Day03回顾

目前反爬总结

- **反爬虫梳理**

```
【1】Headers反爬虫
    1.1) 检查: Cookie、Referer、User-Agent
    1.2) 解决方案: 通过F12获取headers,传给requests.get()方法

【2】IP限制
    2.1) 网站根据IP地址访问频率进行反爬,短时间内限制IP访问
    2.2) 解决方案:
            a) 构造自己IP代理池,每次访问随机选择代理,经常更新代理池
            b) 购买开放代理或私密代理IP
            c) 降低爬取的速度

【3】User-Agent限制
    3.1) 类似于IP限制，检测频率
```

3.2) 解决方案：构造自己的User-Agent池，每次访问随机选择

a> fake_useragent模块

b> 新建py文件，存放大量User-Agent

【4】对响应内容做处理

4.1) 页面结构和响应内容不同

4.2) 解决方案：打印并查看响应内容，用xpath或正则做处理

requests模块参数总结

```
【1】方法一 ： requests.get()
【2】参数
  2.1) url
  2.2) headers
  2.3) timeout
  2.4) proxies

【3】方法二 ： requests.post()
【4】参数
    data ： {}
```

解析模块总结

- **re正则解析**

```
import re
pattern = re.compile('正则表达式',re.S)
r_list = pattern.findall(html)
```

- **lxml+xpath解析**

```
from lxml import etree
p = etree.HTML(res.text)
r_list = p.xpath('xpath表达式')

【谨记】只要调用了xpath，得到的结果一定为'列表'
```

xpath表达式

- **匹配规则**

```
【1】结果: 节点对象列表
   1.1) xpath示例:
//div、//div[@class="student"]、//div/a[@title="stu"]/span

【2】结果: 字符串列表
   2.1) xpath表达式中末尾为: @src、@href、/text()
```

- **最常用**

```
【1】基准xpath表达式: 得到节点对象列表
【2】for r in [节点对象列表]:
        username = r.xpath('./xxxxxx')

【注意】遍历后继续xpath一定要以:  . 开头，代表当前节点
```

Day04笔记

requests.post()

- **适用场景**

```
【1】适用场景 : Post类型请求的网站

【2】参数 : data={}
   2.1) Form表单数据: 字典
   2.2) res = requests.post(url=url,data=data,headers=headers)

【3】POST请求特点 : Form表单提交数据
```

控制台抓包

- **打开方式及常用选项**

【1】打开浏览器，F12打开控制台，找到Network选项卡

【2】控制台常用选项

2.1) Network: 抓取网络数据包

a> ALL: 抓取所有的网络数据包

b> XHR: 抓取异步加载的网络数据包

c> JS : 抓取所有的JS文件

2.2) Sources: 格式化输出并打断点调试JavaScript代码，助于分析爬虫中一些参数

2.3) Console: 交互模式，可对JavaScript中的代码进行测试

【3】抓取具体网络数据包后

3.1) 单击左侧网络数据包地址，进入数据包详情，查看右侧

3.2) 右侧:

a> Headers: 整个请求信息

General、Response Headers、Request Headers、Query String、Form Data

b> Preview: 对响应内容进行预览

c> Response: 响应内容

有道翻译破解案例(post)

- **目标**

破解有道翻译接口，抓取翻译结果

```
# 结果展示
请输入要翻译的词语： elephant
翻译结果： 大象
**************************
请输入要翻译的词语： 喵喵叫
翻译结果： mews
```

- **实现步骤**

【1】浏览器F12开启网络抓包,Network-All,页面翻译单词后找Form表单数据

【2】在页面中多翻译几个单词，观察Form表单数据变化（有数据是加密字符串）

【3】刷新有道翻译页面，抓取并分析JS代码（本地JS加密）

【4】找到JS加密算法，用Python按同样方式加密生成加密数据

【5】将Form表单数据处理为字典，通过requests.post()的data参数发送

- **具体实现**

1、开启F12抓包，找到Form表单数据如下:

```
i: 喵喵叫
from: AUTO
to: AUTO
smartresult: dict
client: fanyideskweb
salt: 15614112641250
sign: 94008208919faa19bd531acde36aac5d
ts: 1561411264125
bv: f4d62a2579ebb44874d7ef93ba47e822
doctype: json
version: 2.1
keyfrom: fanyi.web
action: FY_BY_REALTlME
```

