Film Comment Collection Technology and Realization of Distributed Web Crawler Based on Python

Chao Liu, Nana Nie

Jiangsu University, Zhenjiang, Jiangsu, 212013, China

Keywords*:* Python; Web Crawler Technology; Data Extraction and Processing

Abstract: Crawler technology is one of the important ways to obtain data efficiently and accurately in the current Internet environment, so it is widely used in various Internet industries. It mainly uses the Python language to access and crawl the HTTP hypertext protocol, URL address and so on in the Web page to complete the automatic crawling of the data information in the website. In the face of large-scale data capture requirements, in order to improve the performance of the overall Web crawler system, distributed technology is needed. This paper proposes a model based on a distributed Web crawler, with Douban as the experimental object. This model effectively improves URL normalization and reduces data repetition rate. The program running results show that the crawler model can accurately crawl more than 100,000 data on the Web page and avoid the problem of duplicate items, so that it can effectively extract massive resources for machine learning and recommendation system experiments.

1 Introduction

With the advent of the information age and the rapid development of Internet-related technologies, Internet users have entered the era of information overload. Due to information overload, it is difficult for users to quickly find information that is of interest or valuable to themselves in the face of massive amounts of information. The vertical content platform is produced in this context. At present, accurate Internet products have been increasingly used in various industries of the Internet. Among them, various vertical content communities have also developed, mostly related to users' daily lives. Regarding the research and integration of such data, obtaining high-quality content is of great value and significance. For accurate data crawling, the Python language has a unique advantage. There are many crawler frameworks in Python. Writing with the Python language can capture data more efficiently. Python is an object-oriented interpreted high-level programming language. Its syntax is simpler and more efficient than other programming languages', and it is easier to understand and implement. Therefore, using the Python language to achieve data capture is a good choice.

According to the technical form, Web crawlers can be divided into the following types.

Single-machine crawlers are usually used when the overall data volume is not large, and are generally used for directional crawling of small data.

Scalable Web crawlers' crawling objects expand from some seed URLs to the entire Web, and they mainly collect data for portal site search engines and large Web service providers.

Cloud crawlers are generally provided for people who do not need to develop reptiles, and are highly customized crawlers [1].

Distributed crawlers refer to building a website crawler on a cluster composed of multiple nodes to achieve multiple crawling in parallel to improve crawling efficiency horizontally. How to coordinate multiple nodes in the cluster, how to allocate tasks, and how to deploy are all problems that need to be solved. There can be different communication methods between each node, thus forming different distributed crawler structures, which are mainly divided into master-slave modes, autonomous modes and mixed modes [6].

This design aims at the problem of diversity and sample complexity of Douban, and adopts a distributed scrapy framework to crawl website data. Scrapy is an asynchronous processing

framework based on Twisited. It is a crawler framework implemented through Python and is a fast advanced Web crawling framework [5]. It has a clear architecture, a relatively low degree of coupling between modules and strong scalability, and it can flexibly meet various requirements [7]. Its composition framework is shown in Figure 1.

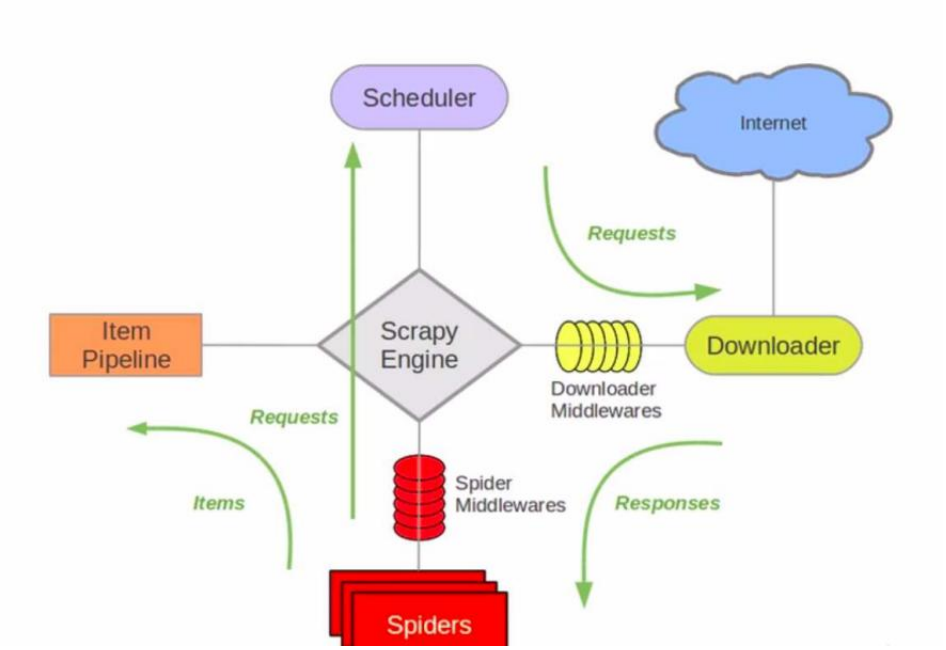

Figure 1. The flow chart of scrapy framework

Scrapy is mainly composed of 8 parts.

