**开源软件基础大作业**

**二手房交易网站数据分析与可视化**

学 院（系）：软件学院

专 业：软件工程

学 生 姓 名：廉佳璐 201992005

杜丽萍 201992147

朱晓筱 201992439

指 导 教 师：任志磊

完 成 日 期：2022年1月2日

大连理工大学

Dalian University of Technology

目录

[1 系统概述 2](#_Toc91858432)

[2 系统架构 3](#_Toc91858433)

[3 小组组成 3](#_Toc91858434)

[4 系统实现 4](#_Toc91858435)

[4.1 数据采集模块 4](#_Toc91858436)

[4.2 数据分析模块 6](#_Toc91858437)

[4.3 数据展示模块 8](#_Toc91858438)

[5 系统运行结果 9](#_Toc91858439)

[6 个人小结 10](#_Toc91858440)

# 1 系统概述

WebMagic的结构分为Downloader、PageProcessor、Scheduler、Pipeline四大组件，并由Spider将它们彼此组织起来。这四大组件对应爬虫生命周期中的下载、处理、管理和持久化等功能。而Spider则将这几个组件组织起来，让它们可以互相交互，流程化的执行，可以认为Spider是一个大的容器，它也是WebMagic逻辑的核心。

MongoDB是一个介于关系数据库和非关系数据库之间的产品，是非关系数据库当中功能最丰富，最像关系数据库的。它支持的数据结构非常松散，是类似json的bson格式，因此可以存储比较复杂的数据类型。Mongo最大的特点是它支持的查询语言非常强大，其语法有点类似于面向对象的查询语言，几乎可以实现类似关系数据库单表查询的绝大部分功能，而且还支持对数据建立索引，支持多种语言。我在其他队员编写的爬虫程序采集到的数据经过spark分析后得到的分析结果的基础上，建立了MongoDB数据库与分析表，同时，为了使分析结果更加直观，也通过web系统访问获得了该数据库中的分析结果，实现了可视化。

Spark是一个以复杂计算为核心的大数据分析框架，是MapReduce的“后继者”，具备高效性、通用性等特点，被广泛应用于大数据处理技术中。

总的来说，Spark使用各种各样原先需要多种不同的分布式平台的场景，包括批处理、迭代算法、交互式查询、流处理。通过在一个统一的框架下支持这些不同的计算，Spark使我们可以简单而低耗地把各种处理流程整合在一起。。而这样的组合，在实际的数据分析过程中很有意义。不仅如此，Spark的这种特性还大大减轻了原先需要对各种平台分别管理的负担。

# 2 系统架构

本系统所需的数据是来源于二手房交易网站，该网站中包含了众多房源的详细信息。在本案例中，我们通过WebMagic（Java爬虫框架）编写网络爬虫程序，采集北京市二手房交易数据，将采集的数据存储到MongoDB数据库中。当数据采集完成后，利用Spark计算框架读取MongoDB中存储的二手房交易数据，并进行离线分析，最后将分析结果存储到MongoDB数据库中。为了可以更加直观的查看分析结果，我们通过Web系统获取MongoDB数据库中存储的分析结果，实现数据的可视化。

# 3 小组组成

|  |  |  |
| --- | --- | --- |
| **姓名** | **角色** | **分工** |
| **廉佳璐** | 组长 | 负责编写爬虫程序，采集数据 |
| **杜丽萍** | 组员 | 负责根据数据建立MongoDB数据库，并实现分析结果可视化 |
| **朱晓筱** | 组员 | 负责使用spark进行数据分析，并进行分析反馈 |

