

**密级：机密**

**大众点评口碑爬虫规范**

**版本 1.0**

**修订历史记录**

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| 2017.3.15 | 1.0 |  | 闫茹晶 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

[1.数据结构设计 3](#_Toc25404)

[1.1用户表(dazhong\_user) 3](#_Toc14132)

[1.2点评表(dazhong\_comment) 4](#_Toc5567)

[2. 概要设计 4](#_Toc6328)

[2.1入口分析 5](#_Toc27189)

[2.2爬虫分析 5](#_Toc1010)

[1. 点评信息分析 5](#_Toc29642)

[2. 用户信息分析 5](#_Toc30218)

[3. 详细设计 6](#_Toc7971)

[3.1点评信息爬虫 6](#_Toc25787)

[1. 创建项目comment 6](#_Toc8183)

[2. item设计 6](#_Toc7720)

[3. settings设计 6](#_Toc22553)

[4. spider设计 6](#_Toc37)

[3.2用户信息爬虫 7](#_Toc27673)

# 1.数据结构设计

大众点评网站主要内容就是用户对各类商家的评论信息，从而帮助其他用户对商家的评判。网站就是实现了用户、商家以及它们之间的点评关系，所以设计两张表：用户表(user)和点评表(coment)。

## 1.1用户表(dazhong\_user)



图 1用户示例图

表 1用户表

|  |  |  |
| --- | --- | --- |
| **字段名** | **类型** | **描述** |
| user\_id | varchar(10) | 用户的id |
| name | varchar(100) | 用户名称 |
| sex | varchar(5) | 用户性别 |
| city | varchar(50) | 用户所在城市 |
| contribution | int | 用户贡献值；综合评价用户包括点评数量、图片等 |
| attention | int | 该用户的关注数量 |
| fans | int | 该用户的粉丝数量 |
| interaction | int | 该用户的互动数量 |
| register\_time | date | 该用户的注册时间 |
| login\_time | date | 该用户的最后登录时间 |
| user\_url | varchar(30) | 用户主页 |

## 1.2点评表(dazhong\_comment)



图 2点评示例图

表 2点评表

|  |  |  |
| --- | --- | --- |
| **字段名** | **类型** | **描述** |
| comment\_id | varchar(20) | 该点评的id |
| user\_id | varchar(10) | 写该点评的用户id |
| user\_url | var char(50) | 该用户的主页 |
| star | int | 用户点评的星级 |
| content | text | 点评的内容 |
| support\_num | int | 点赞数 |
| comment\_num | int | 评论数 |
| time | date | 评论日期 |
| Shop\_id | varchar(15) | 商家的id |
| shop\_name | int(10) | 评论商家的名称 |
| shop\_kind | varchar(20) | 商家所属分类 |
| shop\_url | varchar(50) | 商家的网址 |

# 概要设计

## 2.1入口分析

在大众点评网站(http://www.dianping.com/)的搜索框内输入要搜索的内容（比如火锅），点击搜索后会出现相关的商家信息。搜索结果的url如下：

<http://www.dianping.com/search/keyword/1/0_%E7%81%AB%E9%94%85>

分析：

中间的1代表当前页面城市;

