**大连理工大学本科毕业设计（论文）**

**电商数据获取及分析系统设计与实现**

**Design and Implementation of XXX**

学 院（系）： 软件学院

专 业： 软件工程

学 生 姓 名： 黄 鹏

学 号： 201192498

指 导 教 师： 单世民

评 阅 教 师： X X

完 成 日 期： 2015年6月5日

大连理工大学

Dalian University of Technology

摘 要

随着领先的在线旅游企业建立自己的旅行社和目的地接待体系，以及传统的旅行社逐渐的建立自己的网络平台，网络平台和旅行社呈现相互融合的趋势。业内人士认为，集合了网络信息技术优势与旅行社专业服务的在线旅行社(Online Travel Agent，简称OTA)，将是旅行电子商务的主流和未来。由于在线旅行发展正值发展初期，为了抢夺市场，多家OTA之间的战争是越演激烈，尤其突出在抢夺酒店用户这一领域上。每家OTA为了扩充自己的酒店数据，定期到竞争对手站点抓取酒店数据，然后通过酒店数据去重算法从抓取的酒店数据中筛选出新增的酒店数据，并导入本地数据库。迫于扩展酒店数据的需求，本文实现了电商数据获取及分析系统。

本文应用Django，Scrapy等框架，涉及KDTree，Shingling和LCS等算法，采用B/S结构构造系统构架，使用MySQL作为后台数据库，以Django为应用服务器开发实现了电商数据获取及分析系统。设计开发的电商数据获取及分析系统共包含Spider抓取，数据展示，导入新增酒店数据三个模块。本文对电商数据获取及分析系统开发过程中的需求分析，概要设计，详细设计以及测试的各个阶段进行了较详细的说明和讨论，并对Spider抓取模块和导入新增酒店数据模块进行了重点阐述。

本文设计与实现的电商数据获取及分析系统使用了目前最流行的Python语言，基于Linux作为开发和运行平台，经过具体的编码实现与系统测试，电商数据获取及分析系统中的各个模块均已成功实现，并已投入使用，且运行良好。

关键词：抓取酒店数据；Django；Scrapy；去重算法

Design and Implementation of XXX

Abstract

目 录

1 绪论

## 1.1 课题研究背景

互联网技术的飞速发展，让人们的出行越来越便捷。过去，我们需要到火车站或代售点预定火车票，到航空公司或代售点预定机票，到酒店预定房间，到旅行社团购旅行。如今，人们只需要通过Web或移动客户端就可以轻松实现网上订票和预定酒店，一场说走就走的旅行从此不再是梦想。

由于OTA集合了网络信息技术优势与旅行社的专业服务，极大地方便了人们的出行，所以越来越受到人们的青睐，市场的发展前景极佳。2014年北京APEC会议提出了未来五大热门的行业，其中就包括在线旅行行业。有市场的地方就必然有竞争，由于在线旅游行业正值发展初期，为了抢夺市场，多家OTA之间的战争是越演激烈，尤其突出在抢夺酒店用户这一领域上。每家OTA为了更好地发展自己的线上酒店产品，让自己在酒店竞争中获得先机，所以扩充本地数据库中酒店数据的工作刻不容缓。

## 1.2 课题研究现状

由于在线旅行行业正处于发展阶段，抢夺市场是各个OTA的首要目标。各个OTA在发展自己业务的同时，在条件的允许下会成立Spider小组搭建Spider框架，负责抓取工作。

由于涉及公司隐私，各个OTA使用的Spider技术和酒店数据分析的去重算法我们都无法具体得知，但是广泛使用的Spider框架无非以下几种：

1. PHP的Goutte框架
2. Java的Heritrix框架
3. C#的NWebCrawler爬虫框架
4. Python的Scrapy框架

常用的字符串判重算法包括：

1. Shingling
2. Longest Common Subsequence
3. Jaccard
4. Simhash

以上的Spider框架和字符串判重算法已经比较成熟，但是将Spider和Django框架结合使用的例子还是比较少见。由于涉及公司隐私，对其余OTA的抓取和数据分析工作，我们无法获得比较详细的信息，不过在艺龙，酒店数据静态抓取的工作还从未开展过。

## 1.3 课题研究意义

虽然前部分所描述的技术均为当下比较热门和常用的技术，但是充分利用且基于上述框架设计和开发的电商数据获取及分析系统仍比较少见。虽然在艺龙有专业的Spider小组，但是他们隶属于搜索组，不会负责扩展酒店数据的抓取。所以，设计与开发电商数据获取及分析系统有着非常重要的实际意义：

1. 抢夺酒店市场，扩展自己的线上酒店产品，让自己在酒店竞争中获得先机
2. 填补酒店产品组静态抓取的空白
3. 使用的技术Django，Scrapy以及酒店数据去重算法等技术十分前沿，使用Python作为主要开发语言，使得系统轻便简洁，容易维护，易于扩展。结合使用KDTree，Shingling，LCS等算法，设计出了一种高效的酒店数据去重算法

## 1.4 论文组织结构

本文主要内容是电商数据获取及分析系统设计与实现。在文中对当今的互联网部分新技术进行了介绍与分析，选取Django，Scrapy作为框架，通过Spider抓取和酒店数据去重算法对系统进行实现。完成的主要工作包括：Spider抓取，数据展示，酒店数据去重算法设计，服务器搭建，数据库设计等工作，实现了酒店静态信息抓取及展示，发送抓取结果邮件，导入新增酒店数据到本地数据库等功能。

论文的主要结构如下：

第一章：绪论。介绍了课题的研究背景，现状以及意义。简介的分析了与课题相关的技术与发展趋势。

第二章：电商数据获取及分析系统相关技术介绍。详细介绍了可用于项目开发的相关技术以及特点。

第三章：电商数据获取及分析系统需求分析，根据用户对系统进行功能，业务用例，数据流图等方面的设计。

第四章：电商数据获取及分析系统的系统设计。对系统进行详细的分析与设计，包括系统总体架构，数据库设计，Spider设计，去重算法设计等进行了明确的定义和说明，同时进行了数据模型和系统中类，属性的定义。

第五章：电商数据获取及分析系统的系统实现。对系统的搭建环境进行了说明，并对Spider抓取流程和去重算法进行了分析。

第六章：系统测试。结合系统运行日志和测试方案明确测试用例，并且对测试的结果进行分析。

结论部分对整个系统开发工作进行了总结，并对系统可进一步完善的部分进行了展望。

2 相关技术介绍

本章将详细介绍系统设计所需核心技术，通过对技术的介绍与分析为后文实际开发提供坚实的理论支持。

## 2.1 B/S结构和C/S架构

B/S结构即众所周知的Browser/Server结构，客户端使用浏览器（Browser）进行浏览，浏览器负责与服务器端的交互工作。由于浏览器的标准化工作相对完善，因此客户端的压力比较小，核心的逻辑业务都由服务器端处理。这种模式使用统一的客户端，简化了开发、使用以及维护的过程。由于现在任何操作系统都配有浏览器，B/S的普适性也有很大的优势。

相较于传统的C/S（Client/Server）结构， B/S结构有很多优点，其分布性强，便于共享，便于扩展，兼容性强，容易快速安装配置[n]。

[樊银亭, 何鸿云. 基于客户机/服务器体系的二层与三层结构研究[J].计算机应用研究,2001,18(12):25-26.]

