第十讲 综合案例二：通过PDF文本解析分析上市公司理财公告

这一章节我们将进行一个综合案例实战，通过之前的数据挖掘技术来批量下载巨潮资讯网上的上市公司的理财公告，并通过PDF文本解析技术来分析所获取的理财公告PDF，从中识别潜在的机构投资者（也即合适资金方）。

10.1 巨潮资讯网理财公告PDF批量下载实战

上一章最后一节讲了巨潮资讯网理财公告信息的爬取实战，其中还有些地方可以完善，如爬取多页内容；自动筛选所需内容，比如只要2018年的数据；以及完成理财报告PDF的批量自动下载，在这一小节就把这些知识点进行分析讲解。

**10.1.1 爬取巨潮资讯网多页内容**

在巨潮资讯网上批量爬取多页没有办法用之前爬取普通网站多页的手段：修改网url进行处理，因为对于巨潮资讯网：[http://www.cninfo.com.cn/new/fulltextSearch?notautosubmit=&keyWord=理财](http://www.cninfo.com.cn/new/fulltextSearch?notautosubmit=&keyWord=%E7%90%86%E8%B4%A2)，在进行翻页操作的时候，网址的链接并没有发生变化，因此就没有办法利用网址的变化规律来进行批量爬取多页了。此时解决办法为利用selenium来模拟鼠标点击下图所示的“下一页”按钮，并根据公告数量来确定模拟点击次数，每点击完就获取一下该页的网页源代码，最后把获取到的各页的源代码放到一个列表里。



首先来完成自动模拟点击按钮的操作，这个和8.2.4节讲过的模拟点击“百度一下”那个按钮的道理是一样的，只要获取到这个“下一页”按钮的xpath内容，就可以用selenium进行模拟点击了。在网页上利用F12调出开发者工具，然后选中翻页按钮对应的位置，右键选择Copy，然后选中Copy Xpath，把Xpath内容复制到代码里即可，代码如下：

from selenium import webdriver

import time

browser = webdriver.Chrome()

url = 'http://www.cninfo.com.cn/new/fulltextSearch?notautosubmit=&keyWord=理财'

browser.get(url)

time.sleep(3)

browser.find\_element\_by\_xpath('//\*[@id="fulltext-search"]/div/div[1]/div[2]/div[4]/div[2]/div/button[2]').click()

运行之后发现它已经可以自动翻页跳转到第二页了，这边引入time模块，在访问网址后写一个time.sleep(3)，一是为了方便演示点击翻页的过程，二是防止浏览器页面还处于刷新阶段。

那如果想要把所有的内容都获取下来该如何处理呢，比如这边数万条理财公告的信息。我们只需要获取到所有公告的数量，这个数量在网页上的右下角有显示，然后因为巨潮资讯网每页是显示十条内容，所以把这个数量除以十，那么就是我们需要翻页的次数了，利用F12和观察源代码可以发现该数量的正则规律如下：

p\_count = '<span class="total-box" style="">约(.\*?)条'

获取公告条数和一共多少页的页数的完整代码如下：

from selenium import webdriver

import time

browser = webdriver.Chrome()

url = 'http://www.cninfo.com.cn/new/fulltextSearch?notautosubmit=&keyWord=理财'

browser.get(url)

time.sleep(3)

data = browser.page\_source

p\_count = '<span class="total-box" style="">约(.\*?)条'

count = re.findall(p\_count, data)[0]

pages = int(int(count)/10)

其中有一些注意点，在写公告数量count = re.findall(p\_count, data)[0]的时候，一定要记得在findall最后写一个[0]，因为findall获取的是一个列表，尽管这里只有一个元素，仍然要写一个[0]来提取列表中的元素。另外一个重要的注意点是，在写一共多少页的页数pages = int(int(count)/10)的时候，要写两个int函数。其中首先int(count)是把获取到count转换成数字，因为利用findall获取到的内容虽然长得像数字，但其本质是字符串，需要用int函数转换后才能进行除以10的操作。第二个int的原因是因为int(count)除以十之后变成含小数点的数了，但是页数只能是整数，所以需要利用int函数将其转换成整数。