The engine: it is responsible for communication, signal, data transmission, etc. between Spider, ItemPipeline, Downloader, Scheduler.

The item: it defines the data structure of the crawling result. The crawled data will be packaged as an Item object.

The scheduler: it is responsible for accepting the requests sent by the engine, and sorting and arranging according to a certain way, entering the queue, and returning to the engine when the engine needs it.

The downloader: it is responsible for downloading all requests sent by scrapy engine, and returning the obtained responses to the scrapy engine, which is handed over by the engine to the spider for processing.

The spider: it is responsible for processing all responses, analyzing and extracting data from it, obtaining the data required by the item field, submitting the URL that needs to be followed to the engine, and entering the scheduler again,

The item pipeline: it is responsible for processing the items obtained from the spider, and performing detailed analysis, filtering, storage, etc. [9].

Downloader middlewares: components for customizing extended download functions.

Spider middlewares: functional components that can extend and manipulate the communication between the engine and spider (such as responses into the spider; and requests from the spider)

2. System requirements analysis

The system uses the coordination of multiple components of the scrapy framework to provide good support for asynchronous processing and maximize the use of network bandwidth. The aim is to crawl page data as much as possible and improve data processing capabilities. Under such demand, it is necessary to develop a crawler system suitable for large-scale, distributed and stable data crawling. In general, the system needs the following points.

2.1 Development of crawling strategy

The target website has a strong anti-reptile mechanism, so a variety of measures and strategies need to be taken to deal with the anti-reptile mechanism. The method adopted by this system is to simulate User-Agent, construct proxy IP pool, design cookies middleware and so on [3].

2.2 The business logic of the crawler

Analyze the page situation and data loading process of the target website, write appropriate business logic rules, and process the captured data according to the project requirements and store them in the database.

2.3 De-emphasis

The traditional de-emphasis method is to directly store the requested information in the collection and then judge, which consumes a lot of storage space. Therefore, a more memory-saving de-emphasis algorithm is needed to improve space utilization. According to the relatively large URL of the target website, consider using the Bloom Filter algorithm.

2.4 Large-scale distributed crawling

Under the conditions of established physical resources, the crawling efficiency can be accelerated by adjusting the scheduling structure. According to the page situation of the target website, analyze the page loading logic and extract item information in the light of the established requirements [4]. Therefore, the function of this part is mainly to realize the crawler's content analysis of the target Web page. The sub-link request creates and extracts the required data nodes and delivers them to the item pipeline. After a comprehensive analysis, the detailed functions involved in the crawling process include simulated login, asynchronous data request, callback function binding, response data analysis, sub-link extraction and new request construction, item data extraction and other functions.

2.5 Data format and storage

The captured data needs to be cleaned, formatted and so on to conform to the established demand format, and the data items are stored in the database.

In the working architecture of the scrapy, these parts are more important: scheduling layer, pipeline layer, business logic layer, middleware layer, service layer. There is a master node in the scheduling layer, which is used to control the task queue and the Bloom filter bit array. The pipeline module communicates with the database. For the crawled information, after being cleaned, formatted and so on, the data will be persistently stored in the MongoDB database [11]. The crawler module mainly analyzes and extracts the page data, and it extracts further sub-URL requests and puts them in the scheduling queue. The scheduler module and pipeline module will communicate with the Redis cache server to implement a series of operations such as maintenance of the Request queue and de-emphasis. The service layer provides services such as proxy pool service and cookie pool, which are used to connect to the middleware layer.

3. Detailed system design and implementation

The main task of the distributed crawler system is to crawl the data on Douban. After detailed analysis of the system function modules, the structure is shown in the following figure. Among them, the main engine module and the downloader module can already directly use scrapy’s original modules that have been implemented. The downloader is specifically responsible for downloading data from the target website, and the main engine is used to specifically coordinate and control the data processing flow between each module [10]. For this article, the main modules include Douban crawler module, middleware module and scheduler module.