# 4 系统实现

## 4.1 数据采集模块

利用Java爬虫框架WebMagic编写爬虫程序，用于采集某房屋交易网站中二手房交易数据信息，并将采集的数据存储到MongoDB数据库中。

爬虫部分：

1. #导包
2. #请求
3. **import** requests
4. #解析
5. **from** lxml **import** etree
6. #延时
7. **from** time **import** sleep
8. #存储
9. **import** pandas as pd
10. #数据正则化
11. **import** re
12. #相应存储的列表
13. hu\_xing = []
14. area = []
15. chao\_xiang = []
16. other = []
17. lou\_cen = []
18. lei\_xin = []
19. dan\_jia = []
20. zong\_jia = []
21. #循环进行请求，实现翻页
22. **for** i **in** range(1,100):
23. # 休眠，防止反爬
24. sleep(1)
25. url = f'https://bj.lianjia.com/ershoufang/pg{i}/'
26. headers = {
27. "User-Agent": "Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/95.0.4638.69 Safari/537.36"
28. }
29. html = requests.get(url,headers=headers).content.decode("utf-8")
30. **print**(f'获取第{i}页数据')
31. # 用xpath解析数据
32. e = etree.HTML(html)
33. #房屋信息
34. text = e.xpath('//ul/li//div[@class="houseInfo"]/text()')
35. dan\_jia\_1 = e.xpath('//div[@class="unitPrice"]/span/text()')
36. zong\_jia\_1 = e.xpath('//div[@class="totalPrice totalPrice2"]/span/text()')
37. **for** i,d,z **in** zip(text,dan\_jia\_1,zong\_jia\_1):
38. xin\_xi = i.split('|')
39. hu\_xing.append(xin\_xi[0])
40. area.append(re.findall(r'\d+\.?\d+?',xin\_xi[1])[0])
41. chao\_xiang.append(xin\_xi[2].replace(' ',''))
42. other.append(xin\_xi[3])
43. lou\_cen.append(xin\_xi[4])
44. lei\_xin.append(xin\_xi[5])
45. dan\_jia.append(d)
46. zong\_jia.append(z)
47. #数据汇总，保证无缺失
48. data = {
49. "hu\_xing":hu\_xing,
50. 'area':area,
51. 'chao\_xiang':chao\_xiang,
52. 'other':other,
53. 'lou\_cen':lou\_cen,
54. 'lei\_xin':lei\_xin,
55. 'dan\_jia':dan\_jia,
56. 'zong\_jia':zong\_jia
57. }

60. #存储数据
61. pt = pd.DataFrame(data=data)
62. pt.to\_csv('北京.csv')

定义环境，连接数据库

1. val conf: SparkConf = **new** SparkConf().setAppName(getClass.getName).setMaster("local[\*]")
2. val spark: SparkSession = SparkSession.builder().config(conf)
3. .config("spark.mongodb.output.uri", "mongodb://hdp/test.demo01")
4. .getOrCreate()

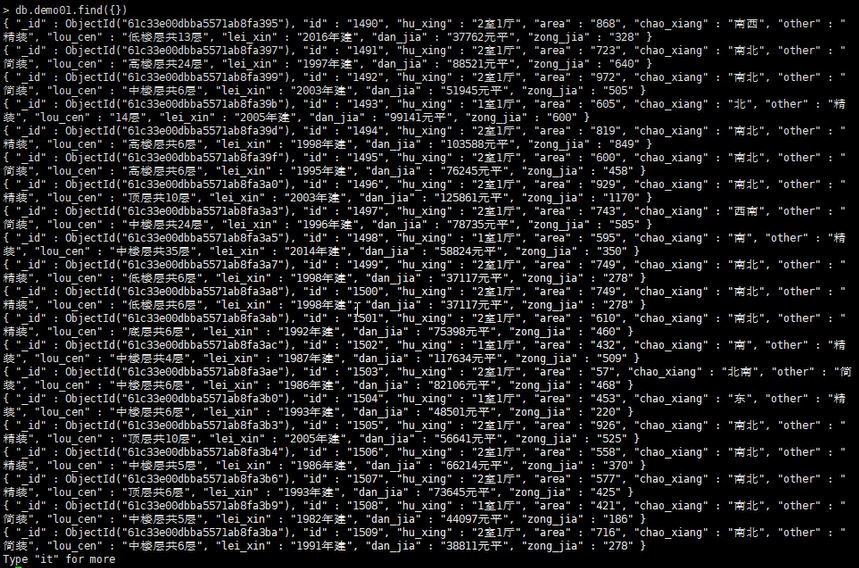
转换实体类

1. val modelRDD: RDD[DataModel] = spark.sparkContext.textFile("input/")
2. .filter(x => x.length > 0 && !x.contains("hu\_xing") && x.split(",").length == 10)
3. .map {
4. x =>
5. val strings: Array[String] = x.replaceAll("[\\s\*$&#/\"'\\.:;?!\\[\\](){}<>~\\\_]+","").split(",")
6. DataModel(strings(0), strings(1), strings(2), strings(3),strings(4),strings(5),strings(6),strings(7)+strings(8),strings(9))
7. }
8. modelRDD.foreach(println(\_))