2、在页面中多翻译几个单词，观察Form表单数据变化

```
salt: 15614112641250
sign: 94008208919faa19bd531acde36aac5d
ts: 1561411264125
bv: f4d62a2579ebb44874d7ef93ba47e822
# 但是bv的值不变
```

3、一般为本地js文件加密，刷新页面，找到js文件并分析JS代码

```
【方法1】 : Network - JS选项 - 搜索关键词salt
【方法2】 : 控制台右上角 - Search - 搜索salt
- 查看文件 - 格式化输出

【结果】 : 最终找到相关JS文件 : fanyi.min.js
```

4、打开JS文件，分析加密算法，用Python实现

```
【ts】经过分析为13位的时间戳，字符串类型
   js代码实现)  "" + (new Date).getTime()
   python实现) str(int(time.time()*1000))

【salt】
   js代码实现)  ts + parseInt(10 *
Math.random(), 10);
   python实现)  ts + str(random.randint(0,
9))

【sign】（'设置断点调试，来查看 e 的值，发现 e
为要翻译的单词'）
   js代码实现) n.md5("fanyideskweb" + e +
salt + "]BjuETDhU)zqSxf-=B#7m")
   python实现)
    from hashlib import md5
    s = md5()
    s.update(string.encode())
    sign = s.hexdigest()
```

5、pycharm中正则处理headers和formdata

```
【1】pycharm进入方法 ： Ctrl + r ，选中 Regex
【2】处理headers和formdata
    (.*): (.*)
    "$1": "$2",
【3】点击 Replace All
```

5、代码实现

```
import requests
import time
import random
from hashlib import md5

class YdSpider(object):
  def __init__(self):
    # url一定为F12抓到的 headers -> General -> Request URL
    self.url = 'http://fanyi.youdao.com/translate_o?smartresult=dict&smartresult=rule'
    self.headers = {
      # 检查频率最高 - 3个
```

```
        "Cookie":
"OUTFOX_SEARCH_USER_ID=970246104@10.169.0
.83;
OUTFOX_SEARCH_USER_ID_NCOO=570559528.1224
236;
_ntes_nnid=96bc13a2f5ce64962adfd6a2784672
14,1551873108952;
JSESSIONID=aaae9i7plXPlKaJH_gkYw;
td_cookie=18446744072941336803;
SESSION_FROM_COOKIE=unknown;
___rl__test__cookies=1565689460872",
        "Referer":
"http://fanyi.youdao.com/",
        "User-Agent": "Mozilla/5.0 (Windows
NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/76.0.3809.100
Safari/537.36",
    }

  # 获取salt,sign,ts
  def get_salt_sign_ts(self,word):
    # ts
    ts = str(int(time.time()*1000))
    # salt
    salt = ts + str(random.randint(0,9))
    # sign
    string = "fanyideskweb" + word + salt
+ "n%A-rKaT5fb[Gy?;N5@Tj"
    s = md5()
```

```
        s.update(string.encode())
        sign = s.hexdigest()

        return salt,sign,ts

    # 主函数
    def attack_yd(self,word):
        # 1. 先拿到salt,sign,ts
        salt,sign,ts =
self.get_salt_sign_ts(word)
        # 2. 定义form表单数据为字典: data={}
        # 检查了salt sign
        data = {
            "i": word,
            "from": "AUTO",
            "to": "AUTO",
            "smartresult": "dict",
            "client": "fanyideskweb",
            "salt": salt,
            "sign": sign,
            "ts": ts,
            "bv":
"7e3150ecbdf9de52dc355751b074cf60",
            "doctype": "json",
            "version": "2.1",
            "keyfrom": "fanyi.web",
            "action": "FY_BY_REALTlME",
        }
```

```
        # 3. 直接发请求:requests.post(url,data=data,headers=xxx)
        html = requests.post(
            url=self.url,
            data=data,
            headers=self.headers
        ).json()
        # res.json() 将json格式的字符串转为python数据类型
        result = html['translateResult'][0][0]['tgt']

        print(result)

    # 主函数
    def run(self):
        # 输入翻译单词
        word = input('请输入要翻译的单词:')
        self.attack_yd(word)

if __name__ == '__main__':
    spider = YdSpider()
    spider.run()
```