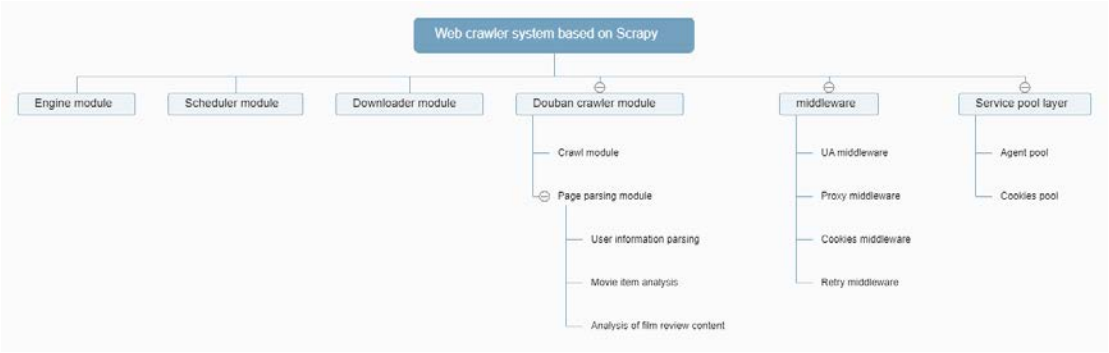

Figure 2. Crawler system function diagram

In addition, this system uses MongoDB database to save the data. Therefore, designing a reasonable table structure has become a relatively critical part of the system. For the Douban movie category, the data includes user information analysis, movie category analysis and movie comment content analysis. These fields include user ID, movie ID, users' rating for the movie and comment time, as shown in Table 1. After the data is acquired, it is processed to clear invalid fields, invalid scores and invalid mood words in the content. Desensitize users' privacy.

Table 1. Crawler key field analysis

Fields	Explanations
userID	user ID (starting from 0, consecutive numbers)
movieId	movie ID in movies.csv
rating	rating, an integer between [1,10]
timestamp	rating time
recommend	comment information

The large-scale crawling of Douban movie categories is to start crawling according to a single movie. First of all, Douban's movie information and comment are a lot, so we choose the list of movies as a starting point and grab the movie's name, rating, time, movie comment content and so on to achieve recursive crawling.

The basic code framework is shown in Figure 3.

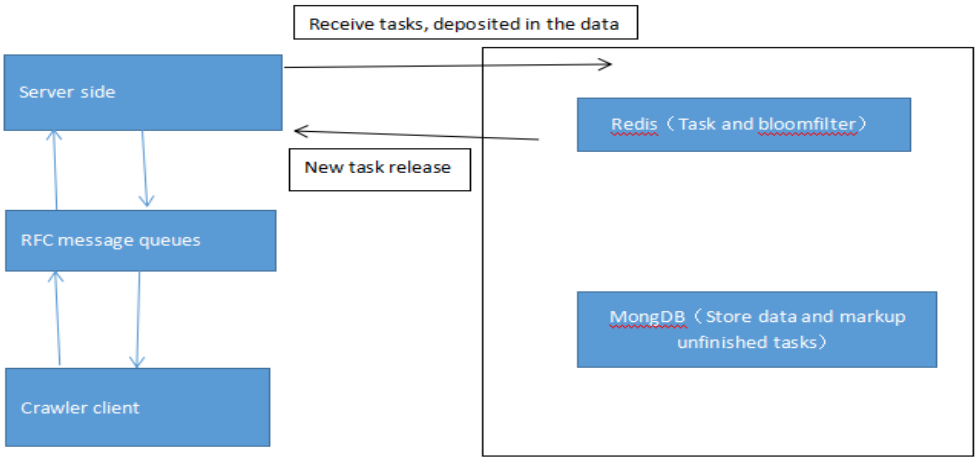

Figure 3. System frame diagram

After the system requirements are analyzed and the process framework of the entire system is understood, the detailed design and implementation of the important modules of the framework are carried out and the specific functional structure is explained.

3.1 Anti-reptile response module

After the early crawling of a single IP address, Douban's anti-reptile strategy is as follows.

When crawling without cookies, the IP address will be blocked after multiple consecutive requests. That is, no matter how user-agent is changed, a 403 error will be returned.

When using cookies, the first request for Douban page will return a cookie named bid, and subsequent requests will bring this cookie, but when the requests are too frequent, this cookie will also be blocked. No IP address will be blocked at this time. When Douban is not very frequently requested, even without a cookie, Douban will return a bid each time.

In order to solve the problem of IP blocking on the target website, cookie middleware is designed in the middleware of the system. Cookies store session ID and other information. Subsequent requests will carry the cookie information that has been generated and send it to the server.

Therefore, according to Douban's anti-reptile strategy, our crawler strategy here is as follows. Request once without a cookie and save the returned cookie. All subsequent requests will be with this cookie. After the request is blocked for a period of time, all cookie information is cleared. The request is made again, and the available cookies are returned in a loop.

