目录

摘要

1. 绪论
   1. 研究背景
      1. 互联网时代数据量急剧提升
      2. 海量数据为机器学习提供了测试样本
      3. 机器学习算法应用于大数据的商业化案例及其意义
   2. 海内外相关应用及其影响综述
      1. 淘宝个性化广告推荐
      2. 大数据杀熟——Amazon公司的差别定价实验
      3. 争议中前行——大数据与隐私权
   3. 本文的研究课题和研究方法
      1. 研究课题
      2. 研究方法
2. 数据采集系统

2.1 数据库

2.1.1 MySQL简介

2.1.2 为什么采用MySQL

2.1.3 MySQL图形化操作工具

2.2 Python简介

2.3 JavaScript、AJAX简介

2.4 高德开放平台简介

2.5 静态页面爬虫——Scrapy框架简介及实施思路

2.5.1 Scrapy各组件及其功能简介

2.5.2 浏览器插件Toggle JavaScript、JSON-handle介绍

2.5.3 学校——爬取高德地图并写入MySQL

2.5.4 公共交通——爬取高德地图并写入MySQL

2.5.5 小区——使用多重爬虫爬取高德地图并写入MySQL

2.5.6 小区房价——爬取链家并写入MySQL

2.5.7 改进——爬取安居客并写入MySQL

2.5.8 更新所有分组中所有元组的房价

2.6 AJAX爬虫机器学习简介

3.1 机器学习概述

3.2 无监督学习和有监督学习

3.3 线性分类器和Logistic回归

1. 数据清洗及处理
2. 展示网站架构
3. 结论与展望

**一、绪论**

1.1 研究背景

1.1.1 互联网时代数据量急剧提升

互联网自诞生之日起，便深刻地影响着人类的行为乃至思考方式。

在互联网初期，硬件成本高昂，接入互联网的人数少，渠道狭窄，

互联网上流通的数据量小，搜索引擎" 雅虎"在最初是通过手工方

式增加用户要求的内容到网站上。

风云变幻，由于摩尔定律（集成电路上可容纳的晶体管的数量，

每隔18个月就会增加一倍）的作用，硬件成本大大降低，加上智

能手机的普及，截至2017年，互联网覆盖率达47%，数十亿人天

登陆互联网，每天产生约2.5EB（）的数据

1.1.2 海量数据为机器学习提供了测试样本

1.1.3 机器学习算法应用于大数据的商业化案例及其意义

1.2 海内外相关应用及其影响综述

1.2.1 淘宝个性化广告推荐

1.2.2 大数据杀熟——Amazon公司的差别定价实验

1.2.3 争议中前行——数据与隐私权

1.3 本文的研究课题和研究方法

1.3.1 研究课题

1.3.2 研究方法

**二、数据采集系统**

2.1 数据库

2.1.1 MySQL简介

MySQL是一个关系型数据库管理系统，由瑞典MySQL AB 公司开发，目前属于 Oracle 旗下产品。MySQL 是最流行的关系型数据库管理系统之一，在 WEB 应用方面，MySQL是最好的 RDBMS (Relational Database Management System，关系数据库管理系统) 应用软件。

2.1.2

2.5 静态页面爬虫——Scrapy框架简介及实施思路

2.5.2 浏览器插件Toggle JavaScript、JSON-handle介绍

工具1：Chrome浏览器插件——Toggle JavaScript

工具下载地址：<https://chrome.google.com/webstore/detail/toggle-> javascript/cidlcjdalomndpeagkjpnefhljffbnlo?utm\_source=chrome-app-launcher-info-dialog（提示：需要科学上网才能打开链接）

工具作用：这个插件可以帮助我们快速直观地检测网页里哪些信息是通过AJAX异步加载而来的，具体操作是启动此插件，则页面中的JS代码将不会执行，因而如果页面中通过AJAX加载的信息将会无法渲染

例子：页面：在"链家"搜索"南铁华庭"

页面URL：<https://gz.lianjia.com/ershoufang/rs南铁华庭/>

正常加载页面：



启动插件后：



对比上面两图可知，启动插件后，本应显示的” 相关小区”板块没有显示，说明该板块是通过AJAX异步加载得到的

使用Toggle JavaScript原因：原生Scrapy框架是无法爬取AJAX页面的，因此在写爬虫前我们需要判断该页面是否通过AJAX渲染

工具2：Chrome浏览器插件——JSON-handle

工具下载地址：<https://chrome.google.com/webstore/detail/toggle->

javascript/cidlcjdalomndpeagkjpnefhljffbnlo?utm\_s

ource=chrome-app-launcher-info-dialog

（提示：需要科学上网才能打开链接）

工具作用：帮我们格式化Json串,从而让我们以一个更友好的方式查看

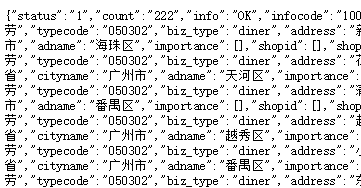
JSON内的信息

例子：

<http://restapi.amap.com/v3/place/text?key=85692e965f1085a96d17e27ac6486979&keywords>=麦当劳&city=广州