转化document对象

1. val docRDD: RDD[Document] = modelRDD.map(
2. x => {
3. val document = **new** Document()
4. document.append("id", x.id).append("hu\_xing", x.hu\_xing).append("area", x.area).append("chao\_xiang", x.chao\_xiang).append("other", x.other)
5. .append("lou\_cen", x.lou\_cen).append("lei\_xin", x.lei\_xin).append("dan\_jia", x.dan\_jia).append("zong\_jia", x.zong\_jia)
6. document
7. }
8. )

运行结果



## 4.2 数据分析模块

通过爬虫程序采集的数据内容，采用Spark计算框架对采集的数据进行统计分析，将分析的数据存入到MongoDB中。

加载数据，创建视图

1. val df: DataFrame = MongoSpark.load(spark)
2. df.createOrReplaceTempView("house\_info")
3. df.show()

分析热门户型

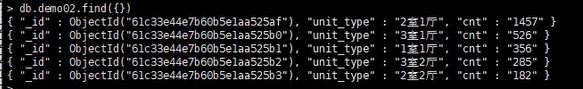
1. val resDf: DataFrame = spark.sql(
2. """
3. |select hu\_xing, count(\*) cnt
4. |from house\_info
5. |group by hu\_xing
6. |order by cnt desc
7. |limit 5
8. """.stripMargin)

转换document对象

1. val docRDD: RDD[Document] = resultRDD.map(
2. x => {
3. val document = **new** Document()
4. document.append("unit\_type", x.\_1).append("cnt", x.\_2)
5. document
6. }
7. )

写入mongo

1. MongoSpark.save(docRDD)

运行结果

## 4.3 数据展示模块

采用Servlet+JSP+Echarts实现动态数据可视化。

1. **from** flask **import** Flask, render\_template
3. app = Flask(\_\_name\_\_)
5. @app.route('/01')
6. **def** echart01():
7. **return** render\_template('./result.html')
9. **if** \_\_name\_\_ == '\_\_main\_\_':
10. app.run()

获取结果表

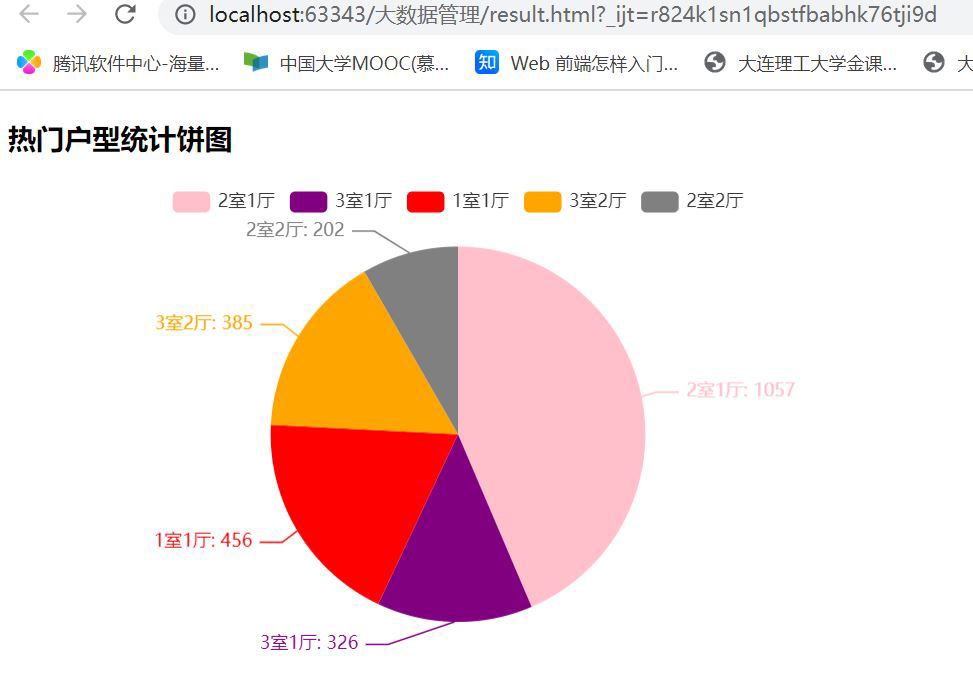
1. **def** get\_db\_df():
2. client = pymongo.MongoClient('hdp', 27017)
3. db = client['test']
4. collection = db['demo02']
5. res = collection.find({},{'\_id':0})
6. **return** res

