Application Research of Crawler and Data Analysis Based on Python

Wu Hejing
East University of Heilongjiang
Heilongjiang, 150086
E-mail: 499917928@qq.com,

Liu Fang
East University of Heilongjiang
Heilongjiang, 150086

Zhao Long
East University of Heilongjiang
Heilongjiang, 150086

Shao Yabin
East University of Heilongjiang
Heilongjiang, 150086

Cui Ran
East University of Heilongjiang
Heilongjiang, 150086

Abstract—Combined with the actual situation, this paper explores how to develop a crawler method based on the specific framework for the complete interface of steam manufacturers and stores, which should be able to automatically and efficiently crawl the data of specific targets, analyze the dynamic pages, and complete the data cleaning, downloading, saving and other operations, explore the methods of general data analysis, and analyze the downloaded data, extract useful information from it, analyze and summarize the specific crawler method and data analysis method through practical application.

Keywords-Python; Scrapy; Selenium; BeautifulSoup

I. INTRODUCTION

The 21st century is a book written by information. With the rapid development of information technology, today's society has become a huge information polymer, and there are various kinds of data in this huge polymer. Data is a kind of embodiment of information. In this era of information explosion, how to efficiently find the data we want from all kinds of miscellaneous data and extract them from the network in batches has become a key problem. However, sometimes the unprocessed data itself may be confusing for people. How to process the huge and complex data obtained through what kind of technical means, and finally become an intuitive number, or trend, and become the information that people can obtain intuitively is also a very important topic to be studied in this data age.

II. STATISTICAL INVESTIGATION ON THE PREFERENCE SALES VOLUME

In this project, the American Steam online game platform mall is selected as the research object of the crawler. By setting a specific game company as a search keyword in steam's online mall, the data of all works of the company in steam mall are crawled, and the useful information is extracted by analyzing the basic data of each manufacturer's preference for game production type, series sales volume, and praise. In addition, the game manufacturers are comprehensively scored and evaluated.
III. RELEVANT TECHNOLOGY AND FRAMEWORK

This project will use the scrapy framework based on Python language to crawl steam website. Python as a language has the advantages of lightweight, simplicity, wide range of application and so on. At present, various crawler frameworks and application libraries based on Python have been very mature, among which the crawler framework is very popular in the application of general web crawlers. Its first version was released in 2008, and now it is quite mature as a crawler framework. The basic principle of the scrapy framework is shown in Figure 1.

IV. DESIGN OF CRAWLER

A. General design idea

The process of crawler itself is actually to simulate the user's operation on the browser with a program. First of all, the starting point and range of crawling need to be specified. As the target of crawling is for manufacturers and their works, the interface of manufacturers is taken as the starting point. For example, the page of paradox, a manufacturer, first analyzes the entire manufacturer's page, and finds that the page links and information of all games or game related DLC downloads of the manufacturer are stored in the recommendation div framework of each sub recommendation of recommendations rows, as shown in Figure 2.

B. Design and implementation of reptile functions

The crawler architecture is composed of items, spiders, piplings and middleware. Among them, items are mainly used to define the items to be crawled, spiders are responsible for defining the whole process of crawling, what means to crawl, pipes are responsible for the basic operations such as data cleaning and saving, middleware can be responsible for the bridge service of scratch and other plug-ins or architectures.
import scrapy

class SteamDevItem(scrapy.Item):
    # define the fields for your item here like:
    # name = scrapy.Field()
    qry_nam = scrapy.Field()
    if_dev = scrapy.Field()
    pub_sum = scrapy.Field()
    pub_gam_sum = scrapy.Field()
    pub_dlc_sum = scrapy.Field()
    dev_nam = scrapy.Field()
    pub_nam = scrapy.Field()
    gam_title = scrapy.Field()
    res_date = scrapy.Field()
    gam_type = scrapy.Field()
    gam_tag = scrapy.Field()
    if_muti = scrapy.Field()
    gam_score = scrapy.Field()
    gam_score_sum = scrapy.Field()
    gam_score_ratio = scrapy.Field()

C. Spider design

The design of spider is the key point of this project. Whether the initial dynamic page connection or the last static page information crawling mode will be defined

First, the items to be crawled are defined in the items file. Finally, these items may be submitted to the analysis part for data analysis. The specific design and implementation code is:

Figure 2. Investigation of HTML page structure of steam manufacturers by using viewers
in this file. In this project, spider will be named steam, and some key implementation codes will be pasted here, with running results and some notes attached. First, introduce start_ the design method of dynamic page crawling of selenium in requests method:

```python
chrome_opt = webdriver.ChromeOptions()
prefs = {
    "profile.managed_default_content_settings.images": 2,
    'permissions.default.stylesheet': 2
}
chrome_opt.add_experimental_option("prefs", prefs)

browser = webdriver.Chrome(options=chrome_opt)

browser.get("https://store.steampowered.com/" + Qry sta + "+" + Qry_Target)
bs = BeautifulSoup(browser.page_source, 'html.parser')
```

The specific store connections of each product exist in the a anchor label of each entry, and these connections are read to the defined links using the loop. In the list list, crawling of the list is completed, but sometimes the text and picture in the entry may contain a tag, and they all point to the same page. If direct application may cause repeated crawling, a loop is used here, and if not in statement is used to de duplicate the list.

After using the print statement to verify the function of the module, the verification results are shown in Figure 3.