但同时B/S也有着安全性相对差，复杂功能实现较困难等缺点。但是随着开发编程语言的发展，这些缺点也在逐渐改善中。B/S与C/S更为深入的比较可以入表2.1所示：

表2.1 B/S模式与C/S模式比较

Tab. 2.1 Comparison between B/S mode and C/S mode

|  |  |  |
| --- | --- | --- |
| 项目 | B/S | C/S |
| 硬件环境 | 广域网 | 专用、小范围网络 |
| 安全性 | 对安全控制能力弱 | 对信息控制力强 |
| 程序架构 | 全面支持网络的构建系统 | 注重流程控制 |
| 软件重用性 | 多重结构，重用性强 | 受整体性限制，重用性弱 |
| 系统维护 | 可以实现无缝升级 | 升级变化数据较难 |
| 处理问题 | 分散地域，操作系统无关 | 相同区域，操作系统相同 |
| 信息流 | 可变化的数据流处理 | 中央集权式的机械处理 |

不断发展的分布式计算技术对于分布式的要求非常高，在适应分布式服务中，B/S相较于C/S有很大的优势。分布式计算对分布性的要求很高，强大的并行以及运算能力使得云端服务器的性能十分卓越。对于分布式的系统而言，B/S比C/S更有优势。

B/S本质上是C/S的一种特殊表现形式，是对C/S这种模式的一种改进。对于本文使用的技术而言，B/S对于实现分布式和大规模的处理更有优势。

C/S架构就是我们经常说的客户端和服务器结构，这是一种典型的系统体系结构。使用这种结构时，我们可以充分利用两方的环境资源，来将任务分配到客户端与服务器端，这样系统的通讯开销就比较合理。这种体系结构比较广泛的应用在各种类型的操作系统中，C/S架构也经常被人们称为主从式架构。随着时间的发展，客户端的概念也在不断扩展，以前的客户端主要指的是在用户电脑端的软件，当今科技的发展更多的时候浏览器上的应用程序也往往被人们认为是分布式的客户端。C/S这种模式有一个特点是，客户端和服务器端经常会距离很远，应用程序将自己的请求提交给服务器程序。服务器程序处理了用户的请求后，再将服务器端的程序作为结果以特定的形式返回给用户。C/S产品具有很多的弱点，其中一个比较突出的弱点是，他的兼容性比较差。根据不同的系统，可能需要开发不同版本的软件。而且软件的更新换代通常比较快，这样的话每次更新客户端都需要付出很大的代价。因此这种缺点是人们在考虑使用这种模式的一大顾虑。

虽然C/S结构有很多缺点，它也有自己独有的优点，首先C/S结构的优点就是他可以很好的利用客户端的处理能力，先将比较繁复的工作在客户端处理之后，在将任务提交给服务器。这样可以使得客户端相应非常快，服务器运行数据的负载就非常低。此外，C/S架构中，存储和管理数据的功能非常透明。数据的存储和管理在数据库的应用中，是独立的运行于服务器程序和客户应用程序。有一些在前端不能违反的规定，就可以集中开发在服务器程序中。这样，前端的用户不用知道后台数据是如何实现的，便可以完成自己的工作。

对于本视频开发系统而言，C/S结构虽然众多优点，但是由于系统有兼容性和分布性的目标。在这几方面看起来B/S结构具有更大的优势，而且B/S结构的前端技术不断发展，对于显示视频流，与云端方便接入等方面也有着C/S结构所不能比拟的优势，因此，在这个视频推荐系统中，应该选择B/S模式作为本系统的开发模式。

## 2.2 Django Web开发框架

本小节主要介绍用于开发的框架Django。从Python语言、MVC设计模式以及Django的开发框架这三个方面进行了系统的介绍。

### 2.2.1 Python编程语言

Python是一种面向对象的解释性的应用程序开发语言，它既有C、Java等系统语言的普适性功能，也具有VB、Ruby等脚本语言的敏捷开发特性。Python语言简洁、清晰，可用类库强大、语法简单，非常适合程序开发人员进行快速开发工作。Python有一个被开发者们广泛称呼的别称：胶水语言，Python作为开发语言，可以将其他语言如（Java，C语言）编写的各种模块轻松的连接起来，或者运用一些逻辑关系，完成两个接口之间的链接。

就Python语言的主要特点而言，由于它本身是开源项目，因此与其他著名的开源项目有着类似的特点。具体说来，他的优点大概有以下方面[n]：

[司维, 曾军葳, 谭颖华. Python基础教程[M].北京:人民邮电出版社, 2010.]

移植性强：开源的本质决定了有大量的开发者将Python移植到多种平台之中。主流的Linux、Windows、Macintosh、Solaris等都已经支持Python语言。甚至一些微型系统入Palm OS、Pocket PC等也支持Python语言。

介于高层的编辑语言：Python良好的封装与设计使得开发者无需考虑程序本身如何操作内存等细节。

扩展性、嵌入型强：由于Python良好的兼容性，可以利用C/C++等程序语言，然后再Python程序中使用。

丰富的资源库：Python标准库涵盖到方方面面。可以处理正则表达式、单元测试、数据库、网页浏览器等很多与系统有关的工作。此外，社区还提供了很多标准库意外的高质量的库文件供用户使用。

面向对象：Python同时支持面向对象与面向过程的编程。开发者可以利用数据功能的组合，对象以及对象的属性操作来对程序进行操作控制。

强缩进控制模式：Python使用的强制缩进控制的方式，使用户可以清晰的读懂代码结构。虽然初学者对于这种格式可能比较不适应，但是有经验的开发者会从中受益很多，很大程度的提高了开发效率。

交互性解释器：Python中使用一种交互性解释器的工具，这种工具是Python开发中最有用的工具之一。开发者可以直接利用解释器来测试代码，解释器既可以检查代码的正确性，还能查看对应的数据结构、键值对等等。

Python拥有“明确，优雅，简单”的设计哲学。Python与经典的Perl语言截然不同，Python总是强调用一种方法解决问题就好了。Python这种完全面向对象的语言，因为它的这些突出特性正在被很多国内外机构广泛使用，有一些大规模的软件开发计划，例如Mnet、BitTorrent、Google等都有使用它。这些知名公司的使用，足以说明Python的强大以及高效。

Django本身是以Python写成，因此作为一个使用Django框架的开发者，我们最好能有一些Python的编程相关经验。在Django的开发中，高效、准确的利用Python会让你的开发工作事半功倍。

### 2.2.2 MVC设计模式

Django使用的是MVC设计模式，MVC就是Mode-View-Controller（模型-视图-控制器）的缩写。这种开发模式迫使应用程序的输入、处理过程和输出结果分离。使得三个核心部件各自处理自己的任务。下面简要介绍一下上述三个部件[n]：

[陆荣幸, 郁洲, 阮永良, 王志强. J2EE平台上MVC设计模式的研究与实现[J].计算机应用研究, 2003, 20(3):91-93.]

模型有时也被称为数据模型，它的功能是封装与要处理的逻辑业务相关的数据，并定义处理数据的方法。模型有着访问数据的权限，操作数据库等操作都在其功能范围内。模型并不关心数据显示与操作的实现，但一般有着监听机制来监视模型上注册的视图。如果视图数据改变，就会触发模型组件的相应操作。

视图组件是用来实现数据的显示的，视图组件通常根据数据显示的定义规定处理的逻辑。通过访问它监视的数据模型来刷新自己的功能。



图2.1 MVC设计模式

Fig. 2.1 MVC Design Mode

控制器组件是用来组织不同层面，控制应用程序处理的流程。它对用户的行为、数据模型的改变进行处理，用合理的方式来响应请求，数据库专家进行数据库处理和数据库设计也通常在此环节。图2.1描述了MVC的基本工作模式。

MVC的工作模式对于开发者来说有很多优点：首先它允许多个视图共享同一个模型。对于程序设计，尤其是Web程序开发而言，同一个应用程序通常会有不同的用户界面，用户希望通过浏览器来进行浏览，也希望通过移动终端来访问。上述问题就要求Web网站可以同时提供Internet和移动终端两种视图。由于MVC中，视图与模型分离，模型响应用户的请求并且返回数据，视图组件负责呈现给用户的外部显示。那么业务逻辑和外在表示就分离了，代码的重用性也就得到很大程度的提高。

由于控制器的独立，应用程序的业务规则或者数据源也得以方便的更改。举个简单的例子，例如系统打算换一种数据源，或者对数据库进行移植。我们可以通过简单的更改控制器来轻松的实现想要实现的数据源改变。而由于三个模块间的独立，我们对于控制器端的改变，视图端以及模型端不需要做任何改变。

除此之外，MVC还具有部署快、可维护性高、有利于工程化软件的管理等突出优点。

### 2.2.3 Django

Django是一个开源代码的Web应用框架，整个Django框架都是由上一章节所述的MVC模式设计的。Django的主要设计目标就是简化复杂的、数据库驱动的网站开发。Django自带丰富的开发组件可以使得开发人员以最少的代码，最快的速度，最方便的模式来进行各种Web应用的开发与调试。图2.2便是Django的架构总览[n]：

[徐旭铭. Django Web开发指南[M].北京：机械工业出版社, 2009.]