动态加载数据抓取-Ajax

- **特点**

【1】右键 -> 查看网页源码中没有具体数据
【2】滚动鼠标滑轮或其他动作时加载,或者页面局部刷新

- **抓取**

【1】F12打开控制台，页面动作抓取网络数据包
【2】抓取json文件URL地址
  2.1) 控制台中 XHR ：异步加载的数据包
  2.2) XHR -> QueryStringParameters(查询参数)

豆瓣电影数据抓取案例

- **目标**

【1】地址： 豆瓣电影 - 排行榜 - 剧情
【2】目标： 电影名称、电影评分

```
<span><a href=".*?type_name=(.*?)&type=(.*?)&interval_id=100:90&action=">
```

- **F12抓包（XHR）**

```
【1】Request URL(基准URL地址) :
https://movie.douban.com/j/chart/top_list?
【2】Query String(查询参数)
    # 抓取的查询参数如下:
    type: 13 # 电影类型
    interval_id: 100:90
    action: ''
    start: 0  # 每次加载电影的起始索引值 0 20 40 60
    limit: 20 # 每次加载的电影数量
```

- **代码实现 - 全站抓取**

```
"""
豆瓣电影 - 全站抓取
"""
import requests
from fake_useragent import UserAgent
import time
import random
import re
import json

class DoubanSpider:
    def __init__(self):
        self.url = 'https://movie.douban.com/j/chart/top_list?'
```

```
        self.i = 0
        # 存入json文件
        self.f = open('douban.json',
'w', encoding='utf-8')
        self.all_film_list = []

    def get_agent(self):
        """获取随机的User-Agent"""
        return UserAgent().random

    def get_html(self, params):
        headers = {'User-
Agent':self.get_agent()}
        html =
requests.get(url=self.url,
params=params, headers=headers).text
        # 把json格式的字符串转为python数据
类型
        html = json.loads(html)

        self.parse_html(html)

    def parse_html(self, html):
        """解析"""
        # html: [{},{},{},{}]
        item = {}
        for one_film in html:
            item['rank'] =
one_film['rank']
```

```
            item['title'] =
one_film['title']
            item['score'] =
one_film['score']
            print(item)

 self.all_film_list.append(item)
            self.i += 1

    def run(self):
        # d: {'剧情':'11','爱情':'13','喜
剧':'5',...,...}
        d = self.get_d()
        # 1、给用户提示,让用户选择
        menu = ''
        for key in d:
            menu += key + '|'
        print(menu)
        choice = input('请输入电影类别: ')
        if choice in d:
            code = d[choice]
            # 2、total: 电影总数
            total =
self.get_total(code)
            for start in
range(0,total,20):
                params = {
                    'type': code,
```

```
                    'interval_id':
'100:90',
                    'action': '',
                    'start':
str(start),
                    'limit': '20'
                }

 self.get_html(params=params)

 time.sleep(random.randint(1,2))

            # 把数据存入json文件

 json.dump(self.all_film_list, self.f,
ensure_ascii=False)
            self.f.close()
            print('数量:',self.i)
        else:
            print('请做出正确的选择')

    def get_d(self):
        """{'剧情':'11','爱情':'13','喜
剧':'5',...,...}"""
        url =
'https://movie.douban.com/chart'
        html =
requests.get(url=url,headers={'User-
Agent':self.get_agent()}).text
```

```
        regex = '<span><a href=".*?
type_name=(.*?)&type=
(.*?)&interval_id=100:90&action=">'
        pattern = re.compile(regex,
re.S)
        # r_list: [('剧情','11'),('喜
剧','5'),('爱情':'13')... ...]
        r_list = pattern.findall(html)
        # d: {'剧情': '11', '爱情':
'13', '喜剧': '5', ..., ...}
        d = {}
        for r in r_list:
            d[r[0]] = r[1]

        return d

    def get_total(self, code):
        """获取某个类别下的电影总数"""
        url =
'https://movie.douban.com/j/chart/top_l
ist_count?type=
{}&interval_id=100%3A90'.format(code)
        html =
requests.get(url=url,headers={'User-
Agent':self.get_agent()}).text
        html = json.loads(html)

        return html['total']
```

```
if __name__ == '__main__':
    spider = DoubanSpider()
    spider.run()
```

json解析模块

- **json.loads(json)**

```
【1】作用 ： 把json格式的字符串转为Python数据类型

【2】示例 ： html = json.loads(res.text)
```