Figure 3. List of URLs obtained by selenium and beautiful soup
D. Start directional climbing

After designing and debugging the spider, run the CMD command window of the system, open the root directory of the crawler file, and input the crawler stream-o SteamDev.csv, crawl the target website.

Input - O SteamDev.csv The purpose is to let the crawler save the last crawled data in the form of CSV table. The saved data appears in the project root. See Figure 4 for the climbing process.

![Figure 4. Executing the start request method selenium pop-up browser to crawl the dynamic page](image)

V. DATA ANALYSIS

Next, we will perform basic visual operations on the crawled data in the form of operation tables. In the crawler project, we crawled for the Paradox Interactive publisher. The crawled data is presented in the form of CSV tables, as shown in Figure 5.

Through the use of spreadsheets and further collation of the crawled data, the following data are obtained: the publisher has published 396 works in steam platform, of which the majority of DLC has published 334 DLC, most of the games published are single player games, and each game published in its mall has an average of 6800 reviews, of which the proportion of favorable reviews is about 76.48%, see the chart below for detailed visual analysis.
Figure 5. Crawled data list

<table>
<thead>
<tr>
<th>game</th>
<th>year</th>
<th>strategy</th>
<th>publisher</th>
</tr>
</thead>
<tbody>
<tr>
<td>Triumph I</td>
<td>1945</td>
<td>7. Very PostStrategies, Age of W</td>
<td>Single-player</td>
</tr>
<tr>
<td>Paradox I</td>
<td>1187</td>
<td>Mostly PostStrategy, Empires of simulation, Strategy</td>
<td>Single-player</td>
</tr>
<tr>
<td>Wargame</td>
<td>1094</td>
<td>7. Very PostReacts, Struggle to advance, Strategy</td>
<td>Single-player</td>
</tr>
<tr>
<td>Paradox I</td>
<td>7040</td>
<td>7. Very PostColony, Struggle to survive, Strategy</td>
<td>Single-player</td>
</tr>
<tr>
<td>Gwy Games</td>
<td>116</td>
<td>Mostly RPG, IndyKnights (Indy, RPG, Simulation)</td>
<td>Single-player</td>
</tr>
<tr>
<td>Sega Gzy</td>
<td>556</td>
<td>7. Very PostWorlds, Realistic action, Simulation</td>
<td>Single-player</td>
</tr>
<tr>
<td>Xoludian</td>
<td>5949</td>
<td>7. Very PostWorlds, Story of Fantasy</td>
<td>Single-player</td>
</tr>
<tr>
<td>Paradox I</td>
<td>3384</td>
<td>Mostly PostStrategy, ROOKIOPOLY</td>
<td>Single-player</td>
</tr>
<tr>
<td>Xoludian</td>
<td>9525</td>
<td>7. Very PostStruggle, Ghosts of a New World</td>
<td>Single-player</td>
</tr>
<tr>
<td>Collinst</td>
<td>74257</td>
<td>7. Very PostCity, Simulated Cities</td>
<td>Single-player</td>
</tr>
<tr>
<td>Farbriz</td>
<td>3528</td>
<td>7. Very PostRPG, Cyber Shadows, Strategy</td>
<td>Single-player</td>
</tr>
<tr>
<td>Inc-Com P</td>
<td>697</td>
<td>7. Very PostStruggle, Parallel Cities</td>
<td>Single-player</td>
</tr>
<tr>
<td>Triumph I</td>
<td>5144</td>
<td>7. Very PostSurvival, Age of W</td>
<td>Single-player</td>
</tr>
<tr>
<td>Farbriz</td>
<td>7411</td>
<td>7. Very PostRPG, Cyber Shadows, Strategy</td>
<td>Single-player</td>
</tr>
<tr>
<td>Post PC</td>
<td>819</td>
<td>7. Very PostSurvival, City of Shadows, Strategy</td>
<td>Single-player</td>
</tr>
<tr>
<td>Schold J</td>
<td>1541</td>
<td>7. Very PostRPG, Turn Knights</td>
<td>Single-player</td>
</tr>
<tr>
<td>Collinst</td>
<td>1016</td>
<td>7. Very PostSimulation, Simulated Cities</td>
<td>Single-player</td>
</tr>
<tr>
<td>Zolock</td>
<td>375</td>
<td>7. Very PostSurvival, Struggle to survive</td>
<td>Single-player</td>
</tr>
<tr>
<td>Veevee</td>
<td>333</td>
<td>7. Very PostSurvival, City of Shadows</td>
<td>Single-player</td>
</tr>
<tr>
<td>Yealc</td>
<td>27</td>
<td>7. Very PostSurvival, City of Shadows</td>
<td>Single-player</td>
</tr>
</tbody>
</table>

Figure 6. Output the publisher platform follower ranking chart

Publisher
VI. CONCLUSION

Through demonstration and part of practice, this paper explores the process of data crawling and basic data analysis of dynamic pages by combining the general Python's story framework with selenium + beautiful soup through crawling the steam online game mall website.

The crawler has good scalability. For example, if you want to compare the crawling data of multiple game manufacturers, you can write a query manufacturer list to get the product URL list from the dynamic web page of the manufacturer list first. In terms of anti-crawler, selenium itself has a very good anti crawler ability. If you want to further anti crawler, you can also expand multiple cookies, and even establish a proxy IP pool.

ACKNOWLEDGMENT

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

REFERENCE