Django核心框架包括：用于数据模型和关联数据库间交互的面向对象的解释器，基于正则表达式的URL派发器，用于处理请求、显示的试图系统和一个模板系统。除此之外，Django还带有一个轻量级的Web服务器用于测试和开发，队请求处理各个阶段进行干涉的中间件，还有可以读取特殊类型XML或者JSON的序列化系统以及一个用于扩展模板引擎的系统。

在核心框架中，Django实现了MVC的设计理念[n]。

[刘班. 基于Django快速开发Web应用[J].电脑知识与技术, 2009,5(7):3-6.]

在实现MVC架构中，最关键的部分在于要正确的分割应用程序的不同层次。在Django对于MVC框架的分层中，改变了一些做法。首先在模型部分，Django缩小了模型部分的定义，Django框架中的模型只负责把数据传入数据库。对于视图部分而言，Django的视图并不单单具有显示数据的唯一功能，它的功能还带有MVC中控制器的一部分。Django开发团队曾说：“在我们理解的MVC里，视图的作用是对要展示给用户的数据进行描述，这种描述不但包括数据的外观，还包括数据的是应该如何被表现的。”也就是说，Django的表示曾丽，视图模型里定义了要对模板中的什么数据进行显示，而模板则定义了最终数据的显示方式。Django框架本身决定了视图与模板之间的交互，以及相应指定请求的机制。框架本身更复合MVC中控制器的功能。



图2.2 Django组件架构图

Fig. 2.2 Infrastructure of Django

在Django的核心框架中，与开发者最为相关的三个模块分别是模型、模板和视图。

模型：所有应用程序的基础，应用程序中，模型是最底部的一层。作为基础，模型通常都十分稳定，不同于视图和模板等可以用不同的表现形式替换。

模板：绝大多数Django开发者们习惯使用模板来渲染HTML页面，模板就是经过特殊格式化的可以输出动态HTML的文本。除此之外，模板还支持逻辑结构与循环。视图可以通过制定模板的方式，来显示所需要显示的信息，而且还可以使用到模板中渲染的结果。

视图：视图总是覆盖Django应用程序里几乎全部的逻辑。通常，视图的定义是链接到一个或者多个URL上的Python函数，用Http响应对象的方式返回。视图通常可以从模型里取出单个或者一列对象并将其显示，视图也可以向模型里添加这样的新对象。这是视图的主要功能。在Django中，你既可以自己编写一些函数来控制这个过程，也可以使用框架本身提供的帮主函数来进行开发。在视图这一层面上，Django的灵活性往往得到了充分的体现。

在Django框架中，MVC模式有时也被称为“MTV”，即上文所述的模型（Model），模板（Template），视图（View）。整个工作流程大致如下，受到的Http Requirement被Web服务器转发给Django框架，框架的中间件对他们接收。然后，URLconfig文件对他们分配，将其转入合理的视图，视图通过模板或者是模型生成相应的需求响应。在经过另外一层中间件，将HTTP的相应转发给Web服务器随即转给用户。

## 2.3 Scrapy Spider开发框架

本小节主要介绍Spider开发框架Scrapy。

如图2.3所示，显示了Scrapy的大体架构，其中包含了它的主要组件及系统的数据处理流程。

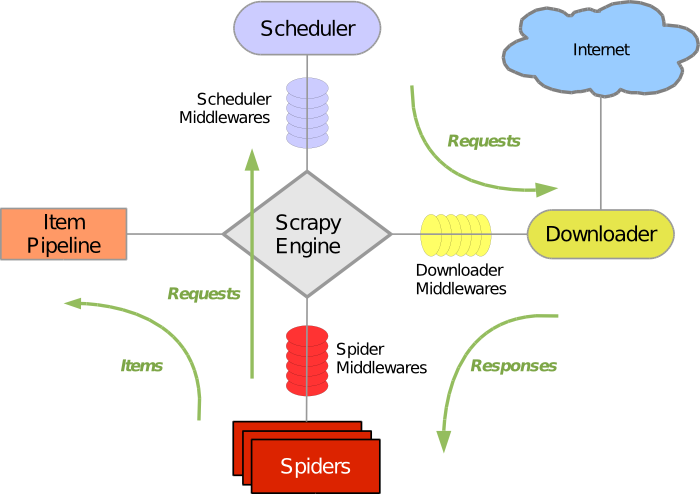


图2.3 Scrapy组件架构图

Fig. 2.3 Infrastructure of Scrapy

### 2.3.1 Scrapy组件

Scrapy Engine(Scrapy引擎)：Scrapy引擎是用来控制整个系统的数据处理流程，并进行事物处理的触发。

Scheduler(调度)：调度程序从Scrapy引擎接受请求并排序列入队列，并在Scrapy引擎发出请求后返还给他们。

Downloader(下载器)：下载器的主要职责是抓取网页并将网页内容返还给Spiders

Spider(蜘蛛)：Spiders是Scrapy用户自定义用来解析网页并抓取指定URL返回的内容的类，每个Spider都能处理一个域名或一组域名。Spider就是用来定义特定网站的抓取和解析规则。Spider的整个抓取流程如下：

1. 首先获取第一个URL的初始请求，当请求返回后调取一个回调函数。第一个请求是通过调用start\_requests()方法。该方法默认从start\_urls中的Url中生成请求，并执行解析来调用回调函数
2. 在回调函数中，你可以解析网页响应并返回项目对象和请求对象或两者的迭代。这些请求也将包含一个回调，然后被Scrapy下载，然后有指定的回调处理
3. 在回调函数中，你解析网站的内容，同程使用的是Xpath选择器（但是你也可以使用BeautifuSoup, lxml或其他任何你喜欢的程序），并生成解析的数据项
4. 最后，从蜘蛛返回的项目通常会进驻到项目管道

Item Pipeline(项目管道)：项目管道的主要责任是负责处理有蜘蛛从网页中抽取的项目，他的主要任务是清晰、验证和存储数据。当页面被蜘蛛解析后，将被发送到项目管道，并经过几个特定的次序处理数据。每个项目管道的组件都是有一个简单的方法组成的Python类。他们获取了项目并执行他们的方法，同时他们还需要确定的是是否需要在项目管道中继续执行下一步或是直接丢弃掉不处理。项目管道通常执行的过程有：

1. 清洗HTML数据
2. .验证解析到的数据（检查项目是否包含必要的字段）
3. 检查是否是重复数据（如果重复就删除）
4. 将解析到的数据存储到数据库中

Downloader middlewares(下载器中间件)：下载中间件是位于Scrapy引擎和下载器之间的钩子框架，主要是处理Scrapy引擎与下载器之间的请求及响应。它提供了一个自定义的代码的方式来拓展 Scrapy的功能。下载中间器是一个处理请求和响应的钩子框架。他是轻量级的，对Scrapy尽享全局控制的底层的系统。

Spider middlewares(蜘蛛中间件)：蜘蛛中间件是介于Scrapy引擎和蜘蛛之间的钩子框架，主要工作是处理蜘蛛的响应输入和请求输出。它提供一个自定义代码的方式来拓展Scrapy的功能。蛛中间件是一个挂接到Scrapy的蜘蛛处理机制的框架，你可以插入自定义的代码来处理发送给蜘蛛的请求和返回蜘蛛获取的响应内容和项目。

Scheduler middlewares(调度中间件)：调度中间件是介于Scrapy引擎和调度之间的中间件，主要工作是处从Scrapy引擎发送到调度的请求和响应。他提供了一个自定义的代码来拓展Scrapy的功能。

### 2.3.2 Scrapy数据处理流程

Scrapy的整个数据处理流程由Scrapy引擎进行控制，其主要的运行方式为：

1. 引擎打开一个域名，时蜘蛛处理这个域名，并让蜘蛛获取第一个爬取的URL
2. 引擎从蜘蛛那获取第一个需要爬取的URL，然后作为请求在调度中进行调度
3. 引擎从调度那获取接下来进行爬取的页面
4. 调度将下一个爬取的URL返回给引擎，引擎将他们通过下载中间件发送到下载器
5. 当网页被下载器下载完成以后，响应内容通过下载中间件被发送到引擎
6. 引擎收到下载器的响应并将它通过蜘蛛中间件发送到蜘蛛进行处理
7. 蜘蛛处理响应并返回爬取到的项目，然后给引擎发送新的请求
8. 引擎将抓取到的项目项目管道，并向调度发送请求
9. 系统重复第二部后面的操作，直到调度中没有请求，然后断开引擎与域之间的联系

