Scrapy-based Incremental Housing Rental Information Crawling System Design

Qichen Shao[1], Dongxiao Ren[2]

[1]School of Science, Zhejiang University of Science and Technology, Hangzhou 310023, China
[2]This work was supported in part by Zhejiang University of Science and Technology College Student Innovation and Entrepreneurship Training Plan Platform Project under Grant 2011-CXCY158.

Abstract
A web-controlled incremental crawling system is designed for incremental crawling of property rental information on websites because of the need for massive data sets to train the housing rental system model, and to solve the problem of always using site-wide crawling and multiple database accesses for crawling websites based on the Scrapy framework. In order to achieve incremental crawling, a download middleware is added to the Scrapy framework, the system loads the seed page, the visited URLs and their hash lists and the control page list when the crawler starts, obtains the URLs of the sub-level pages and enters them into the database, then crawls the sub-level pages in bulk and parses the property information in the sub-level pages. The data is cleaned by verifying the data format, completing missing items, removing duplicate data and detecting abnormal data to get the eligible property data.

Keywords: Scrapy crawler; incremental crawling; download middleware;

1. Overview
A large data set of rental amounts is required to train a machine learning based rental assessment model in order to provide a more objective and realistic representation of the assessment results. Rent data is influenced by time, location, property attributes and price. In addition to the specific rental data that can be found in the transaction records and advertised prices, the data set can also be obtained by searching or crawling through rental websites. The system is designed to crawl data from housing rental websites, extract data, filter data and integrate encapsulated data to serve as the dataset for the evaluation system.
In order to avoid duplicate crawling of part of the website content when doing site-wide crawling, the website is crawled incrementally [1], incremental crawling means that when doing site-wide crawling of the website only the new and changed parts of the content are crawled [2], which can significantly reduce the pressure of accessing the crawled website, shorten the crawling time of the crawler and at the same time reduce the storage space requirements of the crawler system.
Scrapy[4] is a mature Python[3] based open source web crawler framework that provides a clean and powerful way to build and deploy web crawlers. Using it to develop web crawler software systems [5] can reduce the development of repetitive software function modules such as treating crawl queues and their access, and process control of the crawling process. Therefore, the design of an incremental housing rental information crawling system based on Scrapy has practical engineering value.

2. System working principle
2.1 Crawling seed pages
Crawling seed pages is done to get the URL of the seed page and put it in the database in preparation for getting the entity information. The HTML source code of the seed page of the target website is viewed in the following format: Depending on the selector provided by Scrapy, elements can be selected by CSS expressions or XPath expressions. In a practical way, XPath expressions XPath expressions are more suitable in terms of their capabilities and are the basis for Scrapy selectors.

Given the common situation where the seed pages are page-flipped, it is possible to get the XPath expression of the "Next" button to extract the URL for loading into the scheduler after getting the information of the current page. We can also test that the "next page" and "last page" HTML have the same code pattern and the same XPath expression. The extracted URLs need to be verified and corrected.

Pattern check: detects if the URL has a suffix of ".html" or ".htm", or if the URL is a relative address, and needs to be filled in with the missing address.

Repeatability check: queries the database to see if the link already exists.

2.2 Crawling sub-level pages

The sub-level page is the target page of the desired crawl information, mainly displaying the specific information of the rental property. This includes: time of release, district, name of the neighborhood, type of flat, size of the flat, door number, price, orientation, floor/total floor, decoration, lift, age of the property, etc. It is to be expected that the information displayed on the website is not exactly the same, and even the information displayed on different listings on the website may not be complete, so data cleaning needs to be carried out at a later stage, with non-critical information missing being filled in, and critical information missing being discarded.

2.3 Data cleansing

The purpose of data cleansing is to ensure that the data is consistent and usable for subsequent operations.

(1). The HTML language is essentially a plain text language and the content obtained is in string format, so it is particularly important to transform the data type and verify the validity of each piece of information for outlier processing. For example, for 'posting time', a time interval is set for the information extracted from the website and is considered valid if it meets the interval. Premature invalid posting times and incorrect posting times that are overdue are removed. The "District" field, which matches a string in the pre-set district database, is considered compliant if it matches; "House Size", "Price", and The "floor/total floors" should be checked to see if they are legal numbers.

(2). Missing item processing, according to the specific field meaning, such as key parameters "district", "district name", "price" is missing to clear the data items, such as non-key parameters if the fields "release time", "house type", "house area" are empty, then take as None.

(3). Before packaging into the database, repeat the value processing. Because it often happens that the same property is registered for rent on more than one housing rental website, it is necessary to compare the key information to confirm whether the property belongs to the same property with the property in the database before encapsulating it into the database. If the key parameters "district", "block name" and "door number" are the same, and if the parameters "price" are not the same, then the property will be entered into the database, otherwise, it will not be entered into the database.

3. Existing research

Scrapy now supports incremental crawling by using the Scrapy command line with arguments to Hash the URLs and request parameters of the crawled pages [6] and store them persistently in a specified directory. Scrapy does the same hash operation on the URLs and request parameters of the crawled web pages starting from the seed page, and then compares the results with the persisted hash operation to determine if there are any updates, and continues to download if there are any updates. However, in real-world engineering applications, there is a need for more granular control when determining whether there are updates to the content of a web page, controlling the size of the crawled pages, etc. Current research on incremental crawling functionality has been implemented in two ways：

(1). Scrapy only uses the URL crawled on the seed page to determine if a page has been updated, for example, when the content of a sub-level page has changed but the content of the seed page has not changed, this approach will not solve the problem.
(2). Scrapy performs site-wide crawling and then performs reweighting, but this situation will result in data redundancy and a significant increase in storage pressure and crawler stress, and is suitable for target pages with little web content.

