Design of a Python-Based Data Crawling System—Taking House Information Crawling as an Example

Hongxia Mao

School of Computer and Software, Jincheng College of Sichuan University, Chengdu 611731, Sichuan, China

Abstract: ***The widespread application of Internet technology has led to an explosion of online resources, making it extremely time-consuming and labor-intensive to locate desired data within massive datasets. Housing information is one of the hot topics of national concern; by employing web crawler technology, housing information from major platforms can be obtained quickly and accurately. This paper designs a housing information data crawling system using Python combined with crawler technology, creating modules such as a URL manager, web page downloader, web page analyzer, data collector, and data saver. Through system operation, housing information and images from the target website were successfully saved.***

Keywords: Python; Data crawling; Anti-crawling strategies.

1. INTRODUCTION

With the rapid development of Internet technology, information technology has advanced swiftly, especially the data resources on the Internet, which have grown and accumulated at an enormous rate. The explosive growth of online resources makes it increasingly difficult to quickly and accurately find valuable information within massive datasets. Therefore, web crawlers have emerged, capable of accurately and efficiently extracting required data from target websites according to specific needs. Data crawling imposes some burden on websites, so different sites have adopted corresponding anti-crawling strategies. A data crawling system must periodically analyze the target website to study its anti-crawling mechanisms, ensuring the system can operate normally and obtain the required data.

At present, housing information is a key topic of public concern; people are highly interested in the prices of new homes, second-hand homes, and rentals. However, major platforms have data silos and cannot cover all housing information, so building a system capable of crawling housing data from the web is particularly important. This paper uses Python as the programming language for the data-collection system to scrape housing information online. The scientific use of big data technology can reflect the advanced nature of computer information methods at the same time to strengthen the effectiveness of information security management, making the work carried out more smoothly, to provide reliable protection for computer information security. Tu (2025) introduced Log2Learn, an intelligent log analysis framework for real-time network optimization [1]. In recommendation systems, Wang (2025) proposed a joint training method for propensity and prediction models using targeted learning to handle missing-not-at-random data [2]. Healthcare AI innovations include Ding and Wu's (2024) systematic review of self-supervised learning for ECG/PPG signal processing [3], and Restrepo et al.'s (2024) multimodal deep learning approach with vector embedding alignment for low-resource healthcare settings [4]. For NLP, Yang et al. (2025) developed a GAN-based extractive text summarization model combining transductive and reinforcement learning [5]. Visual analytics advancements feature Xie and Chen's (2025) CoreViz, a context-aware reasoning engine for BI dashboards [6]. Zhu (2025) presented TraceLM for temporal root-cause analysis using contextual embedding language models [7], while Zhang (2025) addressed deployment safety with SafeServe for multi-app monetization platforms [8]. In digital content, Hu (2025) created UnrealAdBlend for immersive 3D ads via game engine pipelines [9]. Sustainability research by Wu et al. (2025) linked supply chain digitalization and energy efficiency to carbon neutrality [10]. Domain adaptation techniques include Peng et al.'s (2022) RAIN framework for black-box scenarios [11]. Anomaly detection is extensively explored: Zhang et al. (2025) investigated ML-based methods for biomechanical big data [12]; Wang (2025) built a knowledge graph-based clinical trial anomaly detection system [13]; Qi (2025) designed interpretable slow-moving inventory forecasting with hybrid neural networks [14]; and Huang and Qiu (2025) implemented LSTM-based abnormal electricity usage detection in smart meters [19]. Chen (2023) demonstrated data mining applications in general data analysis [20]. Low-code/SME digitalization was advanced by Fang's (2025) microservice-driven modular platform [15], while spatiotemporal

analytics featured Li's (2025) GIS-integrated U-Net for automated land encroachment detection [16]. Lin (2025) examined generative AI's role in proactive incident management [17], and Li (2025) optimized emergency response with AD-STGNN for urban fire vehicle dispatch [21].

2. WEB-CRAWLER PRINCIPLES

A web crawler is a program whose main purpose is to download web pages from the Internet to the local machine and extract relevant data. It can automatically browse information on the web and, according to specified rules, download and extract the required data. The architecture of a basic web crawler is shown in Figure 1.

