Research on Data Collection and Analysis of Second Hand House in China Based on Python

Hejing Wu  
East University of Heilongjiang  
E-mail: 499917928@qq.com

Ran Cui  
East University of Heilongjiang

Abstract—With the rapid development of domestic Internet, more and more people choose housing leasing activities through the Internet, however, the existing housing rental information website the quantity many, but because of various advertising, information is scattered, so to HOME LINK sites, according to the position provided by the customer, accurate demand such as rent or house type, vertical search related data from the rental information website and in accordance with the provisions, the structure of the storage, rent information data to provide data for subsequent analysis. Bring users a faster experience. At the same time, we should add the reminding of the houses we are interested in and the reminding of price reduction. For the users who cannot look at their mobile phones all the time, timely reminding can help them avoid missing many desirable houses. This system is committed to solving the current people to rent a request for the search of detailed provide the keywords needed. To help make it easier for everyone who needs to rent. The system is mainly composed of data cleaning, data access, algorithm design and implementation, Python implementation of the front and back end of the system, data formatting and auxiliary decision.

Keywords-Python Crawler; Scrapy framework; Django Framework; HOME LINK

I. THE INTRODUCTION

With the rapid development of information technology in today's society, people's demand for information is also greatly increased. The society has gradually become a collection of information, and in this collection of information there are various data. The data is a form of information. In most cases, these data are hidden in the network, and these data are complex and diverse. It is very difficult to extract these complex data from the network by using our traditional processing methods, and to analyze and study to obtain useful information.

II. HOME LINK PLATFORM STATISTICAL SURVEY

This topic will choose HOME LINK mall property information platform as the crawler research object, through the crawl to the guest's regional secondary housing, the second-hand housing prices and in different parts of the room to crawl data, through the analysis of the data from different areas, different house type and price of second hand house information to extract useful information. By further exploring the practical process, some conclusions are drawn on the crawler and basic data analysis methods, and a summary is made.
III. RELATED TECHNOLOGIES AND FRAMEWORKS

In the process of system design, we mainly use Python+Django+Scrapy+WordCloud in the work technology. Scrapy is an open source web crawler framework written in Python. Scrapy was originally designed to grab the network, but also can be used as building data extraction of API or general web crawler Scrapy framework provides a series of efficient and powerful component, through Scrapy framework developers can quickly build a web crawler program, even if is a complex application can also be done through a variety of plug-in or middleware to build. The basics of the Scrapy framework are shown in Figure 1.

![Figure 1. Basic principles of Scrapy frame](image)

BeautifulSoup is a library for parsing HTML or XML text. BeautifulSoup handles ironical markup in HTML or XML text by generating a parse tree and provides an interface that allows developers to easily navigate the parse tree (Navigating), searching and modifying operations. Compared to other HTML/XML parsing tools, BeautifulSoup has the advantages of simplicity, high error tolerance, and developer friendliness.

WordCloud is a third-party library that takes the Wordcloud as an object. It can draw the Wordcloud with the frequency of words in the text as a parameter, and the size, color and shape of the Wordcloud can be set.

IV. DESIGN PROCESS

The Scrapy framework is mainly made up of four parts: items, spiders, pipelines, and middlewares. Based on the basic structure of Scrapy, the crawling tool of house rental information is divided into four modules, which are crawling data module, crawling module, configuration module and data processing module. In Items, you define the item entity to climb, while in this program you define LianjiaItem item, including price, orientation, location, name, floor, area, and layout of the house.

