]吴经纬.协同过滤算法的研究[J].电脑知识与技术,2019,15(03):20-21.

[13]张双庆.一种基于用户的协同过滤推荐算法[J].电脑知识与技术,2019,15(01):19-21.

[14]周泽宇,王春玲.基于协同过滤的个性化选课推荐与评论系统[J].信息记录材料,2018,19(10):156-159.

[15]王晓丽,奚克敏,刘占波,闫实.基于Apriori算法的关联规则分析[J].软件,2019,40(02):23-26.

[16]史月美.关联规则挖掘研究[M].兵器工业出版社，2016.

[17]林甲祥,巫建伟,陈崇成,张泽均,舒兆港.支持度和置信度自适应的关联规则挖掘[J].计算机工程与设计,2018,39(12):3746-3754.

[18]倪东.基于数据挖掘的关联规则研究[J].太原学院学报(自然科学版),2018,36(03):36-39.

[19]葛璐瑶.关联规则挖掘Apriori算法应用研究[J].计算机产品与流通,2018(08):185.

[20]吴小东,曾玉珠.基于Apriori算法的高校学生成绩数据挖掘[J].廊坊师范学院学报(自然科学版),2019,19(01):31-36.

[21]李晓龙,冯俊文.关联规则频繁项集挖掘算法设计与实现[J].特区经济,2018(08):111-114.

[22]李忠,安建琴,刘海军,宋奕瑶.关联挖掘算法及发展趋势[J].智能计算机与应用,2017,7(05):22-25.

致谢

四年前刚开学时的场景还历历在目，在武大的求学生活竟然已悄悄落幕。致谢是本篇论文的最后一个部分，原以为是最好写的地方，却一直难以下笔。我已经忘记了四年前的自己为什么要选择计算机专业，也许是被科幻片中那些黑客随便敲打几下键盘就可以随心所欲的操纵整个世界的酷炫场景所吸引吧，怀着这样的憧憬走进校门，直到听到老师讲程序员还有个别称叫“码农”，才发现这个专业跟自己想象中的相去甚远，随便敲几行代码根本黑不进别人家的系统，只会弹出编译错误。可是当第一次看到自己的代码在屏幕上输出“HelloWorld”时，当第一次解决bug时，当第一次独立的实现一个界面丑陋不堪、代码破绽百出的系统时，那种油然而生的喜悦感仿佛都在跟我说，当年没有选错专业。

首先我要感谢武汉大学提供的学习平台。在武大的四年里，当年那个对算法和编程一无所知的我，在计算机专业科学的课程设置和充足的教学资源的培养下，正逐渐的成长为一个“挥舞着键盘和本子，发誓要把世界写个明明白白”的准程序员。借此机会，我谨向武汉大学表示衷心的感谢，感激四年的培育之恩。

其次我要对我的校内导师喻丹丹副教授和校外导师陈振宇教授表示感谢。毕业设计是对大学四年学习生活的检测，更是对自己这四年有没有虚度光阴的一次扪心自问。从保研之后，我就在考虑毕业设计要做什么，恰逢陈振宇老师与中国移动（海南）进行合作一起解决创建新套餐需要耗费巨大的人力资源且容易出错的问题并且把这个任务交给了我，毕业设计做什么的问题得以解决。从开题到现在，解决方案更换了多次，有些方案因为效果不好被弃置，有些方案因为难度太高无法实现被弃置，经过了很长时间的考虑之后，最终在陈振宇老师的帮助下选择了使用协同过滤算法和关联规则挖掘算法的方案。此外在论文撰写过程中以及在算法的实现过程中，老师们也提供了很多的帮助，这次毕业设计能够顺利完成，两个老师的大力支持和帮助必不可少，由于在毕业设计开始之前我对协同过滤算法和关联规则分析算法了解的并不是很多，而我的校外导师陈振宇教授安排了学长为我答疑解惑，为我的毕业设计清除了很多障碍，感恩之情难以言喻。另外，在开题之后，喻丹丹副教授和陈振宇教授一直在督促我的毕业设计系统实现和论文进展，促使我的毕设工作能够一直按计划进行着，因此我要对我的校内导师喻丹丹副教授和校外导师陈振宇教授表示真挚的谢意，感谢老师们的无私的奉献精神以及严谨的治学态度。

此外，还要对武汉大学计算机学院所有的老师们表示感谢，是老师们的谆谆教诲，让我们能够更好的完成学业，是老师们的悉心培养和教导，让我们能够不断开阔自己的视野，丰富自己的见识。

我还要感谢为我的毕业设计工作提供了帮助的同学们和学长们，是你们的帮助和指导使我的毕业设计工作能够顺利进行，非常感谢你们的帮助！

最后，我还要对百忙之中能够拨冗审阅本文的各位老师表示感谢，你们辛苦了。