我们把自动翻页和获取页数的代码写到一块，同时每次翻页都获取下当页的源代码，并把每一页的源代码储存到一个叫作datas的列表中。这里为了严谨，引入time模块，每次翻页和获取该页的源码之后都休息2秒及1秒再进行下一步操作。

datas = []

datas.append(data) # 这边是把刚刚获取的第一页源代码先放到datas这个列表里

for i in range(pages): # 刚刚获取到的pages页码数

browser.find\_element\_by\_xpath('//\*[@id="fulltext-search"]/div/div[1]/div[2]/div[4]/div[2]/div/button[2]').click()

time.sleep(2)

data = browser.page\_source

datas.append(data)

time.sleep(1)

下面一个需要完成的工作就是把datas这个列表转换成字符串，因为只有在字符串中才能进行正则表达式的编写，从而提取我们想要的各个页面的公告的标题，下载地址，日期等信息。通过**"连接符".join(列表名)**的方式可以把列表转换成字符串，其中引号里的内容为列表各元素间的连接符，这边选择的空字符串作为连接符：

alldata = "".join(datas)

之后只需要把上一讲讲的爬取标题，链接及日期的正则表达式应用到这个所有页面源代码汇总的alldata上面即可：

p\_title = '<span title="" class="r-title">(.\*?)</span>'

p\_href = '<a target="\_blank" href="(.\*?)data-id='

p\_date = '<span class="time">(.\*?)</span>'

title = re.findall(p\_title, alldata)

href = re.findall(p\_href, alldata)

date = re.findall(p\_date, alldata, re.S)

所有代码汇总如下：

from selenium import webdriver

import re

import time

browser = webdriver.Chrome()

url = 'http://www.cninfo.com.cn/new/fulltextSearch?notautosubmit=&keyWord=理财'

browser.get(url)

time.sleep(3)

data = browser.page\_source

p\_count = '<span class="total-box" style="">约(.\*?)条'

count = re.findall(p\_count, data)[0] # 获取公告个数，注意这里要加一个[0],因为findall返回的是一个列表

pages = int(int(count)/10)

# 1.自动翻页获取源码源代码

datas = []

datas.append(data) # 这边是把第一页源代码先放到datas这个列表里

for i in range(3): # 这边为了演示改成了range(3)，想爬全部的话改成range(pages)

browser.find\_element\_by\_xpath('//\*[@id="fulltext-search"]/div/div[1]/div[2]/div[4]/div[2]/div/button[2]').click()

time.sleep(2) # 防止页面没有加载完毕

data = browser.page\_source

datas.append(data)

time.sleep(1)

alldata = "".join(datas)

browser.quit()

# 2.编写正则表达式

p\_title = '<span title="" class="r-title">(.\*?)</span>'

p\_href = '<a target="\_blank" href="(.\*?)data-id='

p\_date = '<span class="time">(.\*?)</span>'

title = re.findall(p\_title, alldata)

href = re.findall(p\_href, alldata)

date = re.findall(p\_date, alldata, re.S)

# 3.清洗数据

for i in range(len(title)):

title[i] = re.sub('<.\*?>', '', title[i])

href[i] = 'http://www.cninfo.com.cn' + href[i]

href[i] = re.sub('amp;', '', href[i])

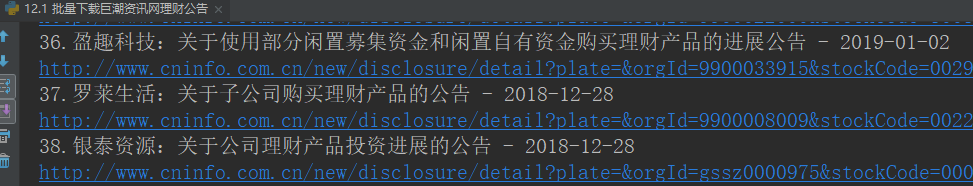
date[i] = date[i].strip()

date[i] = date[i].split(' ')[0]

print(str(i + 1) + '.' + title[i] + ' - ' + date[i])

print(href[i])