In Python, Django is a framework based on MVC constructs. In Django, however, the framework handles the part of the controller that accepts user input, so Django focuses more on models, templates, and Views, called MTV modes. Their respective responsibilities are as follows:

<table>
<thead>
<tr>
<th>Hierarchy</th>
<th>Responsibilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model and data access layer</td>
<td>Handle everything related to the data: how it is accessed, how it is validated, what behaviors are involved, and the relationships between the data.</td>
</tr>
<tr>
<td>Template that is the presentation layer</td>
<td>Handle presentation-related decisions: how to display in a page or other type of document.</td>
</tr>
<tr>
<td>View and business logic layer</td>
<td>The logic associated with accessing the model and fetching the appropriate template. Bridge between model and template</td>
</tr>
</tbody>
</table>

The crawler, which defines the crawling logic and the parsing rules for web content, is responsible for parsing responses and generating results and new requests. Crawler is the key point of this design. It defines how to grab items entities. Both the initial dynamic page connection and static page information crawl are defined in this file. The key code is as follows:
class LianjiaDataSpider(scrapy.Spider):
    pg_num = 1
    name = 'lianjia_data'
    start_urls = ['https://bj.lianjia.com/zufang/pg%d' % pg_num]
    def parse(self, response):
        addr = dict()
        list_data = response.xpath('/html/body/div[3]/div[1]/div[5]/div[1]/div')
        for i in list_data:
            data = i.xpath('./div')
            for j in data:
                link = j.xpath('./a/@href').extract_first()
                if link == '/apartment/35125.html':
                    continue
                p_1 = j.xpath('./a/@href').extract_first()
                if p_1 == '/apartment/35125.html':
                    continue
                p_2 = j.xpath('./div/p[2]/a[1]/text()').extract_first()
                p_3 = j.xpath('./div/p[2]/a[2]/text()').extract_first()
                p_4 = j.xpath('./div/p[2]/a[3]/text()').extract_first()
                f"{p_1}area{p_2}road{p_3}" = details_url
                title = data.xpath('./div[1]/div[3]/p/text()').extract_first().split()[0].split('.')[1]
                if not title:
                    return
                else:
                    title = data.xpath('./div[1]/div[3]/p/text()').extract_first().split()[0]
                    price = data.xpath('./div[1]/div[3]/div[2]/div[2]/div[1]/span/text()').extract_first()
                    layout = house_layout[0]
                    area_s = house_layout[1]
                    item = LianjiaItem()
                    item['name'] = title
                    item['price'] = price
                    item['orientation'] = orientation
                    item['position'] = position
                    item['floor'] = str(floor).split('/')[-1]
                    item['area_s'] = int(area_s.split('㎡')[0])
                    item['layout'] = layout
                    yield item
        data = response.xpath('/html/body/div[3]')
        try:
            title = data.xpath('./div[1]/div[3]/p/text()').extract_first().split()[0].split('.')[1]
            except Exception:
                title = data.xpath('./div[1]/div[3]/p/text()').extract_first()
                if not title:
                    return
            else:
                title = data.xpath('./div[1]/div[3]/p/text()').extract_first().split()[0]
                price = data.xpath('./div[1]/div[3]/div[2]/div[2]/div[1]/span/text()').extract_first()
                layout = house_layout[0]
                area_s = house_layout[1]
                item = LianjiaItem()
                item['name'] = title
                item['price'] = price
                item['orientation'] = orientation
                item['position'] = position
                item['floor'] = str(floor).split('/')[-1]
                item['area_s'] = int(area_s.split('㎡')[0])
                item['layout'] = layout
                yield item

HOME LINK platforms provide second-hand housing landlord page will all kinds of second-hand housing information platform (including price, house type, floor location, etc.) published on the web page, a list of your products to crawl at the beginning of the page load will only load 30 house for candidate, selenium can be used
for web to simulate human drop-down operations, but each time the drop-down will only on the basis of the original in the loading 30 entries, so use Scr num read out to crawl the total number of items of the project, divided by 30 and more down, so you can set drop-down list number according to the total number of entries, close the browser after reading.

The crawling process is shown in the figure:

A. Data analysis

1) General data analysis methods

Data analysis refers to the process of analyzing a large number of collected data with appropriate statistical analysis methods, extracting useful information and forming conclusions, and studying and summarizing the data in detail. Sometimes the resulting data needs to be further processed and extracted before it can be turned into useful information for people. Data analysis can help people make judgments so that they can take appropriate actions. The mathematical basis of data analysis had been well established as early as the 20th century, but it was not until the advent of computers that the practical operation of data analysis became possible and data analysis became widespread. So data analysis is a combination of mathematics and computer science.