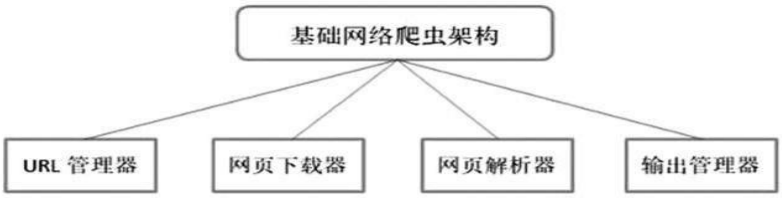

Figure 1: Architecture of a Basic Web Crawler

URL Manager: Manages the URLs to be crawled, preventing duplicate and circular crawling.

Web Downloader: The component that downloads web pages, used to fetch the web page corresponding to a given URL from the Internet to the local machine; it is one of the core parts of the crawler.

Web Parser: The component that parses web pages, used to extract valuable data from them; it is another core part of the crawler.

Output Manager: The component that stores information, used to output the parsed content to files or a database.

3. HOUSING-DATA CRAWLER SYSTEM DESIGN

This paper targets Q for housing-data crawling. The crawler system will collect the site links for each city on Q, along with basic information on second-hand homes, rentals, and new homes, and will save the scraped housing data.

3.1 Web URL Management Analysis

Open Q in Google Chrome, enter the second-hand home section, click on the location to obtain the URL for the corresponding city, then try switching cities. Observation shows that the URLs for second-hand home pages follow a pattern, as shown in Figures 2 and 3:

Figure 2: Beijing Region URL

Figure 3: Foshan Region URL

When the city is changed, the site's URL pattern is found to be: https://city-pinyin.qfang.com/sale.

Each city page on Q displays at most 30 housing listings. Clicking “Next Page” changes the URL according to the following rule: after entering a city, the URL is http://foshan.qfang.com/sale,

Clicking on page 2, the URL is: http://foshan.qfang.com/sale/f2. As shown in Figure 4.

Figure 4: Page 2 of Foshan property listings

Therefore, the construction rule for the URL of page n is:

http://foshan.qfang.com/sale/fn.

Thus, the code for paginating the property listing URLs on Q Fang is:

```
pre_url = ’http://foshan.qfang.com/sale/f’
for x in range(1,11):
    url = pre_url + str(x)
```

3.2 Web Page Download

Requests is a library commonly used when writing crawler code. Requests inherits all features of urllib2. It supports HTTP persistent connections and connection pooling, session persistence via cookies, file uploads, automatic determination of response content encoding, internationalized URLs, and automatic encoding of POST data. The code for downloading web pages using Requests is as follows:

```
pre_url = ’http://foshan.qfang.com/sale/f’
for x in range(1,11):
    url = pre_url + str(x)
    html = requests.get (url, headers=headers)
```

3.3 Web Page Parsing

Using Chrome’s developer tools, you can locate the data to be scraped within the page source, extract the XPath path, as shown in Figure 5.

Figure 5: Front-end code for website housing information

Call the spider function to obtain the housing information for the corresponding page; the code is shown in Figure 6:

```
def spider(url):
    ...crawler function...
    selector = download(url)
    house_list = selector.xpath("//#[@id='cycleListings']/ul/li")
        xiaoqu = house.xpath('div[1]/p[1]/a/text()')[0]
        huxing = house.xpath('div[1]/p[2]/span[2]/text()')[0]
        mianji = house.xpath('div[1]/p[2]/span[4]/text()')[0]
        weizhi = house.xpath('div[1]/p[3]/span[2]/a[1]/text()')[0]
        #construct the detail page URL
        house_url = ('http://beijing.qfang.com'
                    + house.xpath('div[1]/p[1]/a/@href')[0])
        sel = download(house_url)
        house_year = sel.xpath("//div[@class='housing-info']/ul/li[2]/div/ul/li[3]/div/text()")[0]
        mortgage_info = sel.xpath("//div[@class='housing-info']/ul/li[2]/div/ul/li[5]/div/text()")[0]
        #construct the data items to be written to the file
        <item = [community, layout, area, location, total_price, house_year, mortgage_info]</item>
        ##Write to file
        data_writer (item)
        print('Fetching', xiaoqu)
```

Figure 6: House information scraping code

3.4 Output Saving

Save the scraped housing information to a CSV file named afang_foshan, as shown in Figure 7 below.

```
def data_writer(item):
    writer.writerow(item)
        ## ##
```

Figure 7: Code for Saving Housing Listing Information

Save the house image information in binary format, as shown in Figure 8 below.

```
def image_saver(url, xiaoqu):
    Image Saving Function
    param url: image web page URL
    param xiaogy: image neighborhood name
    :return: None
1 ~5
    img = requests.get (url, headers = headers)
    with open('./Qfang_image/{}.jpg'.format(xiaoqu),'wb') as f:
        f.write (img. content)
```

Figure 8: Code for saving listing images

4. ANTI-SCRAPING COUNTERMEASURES

Websites often implement anti-scraping mechanisms against web crawlers; the most common techniques are Header-based anti-scraping and user-behavior-based anti-scraping.