- **json.dump(python,f,ensure_ascii=False)**

```
【1】作用
    把python数据类型 转为 json格式的字符串,一般让你把抓取的数据保存为json文件时使用

【2】参数说明
    2.1) 第1个参数: python类型的数据(字典，列表等)
    2.2) 第2个参数: 文件对象
    2.3) 第3个参数: ensure_ascii=False 序列化时编码

【3】示例代码
    # 示例1
    import json
```

```
    item = {'name':'QQ','app_id':1}
    with open('小米.json','a') as f:

json.dump(item,f,ensure_ascii=False)

    # 示例2
    import json

    item_list = []
    for i in range(3):
      item = {'name':'QQ','id':i}
      item_list.append(item)

    with open('xiaomi.json','a') as f:

json.dump(item_list,f,ensure_ascii=Fal
se)
```

- **json模块总结**

```
# 爬虫最常用
【1】数据抓取 - json.loads(html)
    将响应内容由: json 转为 python
【2】数据保存 - 
json.dump(item_list,f,ensure_ascii=Fals
e)
    将抓取的数据保存到本地 json文件

# 抓取数据一般处理方式
【1】txt文件
【2】csv文件
【3】json文件
【4】MySQL数据库
【5】MongoDB数据库
【6】Redis数据库
```

多线程爬虫

- **应用场景**

```
【1】多进程 : CPU密集程序
【2】多线程 : 爬虫(网络I/O)、本地磁盘I/O
```

知识点回顾

- **队列**

```
【1】导入模块
   from queue import Queue
```

```
【2】使用
    q = Queue()
    q.put(url)
    q.get()    # 当队列为空时，阻塞
    q.empty() # 判断队列是否为空，
True/False

【3】q.get()解除阻塞方式
   3.1) q.get(block=False)
   3.2) q.get(block=True,timeout=3)
   3.3) if not q.empty():
            q.get()
```

- **线程模块**

```
# 导入模块
from threading import Thread

# 使用流程
t = Thread(target=函数名) # 创建线程对象
t.start() # 创建并启动线程
t.join()  # 阻塞等待回收线程

# 如何创建多线程
t_list = []

for i in range(5):
    t = Thread(target=函数名)
```

```
    t_list.append(t)
    t.start()

for t in t_list:
    t.join()
```

- **线程锁**

```
from threading import Lock

lock = Lock()
lock.acquire()
lock.release()
```

【注意】上锁成功后,再次上锁会阻塞

- **多线程爬虫示例代码**

```
# 抓取豆瓣电影剧情类别下的电影信息
"""
豆瓣电影 - 剧情 - 抓取
"""
import requests
from fake_useragent import UserAgent
import time
import random
from threading import Thread,Lock
from queue import Queue

class DoubanSpider:
```

```
    def __init__(self):
        self.url =
'https://movie.douban.com/j/chart/top_l
ist?
type=13&interval_id=100%3A90&action=&st
art={}&limit=20'
        self.i = 0
        # 队列 + 锁
        self.q = Queue()
        self.lock = Lock()

    def get_agent(self):
        """获取随机的User-Agent"""
        return UserAgent().random

    def url_in(self):
        """把所有要抓取的URL地址入队列"""
        for start in range(0,684,20):
            url =
self.url.format(start)
            # url入队列
            self.q.put(url)

    # 线程事件函数：请求+解析+数据处理
    def get_html(self):
        while True:
            # 从队列中获取URL地址
```

```
            # 一定要在判断队列是否为空 和 get() 地址 前后加锁,防止队列中只剩一个地址时出现重复判断
            self.lock.acquire()
            if not self.q.empty():
                headers = {'User-Agent': self.get_agent()}
                url = self.q.get()
                self.lock.release()

                html = requests.get(url=url, headers=headers).json()
                self.parse_html(html)
            else:
                # 如果队列为空,则最终必须释放锁
                self.lock.release()
                break

    def parse_html(self, html):
        """解析"""
        # html: [{},{},{},{}]
        item = {}
        for one_film in html:
            item['rank'] = one_film['rank']
            item['title'] = one_film['title']
```

```
            item['score'] =
one_film['score']
            print(item)
            # 加锁 + 释放锁
            self.lock.acquire()
            self.i += 1
            self.lock.release()

    def run(self):
        # 先让URL地址入队列
        self.url_in()
        # 创建多个线程,开干吧
        t_list = []
        for i in range(1):
            t =
Thread(target=self.get_html)
            t_list.append(t)
            t.start()

        for t in t_list:
            t.join()

        print('数量:',self.i)

if __name__ == '__main__':
    start_time = time.time()
    spider = DoubanSpider()
    spider.run()
    end_time = time.time()
```

```
    print('执行时间:%.2f' % (end_time-
start_time))
```