Data visualization is one of the more representative aspects of data analysis. It is the trends that make data visible to human eyes. According to different needs, there are many methods of data visualization, ranging from training AI to deeply learn various patterns in data and make predictions, to analyzing basic functions in Excel sheets, which can all be the process of data analysis.

2) Key technologies and technical difficulties

a) Engine, processing the whole system of data flow processing, starting things, the core of the framework. Scheduler: The Scheduler accepts requests from the engine, queues them up, and
delivers them to the engine when it requests them again.

b) The Downloader downloads the web content and returns the downloaded content to the spider.

c) Item pipeline, the project pipeline, is responsible for processing the data extracted from the web pages by spiders, mainly responsible for cleaning, verifying and storing data in the database.

d) Downloading middleware DownLoader Middlewares. It's the processing block between Scrapy's Request and requestponse.

e) Spider Middlewares, Spider Middlewares, which is located between the Spider and the Spider, mainly handles the response of the Spider input and the result of the output and the new request MIDDLICWARespy.

f) Front-end and back-end connection: Since data needs to be stored in the database, the database and front-end connection need to be connected. The database connection pool is responsible for allocating, managing and releasing the database connection. It allows the application process to reuse an existing database connection. For data interaction with the back end, this article mainly uses Ajax, which is a small asynchronous framework of JavaScript and an interaction tool. The standard format of Ajax is as follows:

```javascript
$.ajax({
  url: '',//back end path,
  type: '',//Request method
  data: '', // Data sent to the back end
  success: function(data){
    // The operation after success
  }
})
```

- Using Echart to display data: In order to make the data look orderly, this paper adopts Echart to visualize the data and make the data more concise and objective.
- The data must be representative, and the amount of data must be large enough, otherwise it will not be convincing. The data of large size should be clear, and enough time and energy are needed to process the data during data preprocessing, otherwise problems may occur.

3) Data visualization processing results display

a) All data table status display, key code:

```python
def index(request):
    """
    Home page:
    Show all the information
    """

    if request.method == "GET":
        # Paging data
        page_object = Pagination(
            current_page=request.GET.get("page"),
            all_count=queryset.count(),
            base_url=request.path_info,
            query_params=request.GET,
            per_page=30,
        )
        context = page_object.page_html()
        data = queryset[page_object.start:page_object.end]
        return render(request, "home.html",
                       {'data': data, "page_html": context})
```
Figure 3. Full information display

\[ \text{b)} \quad \text{Data pie chart shows the key code} \]

```python
def Per_charts(request):
    """
The pie graph function--->By the block functions
:param request:
:return:
"""
    if request.method == "GET":
        if request.is_ajax():
            position_name = list()
            data = models.lianjia_data.objects.all().values("position")
            for i in data:
                positions = i['position'][:2]
                if positions not in position_name:
                    position_dict = dict()
                    position_dict['value'] = models.lianjia_data.objects.filter(position__startswith=f"{positions}").values()()
                    position_dict['name'] = positions
                    position_name.append(positions)
                    data_list.append(position_dict)
                return JsonResponse({"data": data_list})
            return render(request, "per_chart.html")
```
Figure 4. Pie Chart of rent per district

```python
def Per_charts_ori(request):
    """
    The pie graph function--By the orientation function
    """
    if request.method == "GET":
        if request.is_ajax():
            ori_name = list()
data_list = list()
data = models.lianjia_data.objects.all().values("orientation")
            for i in data:
                ori = i['orientation']
                if ori not in ori_name:
                    ori_dict = dict()
data = models.lianjia_data.objects.filter(orientation=ori).
                    values()
oridict['value'] = data.count()
oridict['name'] = ori
data_list.append(ori_dict)
ori_name.append(ori)
return JsonResponse({"data":
data_list})
```
c) **Data line chart display, key code:**

```python
def line(request):
    Line graph function
    :param request:
    :return:
    if request.method == "GET":
        if request.is_ajax():
            data_list = list()
            name_list = list()
            price_data = list()
            data_dict = dict()
            data = models.lianjia_data.objects.all().values("position", "price")
            for i in data:
                price = int()
                positions = i['position'][:2]
                if positions not in data_list:
                    data_list.append(positions)
                    price_list = models.lianjia_data.objects.filter(position__startswith=positions).values("price")
                    for price_i in price_list:
                        price += price_i['price']
```