## 2.4 去重算法

对于不同的OTA，他们对同一个实体酒店的存储信息很有可能不同，这种不同体现在酒店名称，酒店电话等等。尽管同一个物理酒店的静态信息可能不同，但是同一个实体酒店的信息在不同的OTA中总会极度相似，这便是去重算法的依据。

由于我们已经不需要导入已有的实体酒店数据到本地数据库，所以，我们需要做酒店数据去重的模块，而酒店数据去重模块的核心就是去重算法。

本小节主要介绍酒店数据去重算法，包括KDTree，Shingling及LCS算法。

### 2.4.1 KDTree

KDTree(K-Dimensional Tree)，是一种高维索引二叉树形数据结构，常用于大规模的高维数据空间进行最近邻查找(Nearest Neighbor)和近似最近邻查找(Approximate Nearest Neighbor)。

KDTree性质：

1. KDTree是一棵二叉树
2. 数据只存储在叶子节点
3. 非叶子节点存储的数据结构为(k,m)，k为划分左右子树的维度，m为维度值
4. 对于任意一个非叶子节点，将当前集合按照维度k和维度值m划分为集合left-tree和right-tree，使得任意x属于left-tree满足x(k)<m，任意y属于right-tree满足y(k)>=m

KDTree构建算法：

1. 在k维数据集合中选择具有最大方差的维度k，然后在该维度上选择中值m对该数据集合进行划分，得到两个子集合。同时创建一个结点(k,m)，用于存储维度k和中值m
2. 对两个子集合重复1步骤的过程，直至所有子集合都不能再划分为止。如果某个子集合不能再划分时，则将该子集合中的数据保存到叶子节点

KDTree最近邻查找的算法：

设查询节点为Q

1. 从根节点开始依次与每个非叶子节点作比较， 若Q(k)<m，则访问左子树，否则访问右子树。当到达叶子节点L，计算dcur=dis(Q,L)，pcur=L
2. 进行回溯操作，若对任意一个分支而言，满足条件|Q(k)-m|<dcur且未被访问过，则该分支里可能存在叶子节点L，L满足条件dis(Q,L)<dcur。若寻找到叶子节点L(dis(Q,L)<dcur)，则更新dcur=dis(Q,L)，pcur=L。回溯的判断过程是从下往上进行的，直到回溯到根结点时已经不存在与P更近的分支为止

KDTree在本文作用：

由KDTree的性质和酒店的经纬度坐标，我们可以使用KDTree这种数据结构，通过酒店经纬度坐标完成某个酒店邻近的N个酒店的查找。然后将这N个邻近酒店依次与目标酒店做近似比较。

### 2.4.2 Shingling

Shingling算法常用于计算两个字符串的相似度。

维基百科用一个浅显的例子讲解了shingling算法的原理。比如一个字符串："a student is a student is a student"，分词后的词汇(token，语汇单元)集合是：

(a,student,is,a,student,is, a, student)

那么w=4的4-shingling就是集合：

{(a,student,is,a), (student,is,a,student), (is,a,student,is), (a,student,is,a), (student,is,a,student)}

去掉重复的子集合：

{(a,student,is,a), (student,is,a,student), (is,a,student,is)}

给定shingle的大小,两个字符串A和B的相似度 r 定义为：

r(A,B)=|S(A)∩S(B)| / |S(A)∪S(B)|

其中|A|表示集合A的大小。

因此，相似度是介于0和1之间的一个数值，且r(A,A)=1,即一个文档和它自身 100%相似。

### 2.4.3 LCS

LCS(Longest Common Subsequence)，寻找两个字符串最大公共子序列的算法。可以通过计算len(lcs)/max(len(str1),len(str2))来计算字符串str1与str2的相似度。

LCS类问题属于动态规划(dynamic programming)，通过组合子问题的解而解决整个问题。

描述一个LCS：

给定一个序列X=<x1,x2,...,xm>，对i=0,1,...,m，定义X的第i个前缀为Xi=<x1,x2,...,xi>。例如，如果X=<A,B,C,B,D,A,B>，则X3=<A,B,C>。

设X=<x1,x2,...,xm>和Y=<y1,y2,...,yn>为两个序列，并设Z=<z1,z2,...,zk>为X和Y的任意一个LCS

1. 如果xm=yn，那么zk=xm=yn而且Zk-1是Xm-1和Yn-1的一个LCS
2. 如果xm!=yn，那么zk!=xm蕴含Z是Xm-1和Y的一个LCS
3. 如果xm!=yn，那么zk!=yn蕴含Z是X和Yn-1的一个LCS

LCS递归解：

参照LCS的最优子结构，我们可以得到LCS递归公式如公式2.1所示：

(2.1)

计算LCS长度：

参照LCS递归解，我们可以计算出两个序列的LCS的长度。len(LCS)=c[m][n]

3 需求分析

上一章详细的介绍了电商数据获取及分析系统所要使用的技术，本章将对系统需求进行分析。

## 3.1 功能需求分析

根据用户端和服务器的端需求，电商数据获取及分析系统主要需要实现下述几种功能：

服务器端：

1. 数据库的设计与实现。
2. Spider抓取的设计与实现。
3. 导入新增酒店数据的设计与实现。

用户端：

1. 用户登陆，用户管理，权限管理。
2. Spider抓取数据展示。
3. Spider参数配置。

### 3.1.1 数据库的设计与实现

使用关系型数据库MySQL，存储Spider数据和Spider参数。由于整个系统的核心是Spider抓取和导入新增酒店数据部分，业务层的需求相对较少，所以数据库结构的比较简单。

### 3.1.2 Spider抓取的设计与实现

服务器端最基础的功能，若没有Spider抓取得到的数据，数据展示及导入新增酒店数据模块的功能都没有作用。对于Spider抓取的设计，需要分析站点的浏览顺序和网页Dom结构，然后设计出一套可行的抓取方案。然后在Spider抓取设计的基础上，考虑抓取过程中可能遇到的异常并提供解决方案，抓取站点的酒店静态数据，酒店图片及酒店点评信息。

### 3.1.3 导入新增酒店数据的设计与实现

服务器端最核心功能。从Spider抓取的酒店数据中筛选出新增的酒店数据，导入到本地系统。导入新增酒店数据模块的核心为酒店去重算法。通过结合使用KDTree，Shingling及LCS算法，设计出酒店去重算法。然后结合本地系统数据和Spider抓取数据，实现导入新增酒店数据模块的功能。

### 3.1.4 用户登陆，用户管理，权限管理

在Web前端，进入系统的用户需要事先登陆，且每一个用户都有一定的权限。如果用户有足够的权限，就能实现用户管理和权限管理。

### 3.1.5 Spider抓取数据展示

成功登陆系统的用户，能查看权限范围内的Spider数据。

### 3.1.6 Spider参数配置

成功登陆系统的用户，能查看和修改权限范围内的Spider参数配置。

## 3.2 业务用例设计

基于上一小节的功能设计，本文业务用例总体设计如图3.1所示。

本文业务用例中最高层次用例主要包括用户管理用例和Spider数据管理用例。其中用户管理用例中包含登陆用例，管理用户信息用例，管理用户权限用例。管理用户信息用例中又包括添加用户、删除用户、更改用户和查看用户四个业务用例，管理用户权限用例中又包括添加用户权限、删除用户权限两个业务用例；另外，Spider数据管理用例中包含Spider参数配置用例和Spider抓取数据管理用例。Spider抓取数据管理用例中又包含添加数据、删除数据、更改数据和查看数据四个业务用例。

上述用例包括了本文中的全部功能，也明确了在不同的用例中，相关操作会触发的用户行为。此外，从各个用例间的包含关系与层次关系，可以更加准确的描述系统的结构，用例间的联系。这些都对于从宏观角度开发系统有非常积极的作用。