3.2 De-emphasis module and data storage

When a Web crawler crawls a webpage, there will be multiple URLs pointing to the same resource. If not processed, the crawler's link library data will expand at an extremely fast rate, and the same resource will be crawled multiple times, thus wasting resources. Normalization is the process of converting these different URLs into the same URL. URL normalization can reduce the repeated crawling and storage of pages. It can improve the efficiency of Web crawlers and reduce the workload of page de-emphasis during subsequent indexing. Before development, the format of Douban’s movie category webpage is:

https://movie.douban.com/explore#!type=movie&tag=%E7%83%AD%E9%97%A8&sort=recommend&page_limit=20&page_start= 0

start: from 0 to 9979, refers to the serial number of the first piece of data. 20 pieces of data will be returned each time and there are 10,000 pieces of movie information in total. The return format we request is as follows.

title: "Hunger Platform El hoyo"

rating: "7"

star: 4

url: "https://movie.douban.com/subject/34805219/"

recommend: "https://movie.douban.com/subject/34805219/comments?status=P"

The URL of the response data can be filtered by bloom-filter and stored in our new task queue. After 100,500 requests under ideal conditions, there will be 10,000 movie information in our data. In fact, due to relatively strict Douban anti-reptile strategies, there may be less than 10,000 pieces of data. Among them, the use of redis (task entry and exit management and bloom-filter filter) is more important in the crawling process.

Large crawler systems have thousands of links to crawl, and it is necessary to ensure that crawler links cannot be cycled. This requires the de-emphasis of the link list. Therefore, the URL task of this system is divided into two priorities, a and b, a>b. The launched URL (the movie list of Douban) is stored in arank's redis set, and the URL of the movie details crawled down is saved in brank's redis set after bloom-filter de-emphasis. Then determine whether it exists before crawling.

Bloom-filter is an efficient search algorithm based on a hash function, and it is used to find whether the data is in the bloom set. Compared with direct hash table, bloom-filter has a great advantage in space and can be quickly inserted and queried. Bloom-filter is generally suitable for de-emphasis scenarios where the accuracy requirement is not 100% for a large amount of data.

4. System result analysis

After the key modules of the scrapy framework are completed, unit tests need to be performed in

each module. In this crawler system, it is mainly to test the middleware anti-reptile module and the crawler analysis module of the system to verify whether each part can run normally and whether it can crawl the expected data. After debugging, the distributed crawler system crawled a total of more than 30,000 movies on Douban, 28,000 pieces of users' data, and 2.8 million pieces of ratings and content data. Data cleaning improves the consistency and effectiveness of the data. Data statistics describe the data intuitively, and the data can also be used in the recommendation system experiment in the future.

Conclusion

This paper introduces the design idea and implementation process of the distributed crawler crawling movie review information system based on the Scrapy framework. The core of the implementation process is the scrapy framework, URL de-emphasis module and data storage module. The system operation results show that the system runs relatively smoothly this time. The data crawling is fast and the accuracy is very high. At present, with the development of the Internet, it is easier to achieve greater benefits in user retention and personalized services for accurate content in the vertical field. This system can be used to efficiently crawl vertical content. At the same time, massive data after being processed can also be used to support a personalized movie recommendation system, which has certain commercial value.

Fund support

Jiangsu University Senior Talent Research Startup Fund (1291140025)

References

[1] Yang Z. Analysis on visualization of recruiting information based on Python language [J]. Computer & Network, 2020, 46 (2): 61-64.

[2] Chen Q. Research on Python-based web crawler application [J]. Telecom World, 2020, 27 (1): 202-203.

[3] Zhai P. Comparative analysis of crawling strategies in Python network [J]. Computer Knowledge and Technology, 2020, 16 (1): 29-30 + 34.

[4] Liu C Q. Design and implementation of learning resource acquisition system based on web crawler [J]. Journal of Liaoning Teachers College (Natural Science Edition), 2019, 21 (4): 32-37.

[5] Li Y X, Wang M Y, Tu Y X. Research on Python-based web crawler technology [J]. Information Technology and Informatization, 2019 (12): 143-145.

[6] Zhou W P. Product label extraction based on user reviews under distributed crawler [D]. Nanjing: Nanjing University of Posts and Telecommunications, 2019.

[7] Luo X. User comment analysis system based on distributed crawlers [D]. Nanjing: Nanjing University of Posts and Telecommunications, 2018.

[8] Hou J R, Lv J X. Tmall commodity crawler technology based on Python [J]. Science & Technology Information, 2019, 17 (32): 10 + 12.

[9] Ma L, Feng X W, Dou Y Z, et al. Research and implementation of distributed reptiles [J]. Computer Technology and Development, 2020, 30 (2): 192-196.

[10] Lv T Z, Zhang J. Design and implementation of research ability evaluation based on Hadoop [J]. Century Science Publishing Co, 2019, 6 (10).

[11] Li G M, Li P, Wang C. Distributed crawling and data analysis based on scrapy —taking ZhiHu topic for example [J]. Journal of Hubei Normal University (Natural Science), 2019, 39 (3): 1-7.