Header-based anti-scraping is the most prevalent strategy. Many sites inspect the user-agent in the Headers. To counter this, the crawler code in this paper adds Headers, assigning the browser's user-agent to the crawler's Headers so the server can identify the client's OS and version, CPU type, browser and version, rendering engine, language, plugins, etc., thereby bypassing the anti-scraping measures.

The target site detects user behavior—e.g., the same account performing identical actions repeatedly in a short time—to identify crawlers. To address this, the crawler code introduces a random delay of several seconds

between requests.

5. CONCLUSION

This paper studies the design of a data-scraping system using Python. By leveraging crawler technologies and implementing the corresponding code, it scrapes housing data from the Q website. From URL management, page downloading and parsing, data extraction and storage, image saving, to handling anti-scraping mechanisms, the entire data-scraping system is realized.

REFERENCES

[1] Tu, T. (2025). Log2Learn: Intelligent Log Analysis for Real-Time Network Optimization.
[2] Wang, Hao. "Joint Training of Propensity Model and Prediction Model via Targeted Learning for Recommendation on Data Missing Not at Random." AAAI 2025 Workshop on Artificial Intelligence with Causal Techniques. 2025.
[3] Ding, Cheng, and Chenwei Wu. "Self-Supervised Learning for Biomedical Signal Processing: A Systematic Review on ECG and PPG Signals." medRxiv (2024): 2024-09.
[4] Restrepo, David, et al. "Multimodal Deep Learning for Low-Resource Settings: A Vector Embedding Alignment Approach for Healthcare Applications." medRxiv (2024): 2024-06.
[5] Yang, Jing, et al. "A generative adversarial network-based extractive text summarization using transductive and reinforcement learning." IEEE Access (2025).
[6] Xie, Minhui, and Shujian Chen. "CoreViz: Context-Aware Reasoning and Visualization Engine for Business Intelligence Dashboards." Authorea Preprints (2025).
[7] Zhu, Bingxin. "TraceLM: Temporal Root-Cause Analysis with Contextual Embedding Language Models." (2025).
[8] Zhang, Yuhan. "SafeServe: Scalable Tooling for Release Safety and Push Testing in Multi-App Monetization Platforms." (2025).
[9] Hu, Xiao. "UnrealAdBlend: Immersive 3D Ad Content Creation via Game Engine Pipelines." (2025).
[10] Wu, W., Bi, S., Zhan, Y., & Gu, X. (2025). Supply chain digitalization and energy efficiency (gas and oil): How do they contribute to achieving carbon neutrality targets?. Energy Economics, 142, 108140.
[11] Peng, Qucheng, et al. "RAIN: regularization on input and network for black-box domain adaptation." arXiv preprint arXiv:2208.10531 (2022).
[12] Zhang, Shengyuan, et al. "Research on machine learning-based anomaly detection techniques in biomechanical big data environments." Molecular & Cellular Biomechanics 22.3 (2025): 669-669.
[13] Wang, Y. (2025, May). Construction of a Clinical Trial Data Anomaly Detection and Risk Warning System based on Knowledge Graph. In Forum on Research and Innovation Management (Vol. 3, No. 6).
[14] Qi, R. (2025). Interpretable Slow-Moving Inventory Forecasting: A Hybrid Neural Network Approach with Interactive Visualization.
[15] Fang, Z. (2025). Microservice-Driven Modular Low-Code Platform for Accelerating SME Digital Transformation.
[16] Li, B. (2025). GIS-Integrated Semi-Supervised U-Net for Automated Spatiotemporal Detection and Visualization of Land Encroachment in Protected Areas Using Remote Sensing Imagery.
[17] Lin, Tingting. "The Role of Generative AI in Proactive Incident Management: Transforming Infrastructure Operations."
[18] Huang, Jingyi, and Yujuan Qiu. "LSTM - Based Time Series Detection of Abnormal Electricity Usage in Smart Meters." (2025).
[19] Chen, Rensi. "The application of data mining in data analysis." International Conference on Mathematics, Modeling, and Computer Science (MMCS2022). Vol. 12625. SPIE, 2023.
[20] Li, Binghui. "AD-STGNN: Adaptive Diffusion Spatiotemporal GNN for Dynamic Urban Fire Vehicle Dispatch and Emergency." (2025).