下面将对部分主要用例进行描述，从前置条件、结束条件、基本事件流等方面进行说明。

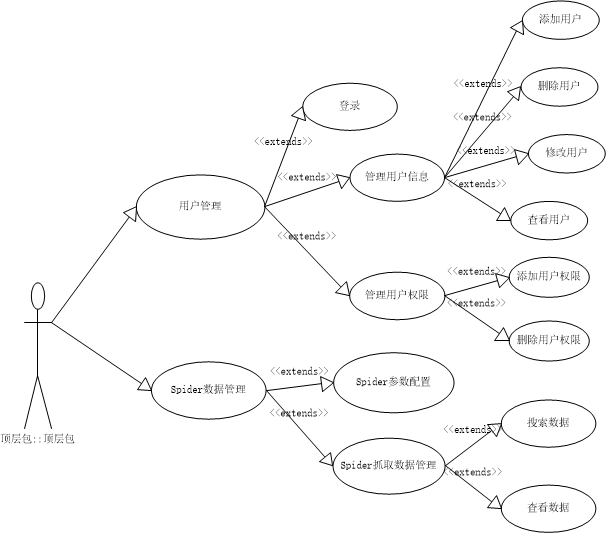


图3.1 系统用例图

Fig. 3.1 Use Case Diagram of System

表3.1是用户登录的用例设计

表3.1 用户登陆用例

Tab. 3.1 User Case Description of Login

|  |  |
| --- | --- |
| 用例 | 用户登陆 |
| 前置条件 | 用户进入欢迎页面 |
| 后置条件 | 成功登陆系统，并跳转至个人主页 |
| 基本事件流 | 跳转至登陆页面，提示用户输入用户名，密码  用户输入用户名，密码，并点击登陆  系统验证用户名、密码信息，如果错误提示用户重新输入  用户输入正确，完成系统登陆 |

表3.2是修改用户信息的用例设计

表3.2 修改用户信息用例

Tab. 3.2 User Case Description of Modification of User’s Information

|  |  |
| --- | --- |
| 用例 | 用户登陆 |
| 前置条件 | 用户成功登录系统能，有修改用户信息的权限 |
| 后置条件 | 成功修改用户信息 |
| 基本事件流 | 跳转至用户列表界面，点击某个用户进入用户信息管理界面  查看用户信息并对相应信息做修改  提示用户确认修改用户信息  确认修改，跳转至用户列表页面 |

表3.3是修改用户权限的用例设计

表3.3 修改用户权限用例

Tab. 3.3 User Case Description of Modification of User’s Privilege

|  |  |
| --- | --- |
| 用例 | 用户登陆 |
| 前置条件 | 用户成功登录系统能，有修改用户权限的权限 |
| 后置条件 | 成功修改用户权限 |
| 基本事件流 | 跳转至用户列表界面，点击某个用户进入用户权限管理界面  查看用户权限并对相应权限做修改  提示用户确认修改用户权限  确认修改，跳转至用户列表页面 |

表3.4是查看Spider抓取数据的用例设计

表3.4 查看Spider抓取数据用例

Tab. 3.3 User Case Description of View of Spider Data Model

|  |  |
| --- | --- |
| 用例 | 用户登陆 |
| 前置条件 | 用户成功登录系统能，有查看Spider抓取数据的权限 |
| 后置条件 | 用户可以查看权限范围内的Spider抓取数据 |
| 基本事件流 | 从主页点击需要查看的数据所在的子菜单  进入需要查看的数据的列表页面  点击所需要查看的数据，进入详细页面查看数据 |

## 3.3 数据流图设计

TODO

# 4 系统设计

## 4.1 系统概要设计

本系统的主要目标是设计并开发一套具有下述主要功能的系统：利用云端以及非关系性数据库搭建服务平台，通过记录用户的历史记录以及兴趣标签，通过优秀的推荐算法为用户推荐视频。作为一个完善的系统，用户也具有账户处理工作。在设计过程中，系统注重各个模块间的分离，以及准确的数据定义，以方便开发者的维护以及扩展编码工作。

### 4.1.1 系统总体架构

根据需求分析与功能、业务逻辑等，设计出如图4.1的系统的总体设计架构：如图所示，用户通过提交表单数据与浏览器进行交互。浏览器通过视图、模板以及返回结果集等方式同注册服务器进行交互。

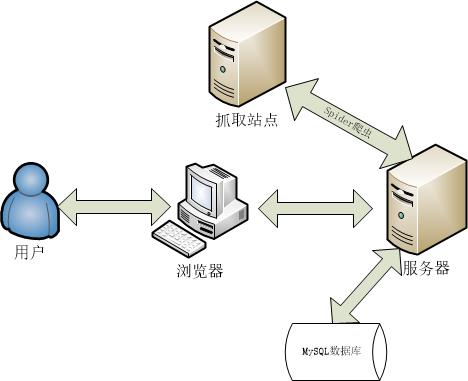


图4.1 系统总体设计

Fig. 4.1 Design of the Whole System

服务器通过Spider爬虫抓取目标站点的数据，然后将数据存储到MySQL数据库。用户请求服务器操作的时候，服务器与MySQL数据库交互，对MySQL数据库中的数据做相应操作。

### 4.1.2 系统交互设计

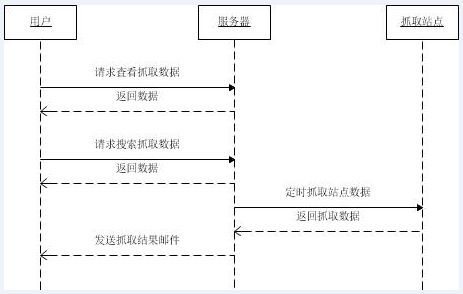


图4.2 系统交互时序图

Fig. 4.2 Interactive Schedule Diagram of Video Recommendation System

整个系统的交互设计如图4.2所示。服务器会定时抓取目标站点的数据，并解析数据存储到MySQL数据库，另外，服务器会对用户发送抓取结果邮件。用户向服务器提交查看和搜索数据的请求，服务器将对应数据展示给用户。

## 4.2 系统详细设计

### 4.2.1 用户功能模块详细设计

完成系统总体设计后，我们将对用户功能模块进行详细设计。图4.3是用户登陆模块详细设计流程图：

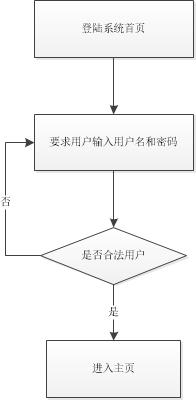


图4.3 用户登陆模块流程图

Fig. 4.3 Flow-Process Diagram of User Register

图4.4为用户搜索、查看抓取数据模块的详细设计流程图：

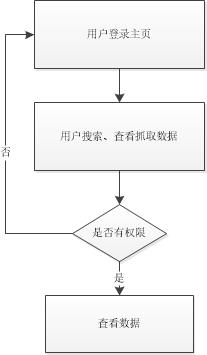


图4.4 用户搜索、查看抓取数据流程图

Fig. 4.4 Flow-Process Diagram of User Searching or View Spider Data

以上两个是用户部分功能流程的详细设计。由于本文的重点是Spider抓取和导入新增酒店数据模块，用户功能模块的逻辑和设计比较简单，基本是通过权限管理来限制用户的行为，所以本文不对用户功能模块多做详细介绍。

### 4.2.2 数据库设计

在MVC的框架当中，数据库的设计非常重要，直接会影响到数据对象以及对数据模型的处理等操作。数据库的设计要对系统的物理设计和逻辑设计都很清晰的情况下才能根据所选择的数据库类型制定准确的存储数据。

数据库的设计首先要根据系统的总体设计结果，参照系统的数据流设计，建立一个正确的数据模型。这个数据模型应当可以准确反应用户的需求，数据如果定义的不妥当，会影响到系统的扩展性，维护性，甚至影响系统的运行效率。数据库的设计一般参照以下基本原则[n]：

[王珊, 萨师煊. 数据库系统概论[M].北京:高等教育出版社, 2006.]