例子说明：该链接是根据高德地图API拼接而成，表示搜索广州内的麦当劳

未使用插件的情况：



使用插件后的情况：



对比上面两图，可以发现，使用插件后整个页面布局变得整洁且优美，更有利于我们对返回的信息进行分析

**2.5.6 小区房价——爬取安居客并写入MySQL**

安居客网站介绍：

安居客（Anjuke lnc.）成立于2007年1月，是国内房地产租售服务平台，专注于房地产租售信息服务。安居客集团在全国31个城市设有分公司，员工超过1500人，网站月独立访问用户已突破6900万



安居客页面加载方式分析：

安居客广州地区小区的搜索链接的一般形式是：

guangzhou.anjuke.com/sale/rd1/?from=zjsr&kw=XXXX(其中XXXX代

表小区名)

优化需要爬取的小区：

在爬取"链家"的时候忽略了一个事实：爬取到的小区中存在大量重复的情况，因此，我们可以对数据表进行去重，之后再爬取

安居客反爬虫措施:

和"链家"不同，"安居客"是设置了一定数量的反扒措施的，例如：

1、每一个URL必须设置UA(User-Agent)，否则是无法建立连接的（爬取”链家”并不需要UA）。并且UA最好采用自己浏览器访问时候的UA，在网络上搜索找到的UA也不一定能够成功访问。User Agent中文名为用户代理，简称 UA，它是一个特殊字符串头，使得服务器能够识别客户使用的操作系统及版本、CPU 类型、浏览器及版本、浏览器渲染引擎、浏览器语言、浏览器插件等

2、定时手动输入验证码。在爬取过程中，一般2~6个小区就会有一个小区能够成功抓取到房价信息，但是在爬虫运行5分钟后，日志中显示已经连续几十个小区没有爬到信息了，这引起了我的警觉：莫非触发了”安居客”的反扒措施？此时，我通过浏览器手动登陆”安居客”网站，发现竟然需要输入验证码才能访问，输入验证码之后，发现又能成功爬取小区房价信息了。后来经过多次观测，我发现大概每隔10分钟左后就需要使用浏览器手动登入”安居客”，这样网站安全防护机制就不会对爬虫生效了。

2.5.8 更新所有分组中所有元组的房价

通过去掉重复的小区之后，需要爬取的小区数量就大大地减少了，加快了爬取速度，但是也带来了新的问题:



如上图所示：一个名为" 百花豪苑"的小区，在数据表中共出现了四条记

录，但是却只有一条记录的price非空，这说明虽然是对小区进行了分

组，然后更新(update)了数据，但是实际上只更新了分组中的一条记录

而已，但是分组往往不止一条记录，因此我们需要将分组的其他记录也

更新上，实现思路：

1. 先把所有含有房价信息(price )的分组从数据表(amapcommercialhouseitem )中提取出来，分组依据是小区名(name )，保存下每一个分组的房价
2. 将选出的分组中每一行的小区名提取出来，根据小区名(name)选出该分组的所有房价为NULL的小区
3. 用步骤一中保存下的房价来设置该分组房价为NULL的小区的房价

流程示意图如下：



实现代码（Python）：

**import** **pymysql**

conn = pymysql.connect(

host='localhost',

db='studentscore',

user='root',

passwd='123456',

charset='utf8',

use\_unicode=**True**

)

cursor = conn.cursor()

sql = 'SELECT id,name,price FROM amapcommercialhouseitem WHERE price IS NOT NULL GROUP BY name'

cursor.execute(sql)

Groups = cursor.fetchall() *# Groups是从"amapcommercialhouseitem"读取到的信息*

cursor.close()

conn.close()

conn2 = pymysql.connect(

host='localhost',

db='studentscore',

user='root',

passwd='123456',

charset='utf8',

use\_unicode=**True**

)

conn3 = pymysql.connect(

host='localhost',

db='studentscore',

user='root',

passwd='123456',

charset='utf8',

use\_unicode=**True**

)

**for** temp **in** Groups:

print('id:', temp[0], ' name:', temp[1], ' price:', temp[2])

*# conn2.ping(True) # Check if the server is alive*

cursor2 = conn2.cursor()

FillGroup = "SELECT id,name,price FROM amapcommercialhouseitem WHERE price IS NULL AND name = *%s*"

cursor2.execute(FillGroup, (temp[1])) *# temp[1]是name(小区名)*

GroupMembers = cursor2.fetchall()

cursor2.close()

**for** g **in** GroupMembers:

*# conn3.ping(True) # Check if the server is alive*

cursor3 = conn3.cursor()

sql = """

UPDATE amapcommercialhouseitem SET price=*%s* WHERE id=*%s*

"""

cursor3.execute(sql, (temp[2], g[0]))

conn3.commit() *# 提交，不然无法保存新建或者修改的数据*

cursor3.close()

conn2.close()

conn3.close()