Figure 5. Pie Chart of rental housing according to the ratio of each direction
name_list.append(positions)
price_data.append(price //
price_list.count())
return JsonResponse(data_dict)
return render(request, "line.html")

data_dict['name'] = name_list
return render(request, "line.html")

data_dict['price'] = price_data

The broken line chart of the average rent price in Beijing

Figure 6. The broken line chart of the average rent price in Beijing

d) Data bar statistics show, key code

```python
def Dot_chart(request):
    """
    Bar graph function
    :param request:
    :return:
    """
    if request.method == "GET":
        if request.is_ajax():
            data_list = list()
            name_list = list()
            name_count_list = list(
                data = models.lianjia_data.objects.all().values("orientation", "position")
            )
            for i in data:
                positions = i['position'][:2]
                if positions not in data_list:
                    count = models.lianjia_data.objects.filter(position__startswith=positions).count()
                    name_list.append(positions)
                    name_count_list.append(count)
                    data_list.append(positions)
                    data_dict = dict()
                    data_dict['name'] = name_list
                    data_dict['value'] = name_count_list
                    return JsonResponse(data_dict)
            return render(request, "Dot_chart.html")
```

for i in data:
e) **Word cloud display, key code:**

```python
def start_worldcloud():
    Read and write database files to write CSV files
    :return:
    with open("data/lianjia.csv", "w+", encoding="utf-8")as f:
        for i in queryset:
            for j in i:
                if j != "id":
                    f.write(str(i[j]))
                    f.write(" ")
                    f.write("\n")
    create_cloud()

def create_cloud():
    ""
    Make word cloud
    ""
    :return:
    ""
    cloud = open(r"data/lianjia.csv", "r", encoding="utf-8").read()
    worldcloud = WordCloud(
        background_color="white",
        width=1000,
        height=860,
        margin=2,
        font_path='SimHei.ttf'
    ).generate(cloud)
    plt.imshow(worldcloud)
    plt.axis("off")
    worldcloud.to_file(r'static/img/word_cloud.png')

def show_worldcloud(request):
    Word cloud page function: get Request return Page
    post Request to receive font-end parameters To obtain data from the database to prepare CSV production
    :param request:
    :return:
    if request.method == "GET":
        return render(request, 'cloud.html')
    if request.method == "POST":
        cursor = connection.cursor()
        data = request.POST.get("data")
        if data == 'all':
            start_worldcloud()
            create_cloud()
```

![Figure 7. The bar chat of Renting in Beijing](image)
return JsonResponse({"load":"/static/img/word_cloud.png"})

deal = str(data).replace("[",
"").replace("]", ").replace("", ").split(",
uc = str(tuple(deal)).replace("",
"").replace(", ", ").replace("", ")
cursor.execute(f"select {uc} from lianjia_data;")
detail = cursor.fetchall()
create_cloud()
return JsonResponse({"load":
"/static/img/word_cloud.png"})

Figure 8. Generating wordcloud presentation

V. CONCLUSION

Through the research and analysis of the second-hand housing data of HOME LINK in Beijing, this paper studies how to scrapy structure climb the rental information on the HOME LINK website, how to build scrapy structure, rental orientation, the location of what impact on the price of Beijing.

ACKNOWLEDGMENT

This paper is about the scientific research project of East University of Heilongjiang in 2019, "Implementation of Crawler Based on Python Scrapy Framework", project number HDFKY190109.

REFERENCES