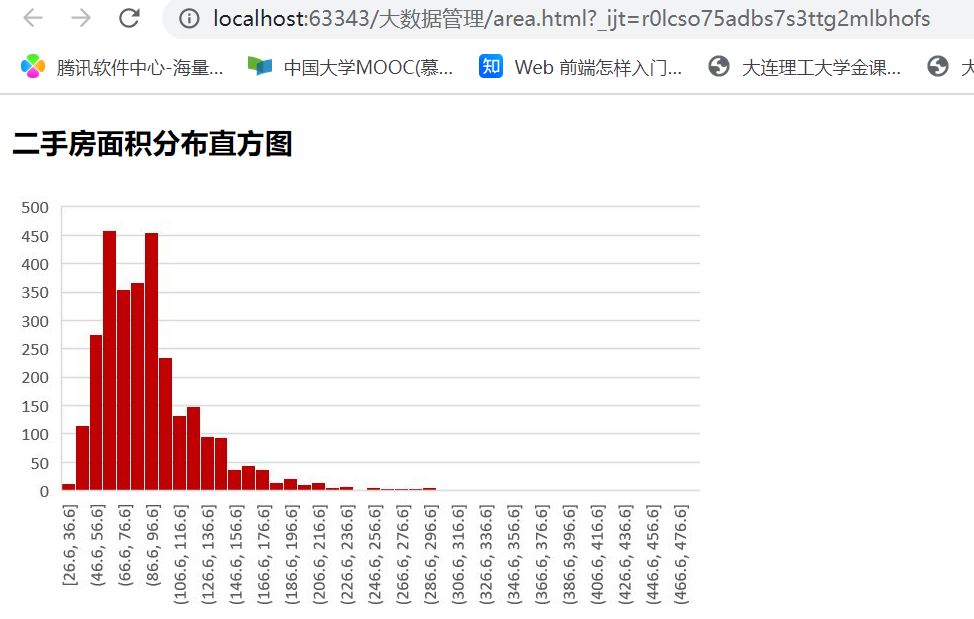
进行可视化

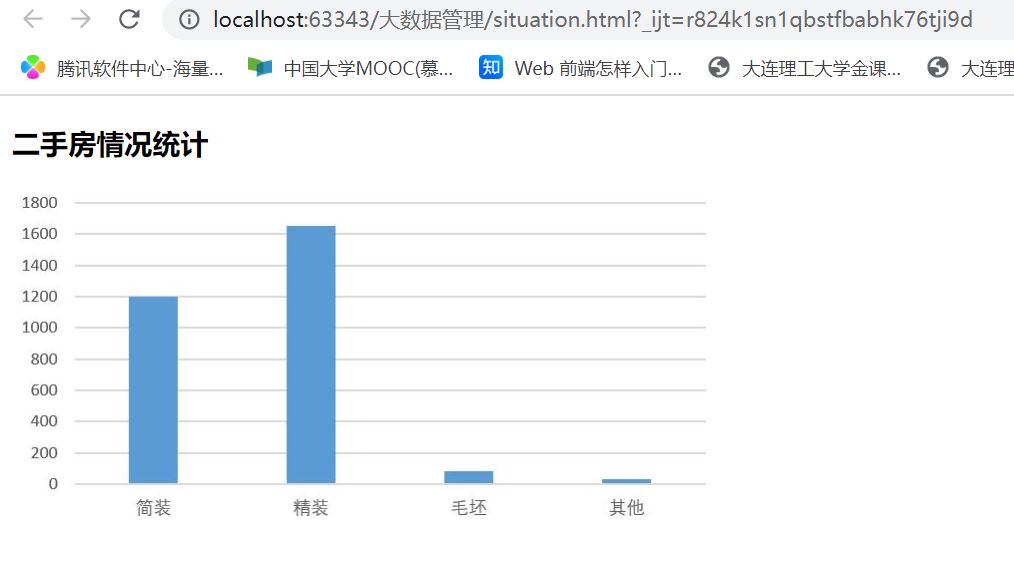
1. **def** drawChart():
2. data = []
3. **for** item **in** get\_db\_df():
4. js = json.loads(str(item).replace("\'",'\"'))
5. row = (str(js['unit\_type']), int(js['cnt']))
6. data.append(row)
8. c = (
9. Pie()
10. .add("", data)
11. .set\_colors(["blcak", "orange", 'green', 'blue', 'grey'])
12. .set\_global\_opts(title\_opts=opts.TitleOpts(title="热门户型"))
13. .set\_series\_opts(label\_opts=opts.LabelOpts(formatter="{b}: {c}"))
14. .render("output\_pic/result.html")
15. )

18. **if** \_\_name\_\_ == '\_\_main\_\_':
19. drawChart()

# 5 系统运行结果





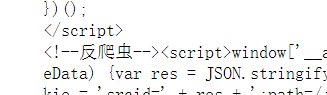


# 6 个人小结

通过这学期开源软件基础课程的学习，我们在任老师的带领下逐步认识了许多开源平台和开源软件，学习了一门新的计算机语言python，每节课都在老师的带领下用jupyter notebook学习使用不同的库进行代码开发，比如coverage和pysnooper。在这学期的课程中老师给我们播放介绍开源软件历史的视频，给我们推荐相关书籍，说一些知名人物的奇闻异事，介绍beautiful soup并做了一个案例，最后学习了Mongodb数据库。