数据的完整性：数据的完整性是指数据的准确性，和依赖关系的稳定性。这两方面的稳定有助于保证输入的错误信息被识别。也防止将一些无效操作、或者数据输入影响系统安全。数据完整性一般包括实体完整性、参照完整性、用户自定义完整性、域完整性等。

数据的一致性：数据的一致性主要是指当数据库事物操作完成时，必须保证所有的数据都是一致的状态。数据必须保持一致性，如果数据不一致，很有可能出现无用对象，冗余数据，导致系统有些功能无法实现的现象。在非关系型数据库中，数据的一致性更为重要，如果数据不保持一致性，可能会导致数据之间识别错误。例如，不同表之间可能有相同的属性，如果这几个属性之间的格式不同。不但会影响到程序逻辑，甚至可能影响到很多其他结果。

数据命名规范性：数据命名规范性对于数据库的设计而言也非常关键，由于数据模型通常都是直接反应在对象设置，变量设置上。命名如果不规范，不但不利于程序的编写，而且还会对编程逻辑产生不良的影响。相反，清晰的命名规则会方便开发人员的代码编写，而且也会帮助数据库管理人员理清数据库表与表之间，字段属性间的物理逻辑关系。

根据需求分析以及酒店数据模型基础，电商数据获取及分析系统一共设计25张表。按照功能分类，可以分为：用户管理表，Spider抓取数据表，Spider爬虫参数配置表。

由于表的数量比较多，以下只列出几张核心的表的详细信息。

表4.1为用户表，存储用户的基本信息。

表4.1 用户表

Tab. 4.1 auth\_user Table

|  |  |  |
| --- | --- | --- |
| 字段名 | 类型 | 描述 |
| id | int(11) | 自增id |
| password | varchar(128) | 用户密码 |
| last\_login | datetime | 最近一次登陆时间 |
| is\_superuser | tinyint(1) | 是否超级用户 |
| user\_name | varchar(30) | 用户账号 |
| first\_name | varchar(30) | 用户名字 |
| last\_name | varchar(30) | 用户姓氏 |
| email | varchar(75) | 用户邮箱 |
| is\_staff | tinyint(1) | 是否能登陆admin模块 |
| is\_active | tinyint(1) | 是否有效账号 |
| date\_joined | datetime | 用户创建时间 |

表4.2为用户权限表，存储用户的权限信息。

表4.2 用户权限表

Tab. 4.2 auth\_permission Table

|  |  |  |
| --- | --- | --- |
| 字段名 | 类型 | 描述 |
| id | int(11) | 自增id |
| name | varchar(50) | 展示给用户的权限名称 |
| content\_type\_id | int(11) | 权限类型 |
| codename | varchar(100) | 在系统中的权限名称 |

表4.3为用户权限关系表，存储用户和权限的映射关系。

表4.3 用户权限关系表

Tab. 4.3 auth\_user\_user\_permissions Table

|  |  |  |
| --- | --- | --- |
| 字段名 | 类型 | 描述 |
| id | int(11) | 自增id |
| user\_id | int(11) | 用户id |
| permission\_id | int(11) | 权限id |

表4.4为携程城市信息表，存储携程城市的基本信息。

表4.4 携程城市信息表

Tab. 4.4 ctrip\_city Table

|  |  |  |
| --- | --- | --- |
| 字段名 | 类型 | 描述 |
| id | int(11) | 自增id |
| href | varchar(100) | 该城市目录下酒店列表主页的超链接 |
| name\_en | varchar(20) | 城市英文名称 |
| name\_ch | varchar(40) | 城市中文名称 |
| group\_name | varchar(1) | 城市所属组名称 |

表4.5为携程酒店信息表，存储从携程网站抓取的酒店数据模型的基本信息。

表4.5 携程酒店信息表

Tab. 4.5 ctrip\_hotel Table

|  |  |  |
| --- | --- | --- |
| 字段名 | 类型 | 描述 |
| id | int(11) | 自增id |
| hotel\_id | varchar(20) | 酒店id |
| href | varchar(100) | 该酒店详细信息页面的超链接 |
| name\_en | varchar(200) | 酒店英文名称 |
| name\_ch | varchar(200) | 酒店中文名称 |
| telephone | varchar(60) | 酒店电话 |
| fax | varchar(100) | 酒店传真 |
| des\_base | longtext | 酒店概要描述 |
| des\_detail | longtext | 酒店详细描述 |
| nation | varchar(30) | 酒店所属郭嘉 |
| province | varchar(30) | 酒店所属省份 |
| city | varchar(30) | 酒店所属城市 |
| location | varchar(30) | 酒店所属区/县 |
| address | longtext | 酒店中文地址名称 |
| road\_cross | longtext | 酒店所在路段 |
| country | varchar(100) | 酒店所属商业区 |
| destination | varchar(30) | 酒店所在目的地 |
| scott\_point | varchar(100) | 酒店经纬度 |
| star | varchar(10) | 酒店星级 |
| score | double | 酒店平均评分 |
| review\_count | int(11) | 酒店被点评次数 |
| services | varchar(200) | 酒店提供的服务 |
| least\_price | double | 酒店参考价格 |
| create\_time | datetime | 数据创建时间 |
| batch\_number | varchar(20) | 数据所属批次号 |

表4.6为携程酒店图片信息表，存储从携程网站抓取的酒店图片数据模型的基本信息。

表4.6 携程酒店图片信息表

Tab. 4.6 ctrip\_hotelimage Table

|  |  |  |
| --- | --- | --- |
| 字段名 | 类型 | 描述 |
| id | int(11) | 自增id |
| hotel\_id | varchar(20) | 酒店id |
| hotel\_name | varchar(200) | 酒店中文名称 |
| title | varchar(128) | 图片名称 |
| category | varchar(128) | 图片所属类别 |
| image | varchar(512) | 图片路径 |

表4.7为携程酒店点评信息表，存储从携程网站抓取的酒店点评数据模型的基本信息。

表4.7 携程酒店点评信息表

Tab. 4.7 ctrip\_hotelreview Table

|  |  |  |
| --- | --- | --- |
| 字段名 | 类型 | 描述 |
| id | int(11) | 自增id |
| hotel\_id | varchar(20) | 酒店id |
| hotel\_name | varchar(200) | 酒店中文名称 |
| customer | varchar(64) | 酒店顾客名称 |
| trip | varchar(64) | 顾客旅行目的 |
| room | varchar(64) | 顾客预订房间类型 |
| review\_date | varchar(64) | 顾客点评时间 |
| score | double | 顾客点评平均分数 |
| small\_c | varchar(64) | 顾客评分明细 |
| advantages | varchar(128) | 顾客评价优点 |
| disadvantages | varchar(128) | 顾客评价缺点 |
| comment\_detail | longtext | 顾客点评明细 |
| useful\_voted | int(11) | 点评获得的点赞次数 |
| hotel\_reply | longtext | 酒店反馈 |

表4.8为爬虫邮件收件人配置表，存储爬虫运行结束之后接收爬虫邮件的收件人邮箱信息。

表4.8 爬虫邮件收件人配置表

Tab. 4.8 scrapy\_manager\_sendemailitem Table

|  |  |  |
| --- | --- | --- |
| 字段名 | 类型 | 描述 |
| id | int(11) | 自增id |
| receiver\_email | varchar(75) | 收件人邮箱 |
| cc\_email | varchar(75) | 抄送邮箱 |
| type | varchar(30) | 收件人类型 |

### 4.2.3 Spider抓取设计

根据定义的数据模型，我们需要设计可行的抓取方案浏览抓取Web站点，并根据网页Dom结构解析需要的数据，然后存储到数据对象中。本文核心的Spider类的UML类图如图4.5所示：