The established approach cannot achieve a balance between perceptual capability and spatial redundancy. Therefore it would be of practical importance to develop and design incremental crawling functionality, and in this paper we will design a three-level approach to incremental crawling of web page control

4 System design

4.1 System flow

The goal of the system is to crawl housing rental information on rental websites, clean and filter the data and integrate it in the database. Objectives of the crawl:
(1). Set the URL of the seed page as a list of seed pages.
(2). To crawl seed pages and sub-level pages formatted using the MD5 algorithm and recorded in the visited list, and to unify them in the Hash list to record whether they are duplicates;
(3). Decode the URL addresses to be put into the visited list to obtain the URLs, crawl and crawl the property details.

4.2 System weighting

Because one of the key issues in determining whether the current page to be crawled is a new page for crawl control is by the form of the URL. However, since the length of the URL varies, and URLs with access parameters can be as many as hundreds of characters, while URLs without parameters may be as short as about 20 characters, it is a good option to apply the MD5 algorithm to map the URL to a string of hexadecimal letters fixed at 32 characters in length. The MD5 algorithm results are almost conflict-free.

The process is that when the web page is crawled, the MD5 algorithm is used to calculate the URL into a 32-bit hexadecimal string, and a Select query is performed from the database list to determine if the web page already exists, and when it does, the page is inserted into the database list. Then the Item properties and data are extracted.

However, there are two problems with this approach.
(1). Inevitably, the whole site is crawled, either in the spider class parse() method or in the Pipline object to do the URL reweighting, the page has been downloaded by the spider.
(2). In the operation of sentencing, each sentencing requires a Select database query operation. When the contents of the database are large, the Select operation will take more time and memory. The fact that Scrapy is a multi-threaded crawl means that the requirements of the operation are multiplied by a factor of one.We have improved this operation by:
- improve all crawling to incremental crawling
- use caching and no longer perform SelectSQL operations on the database.

The best place for the download middleware is between the engine and the downloader, because if it is placed between the engine and the scheduler, then the problem of downloading pages will not be solved.

This is because there will be a situation where the internal content changes, even though the URL has not changed. After we have judged the weighting based on the URL, we will determine if the pages are duplicated based on their content. This is done by comparing the lengths of the recorded pages with the lengths of the recorded pages to determine if the pages have been updated. If there is a change in the length of the web page, then the page has been updated, otherwise it has not.

4.3 Scale control
Since the process of determining duplicates by the length of the page content requires the download of the page, we need to consider the size of the control while considering the efficiency.
The size of the crawl is controlled according to the seed page. The list of seed pages of the site to be crawled is used as a control list. As the number of seed pages to be controlled is small, they can be loaded into the cache.

The URL of the seed page is the first page, i.e. the seed page is the first level page, and the sub-level page of the seed page is the second level page, and so on. When you start the crawl, you load the page control list from the database to get the seed page. The Process_request() method of the download middleware is executed before each page is crawled, so it is crucial to determine whether the URL of the currently crawled page is in the control list of pages. This is done by comparing the URLs recorded under each element in the control list, and if one element can be found then it is in the control list. If it is in the control list of pages, download the page first and then observe if the content of the page has changed. If the content of the page has changed, the control list and the database are updated simultaneously, and none is returned. Because if None is returned in the process_request() method, Scrapy will continue to download the current page and parse the new page to be crawled. If there are no changes to the content of the page, then an empty Response object is returned. Why return an empty Response object? Because if a Response object is returned in the Process_request() method, Scrapy will not download the current page and the crawler's parse() will get the empty Response object so that it can control the crawl from continuing. If it is not in the page control list, it will do a page reassessment. If duplicated, a Response object with empty content is returned; if not duplicated, none is returned.

5 System optimization
When a page is dynamically rendered by JavaScript, downloading the page and viewing the HTML source code reveals an "empty shell" with no real content. (1) Pre-rendering JavaScript; (2) Using a headless browser.) Using a headless browser is more compatible, but less efficient, while using a pre-rendering JavaScript solution allows multiple pages to be processed in parallel, which is more efficient.

Scrapy-Splash is a solution to the JavaScript rendering problem provided by the official Scrapy team, Splash is the module that handles the rendering of web pages, it uses the open source Webkit browser engine internally and uses the rendering service via the HTTPAPI. The web page request is handled in Scrapy via the downloader, which actually requests the Splash interface and gets the rendered data.

6 Conclusions
This incremental was created on demand, in the crawl control strategy to crawl the current web page to judge the weight, to solve the problem of always crawling the whole site and crawl efficiency is low, and in the Scrapy framework placed in the download middleware, and in the control of the scale at the same time, to increase efficiency and reduce the burden. It also works well for extracting dynamically rendered pages in JavaScript.

References
1. Jinmei Xu. Simulation of incremental intelligent crawling of channel information based on window queues [J]. Computer Simulation, 2019, 36(11):190-194.
2. Yu Menghui. Real-time analysis and system implementation of university network public opinion based on Spark [D]. Hangzhou:Zhejiang University of Technology and Industry,2020.

3. Wang Zhao. Python language programming features and applications [J]. Computer Programming Skills and Maintenance, 2021, (03) : 19-20.
4. Shanshan Wang. Topic crawling based on incremental Bayesian algorithm [D]. Nanjing:Nanjing Nanjing:Nanjing University,2018.
5. Deng Ziyun. Design and implementation of a Scrapy-based crawler system for logistics information website group implementation [J]. Logistics Technology and Applications, 2020, 25(8):140-143.
6. Chen L, Ma J. A Scrapy-based algorithm for monitoring the dynamics of agricultural cyberspace information method [J]. Journal of Shandong Agricultural University (Natural Science Edition), 2020, 51(2):253-258.