

不实信息爬虫部分

用到的modules:

```
from selenium import webdriver
from pyquery import PyQuery as pq
import csv
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import time
from multiprocessing import Process
```

首先设置一个返回一个带头模式的浏览器的函数

```
def start_webdriver():
    # 登录浏览器
    options = webdriver.FirefoxOptions()
    # options.add_argument('--headless')
    # options.add_argument('--no-sandbox')
    # options.add_argument('--disable-dev-shm-usage')
    driver =
    webdriver.Firefox(options=options,executable_path='/usr/local/bin/geckodriver')
    return driver
```

然后利用start_webdriver()函数先启动一个浏览器,获取cookies信息,以便后面多线程里的浏览器使用

```
# 手动到微博的登录界面然后登录
driver=start_webdriver()
driver.get('https://weibo.com/login.php')
#手动登录完毕之后获取 cookies
cookies=driver.get_cookies()
driver.close()
```

data_explorer(driver,href)函数:

让driver跳转到href对应的界面后,运用我们对于不实信息网页分析得到的各个我们所需信息的xpath,进行相应的爬取.对于每条不实信息,我们总共需要获取以下内容:

```
[
    "举报人",
    "性别(举)",
```

```

        "所在地区(举)",
        "被举报人",
        "性别(被)",
        "所在地区(被)",
        "被举报人的微博主页",
        "被举报内容的微博网址",
        "被举报人信用等级",
        "被举报人是否是微博会员",
        "不实信息内有无超链接",
        "不实信息具体内容",
        "该举报在不实信息网站原文链接",
        "站方判定",
        "站方判定(全)",
        "被举报微博发布时间",
        "被举报微博发布时间(精确)",
    ]

```

data_explorer(driver,href)函数运行的思路是:

- 首先寻找页面右边被举报人的区域里面是否有"阅读全文"的标签, 如果有的话, 就把阅读原文的标签全部点击
- 然后, 根据各个xpath对于网页进行扫描, 获取需要的信息
- 将这些信息汇总成一个列表, 返回

```

def data_explorer(driver,href):
    Yuanwen_flag=0
    try:
        #把 右边的 阅读全文 全部点了
        t=driver.find_elements_by_xpath("//div[@class='part_report part_report_alone clearfix']/div[@class='W_main_half_r']/a[@sdata-track='key=tblog_service_account&value=view_original_text']")
        for i in t:
            i.click()
        if(t):
            Yuanwen_flag=1
            # print("有 原文 按钮")
        else:
            Yuanwen_flag=0
            # print("无 原文 按钮")
    except:
        Yuanwen_flag=0
        # print("无 原文 按钮")

#####
#####

if(Yuanwen_flag==1): #有 点击全文 按钮
    #被举报的信息
    try:

```

```

t=driver.find_element_by_xpath("//div[@class='W_layer']//div[@class='layer_feed
feed clearfix']//div[@class='con']")
    # if(t.find('#微博辟谣')):
    #     return
    ## 不实微博文本有无超链接
    if t.text.find('http:')!=-1 and t.text.find('https:')!=-1:
        message_link_whether=0
    else:
        message_link_whether=1
    people2_wrong_message=t.text[t.text.find(':')+1:].split('http:')
[0].split('https:')[0]
except:
    Yuanwen_flag=0
if(Yuanwen_flag==0): #无 点击全文 按钮
    #被举报的信息
    t=driver.find_element_by_xpath("//div[@class='part_report part_report_alone
clearfix']//div[@class='W_main_half_r']//div[@class='feed bg_orange2
clearfix']//div[@class='con']")
    ## 不实微博文本有无超链接
    if t.text.find('http:')!=-1 and t.text.find('https:')!=-1:
        message_link_whether=0
    else:
        message_link_whether=1
    people2_wrong_message=t.text[t.text.find(':')+1:].split('http:')
[0].split('https:')[0]

#####
#####
#举报人的性别
people1_sex=driver.find_element_by_xpath("//div[@class='part_report
clearfix']//div[@class='W_main_half_l']//p[@class='mb']//img").get_attribute("title
")
#举报人网名
people1_name=driver.find_element_by_xpath("//div[@class='part_report
clearfix']//div[@class='W_main_half_l']//div[@node-
type='report_user_area']//p[@class='mb W_f14']//a").text
#举报人所在地区
people1_area=driver.find_element_by_xpath("//div[@class='part_report
clearfix']//div[@class='W_main_half_l']//p[@class='mb']").text
#被举报人性别
people2_sex=driver.find_element_by_xpath("//div[@class='part_report
clearfix']//div[@class='W_main_half_r']//p[@class='mb']//img").get_attribute("title
")
#被举报人网名
people2_name=driver.find_element_by_xpath("//div[@class='part_report
clearfix']//div[@class='W_main_half_r']//div[@class='user bg_orange2
clearfix']//p[@class='mb W_f14']//a").text
#被举报人所在地区

