Steampricehistory 数据爬虫

1. 基本思路

1.1 一级页面

首先观察目标网页 `https://steampricehistory.com/popular` 结构为简单的二级结构，每个一级页面由50个二级页面对象组成，二级页面中要的表格即为我们需要爬取的数据。翻页后观察到一级页面的翻页操作可通过更改 `https://steampricehistory.com/popular?page=X` 后的X值实现，因此首先我们可以利用一个简单的循环遍历每个一级页面。

1.2 二级页面

然后我们观察二级页面，发现url中有一串数字id，以游戏 `Portal(传送门)` 为例，对应的二级界面url为 `https://steampricehistory.com/app/400`，其中**400**即为该游戏在steam内部对应的游戏id，而经测试id不是连续的整数，因此从一级界面进入二级界面获取数据仍是较为合理的方式。

1.3 整体实现

使用 `request-html` 库发送请求获取网页对象，之所以不用 `request` 库是因为 `reque-html` 功能更为强大，能实现等待页面渲染防止数据未加载等功能。然后使用 `pandas` 库读取网页中的表格并存入文件中。另外还使用了 `BeautifulSoup` 以及一些其他的库实现读取游戏名称和id等操作。

2. Tips

使用 `request-html` 库发送请求时使用了伪装请求头和代理ip池来对抗网页的基本反爬措施。其中伪装请求头由 `fake-useragent` 库实现，代理ip可以从快代理 `www.kuaidaili.com` 中找到免费的，经过测试有效性还可以（不是广告）。另外，windows不支持文件名中有 `:` 冒号字符，所以游戏名中的冒号全部被用下滑线 `_` 取代了，游戏名中剩下的乱码应该是gbk编码问题，后期数据清洗的时候再加以解决。