本次大作业内容主要做的工作就是对链家二手房网站进行数据抓取，爬取下来的数据进行数据清洗后，使用 Spark大数据框架进行数据的分析，将分析出来的结果进行可视化的展示。

在爬取过程中，我们也遇到了一些意想不到的问题，比如，在第一次爬取链家网的二手房数据时，曾经遭遇了“反爬虫”的问题，在执行编写的爬虫代码后，代码并没有执行成功，而是输出了一段HTML源码，其中含有反爬虫的字样：

。在查阅资料后，我们得知要设置headers,在请求网页爬取的时候，相当于我们进去这个网页的服务器本身，假装自己本身在获取数据，就成功的爬取到了所需要的数据。

以前我们常用的语言是c++和java，现在我们又掌握了一门实用的语言python，这门语言和其他的语言相比更加轻量级，语法也更加灵活，通过这次大作业来爬虫能切身体会到这种语言的便捷之处。

完成数据爬取以及分析之后，本着open-source的原则，我们按照老师的要求上传所创建的开源项目。在此过程中，本小组由于不了解github网站的上传规则，遇到了不小的困难。经过查阅资料，我们得知产生问题的关键在于使用git bash进行项目的clone和push时，要正确地建立本地与github网站的连接。在身份验证时，小组成员习惯性地使用了最简单的账号密码形式，在阅读相关文献之后，改用Token验证，其与基于密码的身份验证相比具有许多安全优势，比如Token特定于github，可以按使用或按设备生成，可以随时单独撤销，而无需更新未受影响的凭据等等。这些经验，如果不亲自去做，是无法轻易得知和总结的。

在完成大作业的过程中，我们锻炼了python代码能力，能够从开源软件上爬取数据，学会了使用GitHub代码托管平台，将项目从本地上传到GitHub上，同时也培养了我们小组成员的协作能力，这些知识和通过实践积累的经验不管是对课内还是以后工作都是受益匪浅的。