%E7%81%AB%E9%94%85 则代表火锅

QQ截图20170422160531

图 3 搜索结果的url

所以爬虫的最开始的url为：

**http://www.dianping.com/search/keyword/ + city\_id + /0\_ + content**

其中：id为要爬取哪所城市的id，content为要爬取的内容

## 2.2爬虫分析

### 点评信息分析

在搜索结果中可以看到，搜索结果最多为50页，每页有18个商家。所以要做的就是从当前页面抓出商家的店铺id然后通过店铺id构建出该商铺的点评网页。

随便点开一个商家后发现，点评信息网址为：

<http://www.dianping.com/shop/69228764/review_all>

所以，通过搜索结果页面，构造出50\*18个如下的网址作为爬出点评信息的网址：

**<http://www.dianping.com/shop/> + shop\_id + /review\_all**

其中：shop\_id为抓取出来的商家的id

### 用户信息分析

在点评页面可以看到，每页都有若干个点评信息，其中有用户的id,name以及用户主页等信息。所以当前页面抓取到的user\_id用于抓取用户信息的入口。



图 4 从点评网页抓取用户主页示例

所以就可以得到如下所示的用户主页网址：

**http://www.dianping.com/member/ + user\_id**

其中：user\_id为用户的id

# 详细设计

## 3.1点评信息爬虫

### 创建项目comment

在cmd窗口中执行：

**scrapy startprject comment**

### item设计

在comment/items.py中输入下列代码，完成items设计

**comment\_id = Field()**

**user\_id = Field()**

**star = Field()**

**content = Field()**

**support\_num = Field()**

**time = Field()**

**shop\_id = Field()**

**shop\_name = Field()**

**shop\_kind = Field()**

**shop\_url = Field()**

### settings设计

在comment/settings.py完成相关设置

(1)设置User\_agent:

**USER\_AGENT = 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.102 Safari/537.36'**

1. 设置文件存储格式以及存储位置

**FEED\_FORMAT = 'CSV'**

**FEED\_URI = u'file:///C:/Users/yanrujing/Desktop/comment.CSV'**

### spider设计

集体实现方法为：将所有网页的url放在同一个数组中，每取出一个判断是否是商家列表页面，还是点评信息页面。

经过测试之后发现，在大众点评商家的点评信息页面有两种结构，所以是否为点评信息页面要判断两次，看是结构一，还是结构二。

用xpath()方法获取三种网页结构的独特之处：

**selector = Selector(response)**

**shop\_list = selector.xpath('//div[@id="shop-all-list"]')**

**shop\_comment\_1 = selector.xpath('//div[@class="main"]')**

**shop\_comment\_2 = selector.xpath('//div[@class="main page-sa Fix"]')**

如果是shop\_list页面：

1. 抓取出该页面所有的商家网址，构造出评论信息的url，并加入到还需爬的网址的数组中：

**url =shop\_list.xpath('//div[@class="tit"]/a[@data-hippo-type="shop"]/@href').extract()**

**for each in url:**

**next = "http://www.dianping.com"+each+"/review\_more"**

**yield Request(next, callback=self.parse)**

1. 然后找出搜索结果的下一页，如果存在，找到下一页的url并加入到数组中：

**nextLink = selector.xpath('//div[@class="page"]/a[@class="next"]/@href').extract()**

**if nextLink:**

**yield Request(self.url+nextLink[0], callback=self.parse)**

如果是comment\_1或者是comment\_2，则在当前页面中抓取items中所列出来的数据，具体见源代码。

值得注意的是评论页面也有下一页，同理：

**nextPage = selector.xpath('//a[@class="NextPage"]/@href').extract()**

**if nextPage:**

**shop\_url\_self = self.url + '/shop/' + str(shop\_id[0])**

**yield Request(shop\_url\_self + '/review\_all' + nextPage[0], callback=self.parse)**

**注：运行爬虫是输入的内容记得打””，如”火锅”**

## 3.2用户信息爬虫

开始和点评信息爬虫是一样的，建项目，设置items，settings等，具体结合源代码和点评信息爬虫可以知道怎么做的了，在这里就不在解释了

不过，用户信息的爬虫的url是从comment表中查找出来的，具体实现如下：

**def start\_requests(self):**

**file\_object = open(r'C:\Users\yanrujing\Desktop\comment.csv', 'r')**

**try:**

**for line in file\_object:**

**x = line.strip()**

**user\_id = x.split(',')[-1]**

**self.start\_urls.append("http://www.dianping.com/member/" + user\_id)**

**for url in self.start\_urls:**

**yield self.make\_requests\_from\_url(url)**

**finally:**

**file\_object.close()**

**说明：打开comment爬虫的comment表，每次读一行，用 ”,” 分割，由于user\_id位于最后一位，取出user\_id然后构造出用户主页并加入到start\_url中**

注：其中源代码中有一个export.py文件，它是用于指定文件输出顺序的，其中的一些参数位于settins.py文件中。

# 城市id

源代码中有一个city\_id.csv文件，其中就是各个城市的id以及城市名，具体爬哪座城市的信息，对应在该文件查出对应的id即可，开始运行爬虫