3. 完整代码

1. 爬虫本体

```
#尝试爬取steamhistoryprice的popular目录界面
import socket
import urllib.error
import html5lib
from requests_html import HTMLSession
import json
from urllib import request,parse
import time
import random
from fake_useragent import  UserAgent
from lxml import html
import requests
import re
from bs4 import BeautifulSoup
```

```
import pandas as pd


#定义一个爬虫类
class SteamSpider(object):
    #初始化url属性
    def __init__(self):
        self.url='https://steampricehistory.com/popular?{}'
        # 添加代理ip
        self.proxies = {
            'http': 'http://61.216.156.222:60808',
            'http': 'http://112.14.47.6:52024',
            'http': 'http://182.139.110.52:9000',
            'http': 'http://117.93.180.175:9000',
            'http': 'http://222.74.73.202:42055',
            'http': 'http://27.42.168.46:55481',
            'http': 'http://27.42.168.46:9000',
            'http': 'http://182.139.110.207:9000',
            'http': 'http://116.9.163.205:58080',
            'http': 'http://61.164.39.68:53281',
            'http': 'http://117.114.149.66:55443',
            'http': 'http://49.85.15.144:9000'
        }
    # 1.请求函数，得到一级页面，传统三步
    def get_html(self,url):
        #req=request.Request()
        session = HTMLSession()
        res=session.get(url=url, headers={'User-Agent': UserAgent().random}, proxies
= self.proxies)
        #res=requests.get(url=url, headers={'User-Agent': UserAgent().random})
        #windows会存在乱码问题，需要使用 gbk解码，并使用ignore忽略不能处理的字节
        #linux不会存在上述问题，可以直接使用decode('utf-8')解码
        res.html.render(retries=3, wait=3, sleep=3, timeout=20)
        html=res.content.decode('gbk', 'ignore')
        return html
    # 2.解析函数
    def parse_html(self):
        pass
    # 3.保存文件函数
    def save_html(self,filename,html):
        with open(filename,'w') as f:
            f.write(html)
    # 4.获取二级页面url
    def get_detailurl(self, url):
        #定义前缀
        prefix = "https://steampricehistory.com/app/"
        # 获取网页内容
        session = HTMLSession()
        response = session.get(url=url, headers={'User-Agent': UserAgent().random},
proxies = self.proxies)
        response.html.render(retries=3, wait=2, sleep=2, timeout=20)
        content = response.content
```

```
        # 解析HTML
        tree = html.fromstring(content)
        # 提取所有链接
        links = tree.xpath('//a/@href')
        # 过滤出符合前缀的链接
        pattern = re.compile(prefix)
        filtered_links = [link for link in links if pattern.match(link)]
        # 去除URL中的查询参数
        clean_links = [link.split("?")[0] for link in filtered_links]
        # 去除重复元素并存入集合
        unique_links = set(clean_links)
        return unique_links
    # 5.提取二级页面表格
    def extract_xlsx(self, url, filename):
        # 异常处理，不知道为什么有时候会读取不到表格所以用死循环实现出现异常时重试
        while True:
            try:
                html = self.get_html(url)
                tables = pd.read_html(html, flavor='bs4')
                break
            except:
                print('Table error')
                time.sleep(2)
        table = tables[1]
        table["Date"] = pd.to_datetime(table["Date"], format="%B %d, %Y")

        # 将Date列格式化为yyyy-dd-mm的格式
        table["Date"] = table["Date"].dt.strftime("%Y-%m-%d")
        # 将冒号替换为下划线
        filename = filename.replace(":", "_")
        # 使用to_excel方法将表格写入XLS文件
        table.to_excel(filename + ".xlsx", index=False)
    # 6.入口函数
    def run(self):
        begin=int(input('输入起始页：'))
        stop=int(input('输入终止页：'))
        # +1 操作保证能够取到整数
        for page in range(begin,stop+1):
            pn=page
            params={
                'page':str(pn)
            }
            #拼接URL地址
            params=parse.urlencode(params)
            url=self.url.format(params)
            #发请求
            fst_html = self.get_html(url)
            scnd_urls = self.get_detailurl(url).copy()
            for suburl in scnd_urls:
                # 定义模式
                pattern = "(\d+)"
                # 使用search方法匹配数字ID
```

```
                match = re.search(pattern, suburl)
                # 使用group方法获取数字ID
                id = match.group(1)
                # 获取游戏名称
                # 创建一个BeautifulSoup对象
                soup = BeautifulSoup(fst_html, "html.parser")
                # 找到a标签
                a_tags = soup.find_all("a", href=lambda x: x and
x.startswith("https://steampricehistory.com/app/" + id))
                if(len(a_tags) == 0):
                    print('error')
                    self.save_html('errorpage', fst_html)
                a_tag = a_tags[1]
                # 获取a标签中的文本
                text = a_tag.text
                #定义路径
                filename='{}-{}'.format(id, text)
                print(filename)
                #self.save_html(filename,html)
                self.extract_xlsx(suburl, filename)
                #每爬取一个页面随机休眠1-2秒钟的时间
                #time.sleep(random.randint(1, 2))
            print('爬取成功第', pn, '页')
#以脚本的形式启动爬虫
if __name__=='__main__':
    start=time.time()
    spider=SteamSpider() #实例化一个对象spider
    spider.run() #调用入口函数
    end=time.time()
    #查看程序执行时间
    print('执行时间:%.2f'%(end-start))  #爬虫执行时间
```

2. 测试ip是否有效的小程序

```
import requests

url = 'https://steampricehistory.com/popular?page=2'
proxies_list = [
            'http://61.216.156.222:60808',
            'http://112.14.47.6:52024',
            'http://182.139.110.52:9000',
            'http://117.93.180.175:9000',
            'http://222.74.73.202:42055',
            'http://27.42.168.46:55481',
            'http://27.42.168.46:9000',
            'http://182.139.110.207:9000',
            'http://116.9.163.205:58080',
            'http://61.164.39.68:53281',
            'http://117.114.149.66:55443',
            'http://49.85.15.144:9000'
]
```

```
ip_list = []

for proxy_ip in proxies_list:
    print(proxy_ip)
    # print(proxies_list)
    proxies = {'http': proxy_ip}
    try:
        wb_data = requests.get(url=url, proxies=proxies)
        flag = True
    except:
        proxies_list.remove(proxies['http'])
        flag = False

    if flag:
        ip_list.append(proxies['http'])
print(ip_list)
```