selenium+PhantomJS/Chrome/Firefox

- **selenium**

```
【1】定义
    1.1) 开源的web自动化测试工具

【2】用途
    2.1) 对web系统进行功能性测试,版本迭代时避免重复劳动
    2.2) 兼容性测试(测试web程序在不同操作系统和不同浏览器中是否运行正常)
    2.3) 对web系统进行大数量测试

【3】特点
    3.1) 可根据指令操控浏览器
    3.2) 只是工具，必须与第三方浏览器结合使用

【4】安装
    4.1) Linux: sudo pip3 install selenium
    4.2) Windows: python -m pip install selenium
```

- **PhantomJS浏览器**

```
【1】定义
    phantomjs为无界面浏览器(又称无头浏览器)，在内存中进行页面加载,高效

【2】下载地址
    2.1) chromedriver : 下载对应版本

http://npm.taobao.org/mirrors/chromedriver/

    2.2) geckodriver

https://github.com/mozilla/geckodriver/releases

    2.3) phantomjs

https://phantomjs.org/download.html

【3】Ubuntu安装
    3.1) 下载后解压 : tar -zxvf geckodriver.tar.gz

    3.2) 拷贝解压后文件到 /usr/bin/ （添加环境变量）
         sudo cp geckodriver /usr/bin/
```

```
    3.3) 添加可执行权限
        sudo chmod 777
/usr/bin/geckodriver

【4】Windows安装
    4.1) 下载对应版本的phantomjs、
chromedriver、geckodriver
    4.2) 把chromedriver.exe拷贝到python安
装目录的Scripts目录下(添加到系统环境变量)
        # 查看python安装路径: where
python
    4.3) 验证
        cmd命令行: chromedriver

**********总结
**********
【1】解压 - 放到用户主目录(chromedriver、
geckodriver、phantomjs)
【2】拷贝 - sudo cp
/home/tarena/chromedriver /usr/bin/
【3】权限 - sudo chmod 777
/usr/bin/chromedriver

# 验证
【Ubuntu | Windows】
ipython3
from selenium import webdriver
webdriver.Chrome()
或者
```

```
webdriver.Firefox()

【mac】
ipython3
from selenium import webdriver
webdriver.Chrome(executable_path='/Users/xxx/chromedriver')
或者
webdriver.Firefox(executable_path='/User/xxx/geckodriver')
```

- **示例代码**

```
"""示例代码一：使用 selenium+浏览器 打开百度"""

# 导入seleinum的webdriver接口
from selenium import webdriver
import time

# 创建浏览器对象
browser = webdriver.Chrome()
browser.get('http://www.baidu.com/')
# 5秒钟后关闭浏览器
time.sleep(5)
browser.quit()
```

```
"""示例代码二：打开百度，搜索赵丽颖，点击搜索，查看"""

from selenium import webdriver
import time

# 1.创建浏览器对象 - 已经打开了浏览器
browser = webdriver.Chrome()
# 2.输入: http://www.baidu.com/
browser.get('http://www.baidu.com/')
# 3.找到搜索框,向这个节点发送文字: 赵丽颖
browser.find_element_by_xpath('//*[@id="kw"]').send_keys('赵丽颖')
# 4.找到 百度一下 按钮,点击一下
browser.find_element_by_xpath('//*[@id="su"]').click()
```

- **浏览器对象(browser)方法**

```
【1】browser.get(url=url)    - 地址栏输入url地址并确认
【2】browser.quit()          - 关闭浏览器
【3】browser.close()         - 关闭当前页
【4】browser.page_source     - HTML结构源码
【5】browser.page_source.find('字符串')
     从html源码中搜索指定字符串,没有找到返回：-1,经常用于判断是否为最后一页
【6】browser.maximize_window() - 浏览器窗口最大化
```