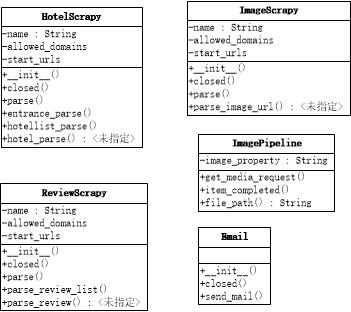


图4.5 Spider抓取模块UML类图

Fig. 4.5 UML Class Diagram of Spider Scrapy

根据系统的类和属性，以及系统的功能需求我们可以定义如下接口：

HotelScrapy类定义的接口：

1. \_\_init\_\_(self, cityid=None, info\_log=None)：构造函数，检查传递参数cityid和info\_log的正确性以及初始化运行Spider所需要的城市URL入口
2. closed(self, reason)：Spider运行结束后调用，记录Spider运行时间及更改Spider运行状态
3. parse(self, response)：通过每个城市的URL入口按照城市分类访问酒店列表
4. entrance\_parse(self, response)：从每个城市的酒店列表中抓取分页数量并访问每个分页
5. hotellist\_parse(self, response)：从每个分页的酒店列表中抓取hotel\_id，将hotel\_id嵌入到URL中，访问该URL进入到酒店详细信息页面
6. hotel\_parse(self, response)：从酒店详细页面抓取需要的关键信息并存储到item中，然后返回item到Pipeline中

ImageScrapy类定义的接口：

1. \_\_init\_\_(self, cityid=None, info\_log=None)：构造函数，检查传递参数cityid和info\_log的正确性
2. closed(self, reason)：Spider运行结束后调用，记录Spider运行时间及更改Spider运行状态
3. parse(self, response)：从现已经抓取的酒店数据中查询city\_id和hotel\_id，将city\_id和hotel\_id嵌入到URL中，访问该URL进入到酒店图片列表页面
4. parse\_image\_url(self, response)：从酒店图片列表页面抓取酒店图片的URL并存储到item中，同时item中还会保存酒店的一些关键字段，然后返回item到Pipeline中

ImagePipeline类定义的接口：

1. get\_media\_requests(self, item, info)：根据item中酒店图片URL，抓取酒店图片
2. item\_completed(self, results, item, info)：记录所有酒店的图片统计信息和每张图片的信息
3. file\_path(self, request, response=None, info=None)：指定每张图片的存储位置，每张图片的存储路径为full/city\_name/hotel\_id/title/image\_guid.jpg

ReviewScrapy类定义的接口：

1. \_\_init\_\_(self, cityid=None, info\_log=None)：构造函数，检查传递参数cityid和info\_log的正确性
2. closed(self, reason)：Spider运行结束后调用，记录Spider运行时间及更改Spider运行状态
3. parse(self, response)：从现已经抓取的酒店数据中查询酒店的href，将该href转化成酒店点评页面的URL，然后访问该URL
4. parse\_review\_list(self, response)：从每个酒店的点评列表中抓取分页数量并访问每个分页
5. parse\_review(self, response)：解析单个酒店的单页点评并存储到item中，然后返回item到Pipeline中

Email类定义的接口：

1. \_\_init\_\_(self, result=None, log\_file=None, scrapy\_type=None)：构造函数，检查传递参数result，log\_file，scrapy\_type的正确性以及从数据库中查询邮件的收件人
2. closed(self, reason)：Spider运行结束后调用，根据scrapy\_type和log\_file参数寻找附件，并调用发送邮件的方法将附件内容发送到收件人
3. send\_email(self, to=[], cc=[], subject="爬虫运行异常", body="", attachs=[])：发送邮件

### 4.2.4 导入新增酒店数据设计

在Spider抓取运行成功且抓取数据成功存储到数据库之后，我们可以根据本地的酒店数据和Spider抓取的酒店数据做数据分析，导入新增酒店数据到本地数据库。

导入新增酒店数据模块的核心为酒店数据去重算法，针对酒店数据去重算法，我们设计出UML类图如图4.6：

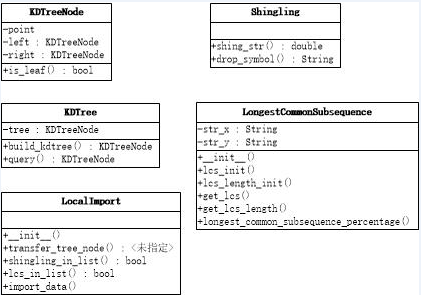


图4.6 导入新增酒店数据模块UML类图

Fig. 4.6 UML Class Diagram of Importing of New Hotel Data

根据酒店去重算法的核心思想，我们可以定义如下接口：

KDTree类定义的接口：

1. build\_kdtree(point\_list, depth)：创建KDTree
2. query(self, query\_point, t=1)：查询目标节点附近的t个最临近节点

Shingling类定义的接口：

1. shing\_str(pre\_str, cmp\_str)：shingling算法比较字符串pre\_str和cmp\_str的相似度
2. drop\_symbol(str)：去除字符串str中的特殊符号

LongestCommonSubsequence类定义的接口：

1. \_\_init\_\_(self, str\_x, str\_y)：构造函数，将字符串转码成UTF-8
2. lcs\_length\_init(self)：在不需要获得LCS而只需要获得len(LCS)的时候的初始化工作
3. get\_cls\_length(self)：返回len(LCS)
4. lcs\_init(self)：在需要获得LCS的时候的初始化工作
5. get\_cls(self)：返回LCS
6. longest\_common\_subsequence\_percentage(str\_x, str\_y)：找出字符串str\_x和字符串str\_y的len(LCS)，用len(LCS)/max\_len(str\_x, str\_y)获得字符串str\_x和str\_y的相似度

LocalImport类定义的接口：

1. \_\_init\_\_(self)：构造函数，读取本地酒店数据，建立KDTree
2. transfer\_tree\_node(self, hotel\_model)：将从数据库查询得到的酒店数据对象转化为KDTree节点对象
3. shingling\_in\_list(self, target\_node, nearest\_node\_list)：将目标节点酒店名称与所有临近节点酒店名称做shingling比较
4. lcs\_in\_list(self, target\_node, nearest\_node\_list)：将目标节点酒店名称与所有临近节点酒店名称做lcs比较
5. import\_data(self, hotel\_list)：对hotel\_list中的酒店对象，用酒店去重算法筛选出新增的酒店对象

5 系统实现

## 5.1 系统环境搭建

### 5.1.1 系统环境综述

开发平台上本系统选用了Ubuntu14.04作为开发的操作系统，在生产环境选用CentOS5.6作为运行的操作系统。

开发环境选用的是Python2.7，Django1.6.8，Scrapy0.24.4，MysqlServer5.6。其中Python2.7是Ubuntu14.04系统中含有的，其他程序都需要自己安装。

### 5.1.2 MysqlServer开发环境搭建

数据库是开发系统的核心，在这里，我们选用开源且使用范围较广的MySQL。

安装MySQL：

sudo apt-get install mysql-server-5.6

由于MySQL默认编码方式不为UTF-8，为了让MySQL良好的支持中文，我们需要配置MySQL的字符集：

locate my.cnf

找到my.cnf文件之后，打开该文件，在[client]下添加行default-character-set=utf8，在[mysqld]下添加行character-set-server=utf8。

sudo /etc/init.d/mysql restart

mysql -u username -p password

show variables like 'character%';