```

```

        people2_area=driver.find_element_by_xpath("//div[@class='part_report
clearfix']/div[@class='W_main_half_r']/p[@class='mb']").text

# print('1')

        temp=driver.find_elements_by_xpath("//div[@class='part_report
clearfix']/div[@class='W_main_half_r']/p[@class='mb W_f14']/a")
        #被举报人微博
        people2_weibo=temp[0].get_attribute('href')
        #被举报人的信用等级
        t=driver.find_element_by_xpath("//div[@class='part_report
clearfix']/div[@class='W_main_half_r']/p[@class='mb
W_f14']/img").get_attribute('title')
        people2_credit_rating=t.split(': ')[1]

# print('2')
        #被举报信息原文链接
        try:

            people2_wrong_message_weibo_link=driver.find_element_by_xpath("//div[@class='part_
report part_report_alone
clearfix']/div[@class='W_main_half_r']/p[@class='publisher']/a").get_attribute('
href')

            # people2_delete_or_not='否'
        except:
            people2_wrong_message_weibo_link="###"
            # people2_delete_or_not='是'

        #不实信息发布时间
        t=driver.find_element_by_xpath("//div[@class='part_report part_report_alone
clearfix']/div[@class='W_main_half_r']/p[@class='publisher']")
        #发布的精确时间
        judge_time_precisely=t.text[t.text.find(': ')+1:].split("|")[0]
        if judge_time_precisely=="被举报微博":
            judge_time_precisely=""
        #发布时间
        judge_time=judge_time_precisely.split(" ")[0]
        if judge_time=="被举报微博":
            judge_time=""

        j=driver.find_element_by_xpath("//div[@class='middle middle_long']/p").text
        #站方判定(大概)
        official_judgement=j[j.find('"')+1:].split("'")[0]
        #站方判定(具体)
        official_judgement_all=j

        #该举报在不实信息网站原文链接
        judge_href=href

```

```

# print('3')

#被举报人是否是会员
try:
    try:
        driver.get(people2_weibo)
    except:
        driver.execute_script('window.stop ? window.stop() :
document.execCommand("Stop");')
        # switch_to_active(driver)
        driver.find_element_by_xpath("//a[@title='微博会员']")
        people2_whether_member=1
    except:
        people2_whether_member=0
        # driver.get(href)
        # switch_to_active(driver)
# print('4')

per_data=[
    people1_name,
    people1_sex,
    people1_area,
    people2_name,
    people2_sex,
    people2_area,
    people2_weibo,
    people2_wrong_message_weibo_link,
    people2_credit_rating,
    people2_whether_member,
    message_link_whether,
    people2_wrong_message,
    judge_href,
    official_judgement,
    official_judgement_all,
    judge_time,
    judge_time_precisely,
    # people2_delete_or_not
]

# print(per_data)
print("该条信息已经爬完")

return per_data

```

data_get(driver,href): 上面的data_explorer的外层函数。由于可能会出现页面一直在加载，但实际上页面的整体内容已经加载出来了的情况，最终会导致浏览器的超时。当浏览器跳转到对应的界面之后，一直如果还在加载并抛出异常的话，就执行相应的js语句，强行停止加载。

```

def data_get(driver,href):
    # tag.click()

    try:
        driver.get(href)

    except:
        driver.execute_script('window.stop ? window.stop() :
document.execCommand("Stop");')

    try:
        return data_explorer(driver,href)

    except:
        print("该条信息爬取失败")
        return