- **定位节点八种方法**

```
【1】单元素查找('结果为1个节点对象')
    1.1) 【最常用】
browser.find_element_by_id('id属性值')
    1.2) 【最常用】
browser.find_element_by_name('name属性值')
    1.3) 【最常用】
browser.find_element_by_class_name('class属性值')
    1.4) 【最万能】
browser.find_element_by_xpath('xpath表达式')
    1.5) 【匹配a节点时常用】
browser.find_element_by_link_text('链接文本')
```

```
    1.6) 【匹配a节点时常用】
browser.find_element_by_partical_link_text('部分链接文本')
    1.7) 【最没用】
browser.find_element_by_tag_name('标记名称')
    1.8) 【较常用】
browser.find_element_by_css_selector('css表达式')

【2】多元素查找('结果为[节点对象列表]')
    2.1)
browser.find_elements_by_id('id属性值')
    2.2)
browser.find_elements_by_name('name属性值')
    2.3)
browser.find_elements_by_class_name('class属性值')
    2.4)
browser.find_elements_by_xpath('xpath表达式')
    2.5)
browser.find_elements_by_link_text('链接文本')
    2.6)
browser.find_elements_by_partical_link_text('部分链接文本')
```

```
    2.7)
browser.find_elements_by_tag_name('标记名称')
    2.8)
browser.find_elements_by_css_selector('css表达式')
```

- **猫眼电影示例**

```
from selenium import webdriver
import time

url = 'https://maoyan.com/board/4'
browser = webdriver.Chrome()
browser.get(url)

def get_data():
    # 基准xpath: [<selenium xxx li at xxx>,<selenium xxx li at>]
    li_list = browser.find_elements_by_xpath('//*[@id="app"]/div/div/div[1]/dl/dd')
    for li in li_list:
        item = {}
        # info_list: ['1', '霸王别姬', '主演：张国荣', '上映时间：1993-01-01', '9.5']
        info_list = li.text.split('\n')
        item['number'] = info_list[0]
```

```
            item['name'] = info_list[1]
            item['star'] = info_list[2]
            item['time'] = info_list[3]
            item['score'] = info_list[4]

            print(item)

while True:
    get_data()
    try:

 browser.find_element_by_link_text('下一页').click()
            time.sleep(2)
    except Exception as e:
            print('恭喜你!抓取结束')
            browser.quit()
            break
```

- **节点对象操作**

```
【1】文本框操作
    1.1) node.send_keys('')  - 向文本框发送内容
    1.2) node.clear()        - 清空文本
    1.3) node.get_attribute('value') - 获取文本内容

【2】按钮操作
    1.1) node.click()        - 点击
    1.2) node.is_enabled()  - 判断按钮是否可用
    1.3) node.get_attribute('value') - 获取按钮文本
```

chromedriver设置无界面模式

```
from selenium import webdriver

options = webdriver.ChromeOptions()
# 添加无界面参数
options.add_argument('--headless')
browser =
webdriver.Chrome(options=options)
```

==selenium - 鼠标操作==

```
from selenium import webdriver
# 导入鼠标事件类
from selenium.webdriver import
ActionChains

driver = webdriver.Chrome()
driver.get('http://www.baidu.com/')

# 移动到 设置，perform()是真正执行操作，必须有
element =
driver.find_element_by_xpath('//*
[@id="u1"]/a[8]')
ActionChains(driver).move_to_element(elem
ent).perform()

# 单击，弹出的Ajax元素，根据链接节点的文本内容查
找
driver.find_element_by_link_text('高级搜
索').click()
```

今日作业

【1】肯德基餐厅门店信息抓取（POST请求练习,非多线程）

1.1) URL地址: http://www.kfc.com.cn/kfccda/storelist/index.aspx

1.2) 所抓数据: 餐厅编号、餐厅名称、餐厅地址、城市

1.3) 数据存储: 保存到本地的json文件中,kfc.json

1.4) 程序运行效果:

请输入城市名称：北京

把北京的所有肯德基门店的信息保存到json文件

【2】链家二手房 ：使用多线程来实现

【3】豆瓣图书 ：使用多线程