若控制台打印结果如图5.1所示，则表示将MySQL字符集设置为UTF-8成功。

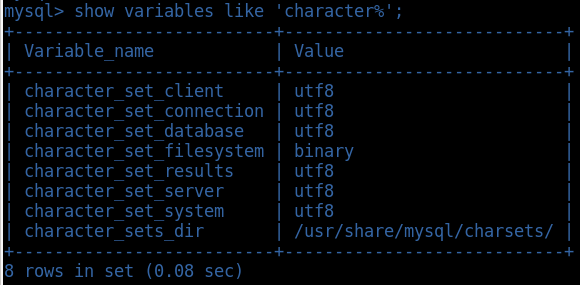


图5.1 MySQL字符集设置结果

Fig. 5.1 Result of MySQL Character Setting

### 5.1.3 Django Web 2.0开发环境的搭建

安装Django以及使用Django搭建Web应用十分简单。

安装Django只需要简单的执行以下几行命令：

tar -zxvf Django-1.6.8.tar.gz

cd Django-1.6.8

sudo python setup.py

等待命令执行完毕之后，即可完成对Django1.6.8的安装。然后执行以下步骤校验Django是否安装成功。如图5.2所示：

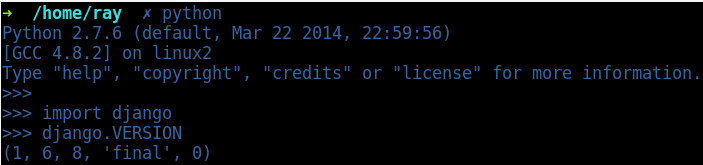


图5.2 Django安装成功验证

Fig. 5.2 Verification of Django Installed

搭建Django Web应用程序也比较简单，一般来说开发一个Django Web应用的步骤大致如图5.3所示。



图5.3 Django开发流程图

Fig. 5.3 Flow-Process Diagram of Developing a Django Project

首先会用django-admin.py文件创建一个Django Web工程，然后在工程目录下创建相应的应用。

创建电商数据获取及分析系统的命令如下所示：

django-admin.py startproject scrapy\_project

cd scrapy\_project

python manage.py startapp app

python manage.py startapp common

cd app

python ../manage.py startapp elong

python ../manage.py startapp ctrip

python ../manage.py startapp scrapy\_manager

然后更改项目中的settings.py文件的配置，在文件中配置database的参数，应用等对应的关系。

随后就是开发者根据数据模型开发models.py文件，按照数据库和类图的定义在此文件中定义数据模型和一些操作数据模型的方法。定义好数据模型之后，在views.py文件中定义如何操作models.py中的数据对象以及接口的实现。然后通过编写前端展示views.py文件中接口返回的数据。最后的流程是添加URL映射，这个映射相当于指定那些页面调用那些视图方法。

以上是一个Django Web开发完成的流程，不过电商数据获取及分析系统的核心是Spider抓取和数据分析，使用Django Web只是为了方便管理和展示静态数据。所以，为了简便开发过程，开发者使用Django框架自带的Admin模块来管理数据模型。因此在该系统中，开发者在Django Web部分只需要关注models.py和admin.py文件。通过在admin.py文件中配置models.py文件中数据模型，即可在Django Web Admin模块中展示和修改models.py文件中的数据对象，极大地提高了系统开发效率。

### 5.1.4 Scrapy开发环境搭建

从Scrapy的官方文档中，我们可以获取简易的Scrapy安装方式。其安装命令如下：

sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 627220E7

echo 'deb http://archive.scrapy.org/ubuntu scrapy main' | sudo tee /etc/apt/sources.list.d/scrapy.list

sudo apt-get update && sudo apt-get install scrapy-0.24

通过以上命令完成Scrapy0.24的安装，由于电商数据获取及分析系统抓取了图片，所以我们还需要安装图片库。

sudo apt-get install libjpeg-dev libfreetype6-dev zlib1g-dev

sudo pip install Pillow

在创建好Django Web工程及搭建好Scrapy开发环境之后，我们接下来就将Scrapy工程嵌入到Django Web工程中，执行一下命令：

cd scrapy\_project

mkdir scrapys && cd scrapys

mkdir ctrip && cd ctrip

scrapy startproject hotel && cd hotel/hotel

这样，我们就创建了一个抓取酒店静态数据的Spider，然后我们需要更改当前目录下的settings.py文件，配置Spider的环境变量，访问网页时的DEFAULT\_REQUEST\_HEADERS和USER\_AGENT，每两次抓取间隔时间DOWNLOAD\_DELAY等等配置。

其中配置Spider环境变量比较重要，通过配置Spider环境变量，使得Spider与Django框架融合。该配置如下：

import os

import sys

sys.path.append("../../../../")

os.environ['DJANGO\_SETTINGS\_MODULE'] = 'scrapy\_project.settings'

通过以上步骤，我们基本完成Scrapy开发环境的搭建。然后，我们需要在items.py文件中引用在Django的models.py文件中定义的数据模型或者创建数据模型，在spider目录下创建Spiders完成数据抓取以及解析，在pipelines.py文件中对抓取及解析获得的item进行处理。

## 5.2 Spider抓取实现

TODO

## 5.3 去重算法实现

TODO

6 系统测试

## 6.1 测试方法及用例设计

### 6.1.1 单元测试

TODO

### 6.1.2 兼容性测试

传统的C/S架构中，兼容性测试大多数指的是系统的兼容性。随着B/S架构和浏览器的多元化发展，兼容性还有了一个新的分类就是浏览器兼容测试。在本视频推荐系统中，前端的设计尽量保持浏览器制定的标准。所以系统和浏览器的兼容性是本方案考虑的问题。

在测试系统的选择上，在Ubuntu和Mac OS上系统均可以正常运行。在浏览器的选择上，Ubuntu上Firefox、Chromium，Mac OS上Safari、Chrome、Firefox也都可以正常显示。在以上的环境中，系统均可以正常运行，显示了系统稳定与良好的兼容性。

### 6.1.3 维护性测试

优秀的系统应当有容易维护的特点，Django模块间松耦合的设计也使系统的维护性大大增强。

TODO

## 6.2 测试结果分析

TODO

## 6.3 系统运行效果

本小节主要介绍系统的运行的效果，完成系统环境的配置。登陆服务器首页，会弹出如图6.1的登陆页面。

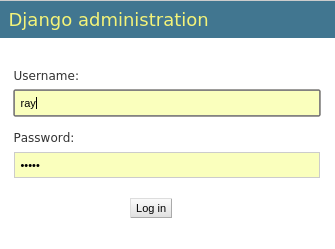


图6.1 服务器登陆首页

Fig. 6.1 Login Page of System

图6.2为服务器Admin首页页面，可以从Admin首页页面中进入到更改用户信息和权限的模块，查看Spider抓取数据模型模块，Spider参数配置模块。

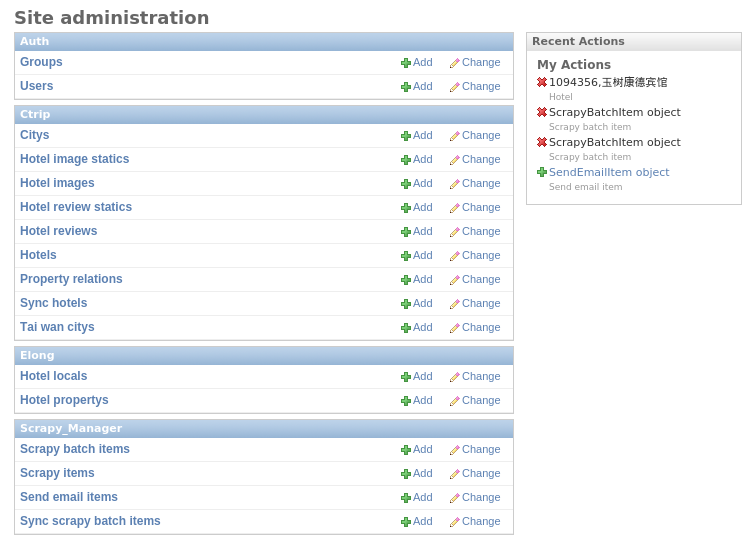


图6.2 服务器Admin模块首页

Fig. 6.2 Home Page of System’s Admin Part

结 论

TODO

参 考 文 献

[] TODO

致 谢

毕业设计即将结束也意味着我四年本科生的生活即将划上终止符。四年时光，转眼即逝，在此回望之际，我要衷心感谢以下人士。

首先，我要感谢我的导师单世民副教授。从最初的数据库系统认识了单老师，当时就被单老师严谨的教学风格，诙谐幽默深深吸引。在以后深入接触的过程中，更发现单老师是一个品德高尚，关爱学生的老师。单老师的师说心语：“此时的你，不论是在踌躇满志还是黯然神伤，都不要使过去的自己成为挥之不去的负担；此时的你，不论是胸怀鸿鹄之志还是疲于随波逐流，都不要让未来的你讨厌现在的自己。静下心来，找回自己，脚踏实地，活在当下。”无时无刻不在提醒我要做什么样的人。在这里，我由衷地向单老师道出一句心生：谢谢老师您的教诲。

然后，我要感谢姜国海和蒋光远老师。是两位老师给我提供了宝贵的实习经历，让我了解了开发Java Web的整个流程，并在我遇到难题的时候帮助我解决困难。

然后，我要感谢我的室友刘延祥、李嘉晨，感谢我的朋友张仁宇、辛佳俊、肖萧、尹顺康、焦华蕾。感谢你们丰富我的大学生活。

最后感谢百忙之中抽空对本文进行评阅的答辩组老师们，是你们的指点让我可以更加明确努力的方向。