```

main_func: 最终由于cpu资源充足，我们选择利用多进程进行爬取，而main_func就是我们提交给每个进程需要执行的任务。里面涉及到了：

- 对于每个进程的监控：
 - 利用打印的信息，判断进程是否在运行
 - 利用对每个进程设置的日志文件（对应count_txt），判断当前已爬取的页数，以及爬取的每页的数据量
 - 利用日志文件里面开头和结尾的时间信息，判断每个进程最终运行的时间
- 对于爬取的每页数据（一个包含一页所有不实信息条目的href的列表），遍历每一个不实信息条目，运行data_explorer(driver,href)
- 对于每页数据的写入：
 - 由于计算机的 I/O 操作需要占用大量的cpu时间，所以我们选择在爬完一整页之后在进行数据的写入
- 目标500页都爬取完成之后，跳转到一个特殊网页作为标志

```

def main_func(start,need,date,number,cookies,home_page):
    driver=start_webdriver()
    driver.get('https://weibo.com/login.php')
    for i in cookies:
        driver.add_cookie(cookie_dict=i)
    print("第{}号浏览器开启成功".format(number))

    driver.set_page_load_timeout(10)

    # time.sleep(120)
    # for i in cookies:
    #     driver.add_cookie(i)
    count_txt="count_"+date+"_"+number+".txt"
    with open(count_txt, "a+", newline="") as g:
        writer1 = csv.writer(g)
        writer1.writerow(["第{}号进程正式开始于: {}".format(number,time.ctime())])
        print("第{}号进程正式开始于: {}".format(number,time.ctime()))
    g.close()

```

```

# driver.get('https://www.cnblogs.com/the-only-flash/')

with open("weiboWrongMessages_"+date+"_"+number+".csv", "a+", newline="") as f:
    writer = csv.writer(f)
    # 确定要爬取什么东西
    writer.writerow([
        "举报人",
        "性别(举)",
        "所在地区(举)",
        "被举报人",
        "性别(被)",
        "所在地区(被)",
        "被举报人的微博主页",
        "被举报内容的微博网址",
        "被举报人信用等级",
        "被举报人是否是微博会员",
        "不实信息内有无超链接",
        "不实信息具体内容",
        "该举报在不实信息网站原文链接",
        "站方判定",
        "站方判定(全)",
        "被举报微博发布时间",
        "被举报微博发布时间(精确)",

        # "被举报人是否注销用户"
        # "被举报人 微博 1",
        # "被举报人 微博 2",
        # "被举报人 微博 3",
        # "被举报人 微博 4",
        #"申诉结果"
    ])
    f.close()

#用循环实现爬取
for num in range(start,need+1):
    page_datas=[]
    print("现在开始爬取第{}页".format(num))
    try:
        driver.get( home_page+str(num) )
    except:
        driver.execute_script('window.stop ? window.stop() :
document.execCommand("Stop");')
    #获取该列表页所有不实信息标签

    all_tag=driver.find_elements_by_xpath('//div[@class="m_table_tit"]//a[@target="_blank"]')

    haven_count=0

    all_tag_href=[]
    for tag in all_tag:
        all_tag_href.append(tag.get_attribute('href'))

```

```

for href in all_tag_href:
    # driver.get(href)
    haven_count=haven_count+1
    print("第{}页 第{}条 数据".format(num,haven_count))
    #获取页面数据
    data_per=data_get(driver,href)
    if(not data_per):
        print("ERROR")
    page_datas.append(data_get(driver,href))

#一页的数据压进 csv
with open("weiboWrongMessages_"+date+"_"+number+".csv", "a+", newline="")
as w:

    writer = csv.writer(w)
    counted=0
    for i in page_datas:
        if(not i):
            continue
        # print(i)
        counted=counted+1
        writer.writerow(i)

    print("第{}页一共存了{}条数据@@@@@@@@@@@@@@@@".format(num,counted))
    w.close()

#每获取完一页的数据时候写一次，因为代码的准确性已经可以确定，现在要求最快速
with open(count_txt, "a+", newline="") as g:
    writer1 = csv.writer(g)
    writer1.writerow(["第{}页已爬完##### 一共存了{}条数据
@@@@@@@@@@@@@@@@@@@@".format(num,counted)])
    g.close()

driver.get('https://www.cnblogs.com/the-only-flash/')
with open(count_txt, "a+", newline="") as k:
    writer1 = csv.writer(k)
    writer1.writerow(["第{}号进程最终结束于: {}".format(number,time.ctime())])
    print("第{}号进程最终结束于: {}".format(number,time.ctime()))
    k.close()

```

创建多进程：查阅自己电脑的信息，发现一共有4个逻辑内核，最终选择利用4个进程


```
#start,need,date,number
processes = []
target_pages=500
num=4
average=int(target_pages/num)
date="5_10"
home_page="https://service.account.weibo.com/index?
type=5&status=4&page="+str(page)
for i in range(num):
    processes.append( Process(target=main_func,args=(average*i+1,average*
(i+1),date,str(i+1),cookies,home_page,)) )
```

四个进程最终平均运行时间大概为7个小时

所有进程一共爬取了9985条不实信息

爬取正常微博信息

有了上面关于爬取等量正常信息的经验，稍微修改代码过后，就可以得到爬取正常微博的代码。

在爬取正常信息的时候，我们并没有直接爬取之前不实信息发布者的正常微博，具体操作如下：

- 假设原来没有经过去重的所有爬取数据以用户信息来划分后的集合（无重复）为 a ，而经过去重之后剩下的不实微博数据为以用户信息来划分后的集合（无重复） b ，然后我们获得差集 $c = a - b$ 。
- 在差集 c 所包含的用户中，爬取正常的微博信息

最终我们爬取到了736条正常微博信息

我们这么处理的原因如下：对于我们确定的那9个数据特征，如果再去获取去重之后剩下的那些用户的正常信息，那么在性别，地区等特征上就会导致重复。

与正常信息爬取有关的代码如下：

- all_5_10.csv是爬取的所有不实信息
- 5_10_corrected_3.csv是经过多次去重之后得到的数据

```

all_data=pd.read_csv("all_5_10.csv")
data=pd.read_csv("5_10_corrected_3.csv")
all_data1=all_data["被举报人的微博主页"]
all_data2=all_data["被举报内容的微博网址"]
all_data3=all_data["被举报人"]
data1=data["被举报人的微博主页"]
data2=data["被举报内容的微博网址"]
data3=data["被举报人"]
cross_data3=list(set(all_data3)^set(data3))
cross_data1=list(set(all_data1)^set(data1))

```

然后我们需要取获取以下数据：

```

def right_get(driver,href,num):
    try:
        driver.get(href)
    except:
        driver.execute_script('window.stop ? window.stop() :
document.execCommand("Stop");')
        count=1
        while(count):
            if count > 2:
                break
            try:
                locator=(By.XPATH,"//a[@node-type='feed_list_item_date']")
                element =
WebDriverWait(driver,3).until(EC.presence_of_element_located(locator))
                break
            except:
                driver.refresh()
                count+=1
                continue
        # tt获取所有的“框”
        tt=driver.find_elements_by_xpath("//a[@node-type='feed_list_item_date']")
        all_href=[]
        for i in tt:
            all_href.append(i.get_attribute("href"))
        # print(len(tt))
        count=0
        message=[]
        for i in all_href:
            try:
                driver.get(i)
            except:
                driver.execute_script('window.stop ? window.stop() :
document.execCommand("Stop");')
                if count>=1:

```

```

        break

    # try:
    # tt[i].click()
    # driver.close()
    # switch_to_active(driver)
    # except:
    #     driver.execute_script('window.stop ? window.stop() :
document.execCommand("Stop");')

    locator=(By.XPATH,"//div[@class='WB_detail']//div[@class='WB_text W_f14']")
    while(1):
        try:
            element =
WebDriverWait(driver,3).until(EC.presence_of_element_located(locator))
            break
        except:
            driver.refresh()
            # element =
WebDriverWait(driver,60).until(EC.presence_of_element_located(locator))
            continue
        try:
            driver.find_element_by_xpath("//div[@class='W_tips tips_rederror
clearfix']")
            continue
        except:
            # time.sleep(5)
            is_hyperlink=0

    a=driver.find_elements_by_xpath("//div[@class='WB_detail']//div[@class='WB_text
W_f14']//a")

    b=driver.find_element_by_xpath("//div[@class='WB_detail']//div[@class='WB_text
W_f14']").text

    for i in a:
        if "L" in i.text:
            is_hyperlink=1
    b_raw=b
    for i in a:
        b="".join(b.split(i.text))
    if not b_raw:
        message.append("##")
        message.append("##")
        message.append("##")
    else:
        message.append(b)
        message.append(b_raw)
        message.append(is_hyperlink)

    count+=1

    print("第{}条数据 第{}条正常原文".format(num,count))

    # time.sleep(2)

```

```
# driver.close()
# driver.get(href)
# switch_to_active(driver)
return message
```