其中for i in range(pages):这一行代码写的是for i in range(3):主要是为了快速展示，即点击3次翻页按钮，先爬取4页给大家看看效果，如果想爬全部，可以把3换成pages即可。最后获取的效果如下图所示，利用上面的代码，我们可以获取所有公告的标题、网址和日期信息：



如果不想看到弹出的模拟浏览器，可以仿照上一章的内容进行无界面浏览器设置。

**10.1.2 自动筛选所需内容**

如何把爬到的内容进行筛选呢，比如说这里只想要日期为2019年的内容，或者2019年和2020年的内容。这里可以通过if语句把不符合条件的内容都赋值为空字符串，如果符合条件的，则让它等于原来的值，这和第五章数据清洗的相关代码类似，代码如下：

for i in range(len(title)):

if '2019' in date[i] or '2020' in date[i]: #2019和2020年的

title[i] = title[i]

href[i] = href[i]

date[i] = date[i]

else:

title[i] = ''

href[i] = ''

date[i] = ''

这样不符合日期为2019年或者2020年的内容，都会被处理为一个空字符串，不过这些空字符串还是存在于title，href，以及date这些列表里的，所以还需要把这些空字符串给清除掉。我们利用while循环遍历这些个列表，一旦发现有某个元素是空字符串，就直接用**列表.remove（元素）**的办法把这些空字符串给剔除掉。代码如下：

while '' in title:

title.remove('')

while '' in href:

href.remove('')

while '' in date:

date.remove('')

完整代码如下所示，我们只需要把该代码放到上一节讲的代码之后即可：

for i in range(len(title)):

if '2019' in date[i] or '2020' in date[i]: #2019和2020年的

title[i] = title[i]

href[i] = href[i]

date[i] = date[i]

else:

title[i] = ''

href[i] = ''

date[i] = ''

while '' in title:

title.remove('')

while '' in href:

href.remove('')

while '' in date:

date.remove('')

利用同样的思路，我们也能对标题进行清洗，如果该标题里还有某些不想要的关键词，比如说“保底”、“刚兑”等关键词，我们就把它赋值为空值，代码如下：

for i in range(len(title)):

if '保底' in title[i] or '刚兑' in title[i]:

title[i] = ''

href[i] = ''

date[i] = ''

else:

title[i] = title[i]

href[i] = href[i]

date[i] = date[i]

while '' in title:

title.remove('')

while '' in href:

href.remove('')

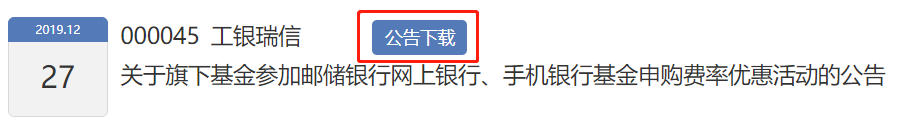
while '' in date:

date.remove('')

这里的else里的代码其实也可以不写，因为如果没有触发if的条件，原内容也不会改变，执行的就是else里的内容。这边还有个注意点，在通过关键词筛选之前，一定要先进行上一章讲过的数据清洗，因为未清洗的标题里有很多类似<.\*?>的内容，这些内容会把标题里的文本分割开来，导致if判断条件失效。

**10.1.3 理财公告PDF的自动批量下载**

这里首先讲解如何利用selenium从网页上下载一个PDF。如下图所示，对于一个刚刚获取到的PDF下载地址：<http://www.cninfo.com.cn/new/disclosure/detail?orgId=jjjl0000090&announcementId=1207206145&announcementTime=2019-12-27>，我们需要做的就是让selenium模拟点击一下下图的下载按钮即可：



这个的模拟点击其实和上一节讲过的模拟点击“下一页”那个按钮很相似，同样是用F12获取它的xpath内容或者css\_selector内容，然后用selenium模拟点击即可，通过F12可以获得该下载按钮的xpath内容为：'/html/body/div/div[1]/div[2]/div[1]/a[3]'，然后通过下面的四行代码就能够实现PDF的自动下载了：

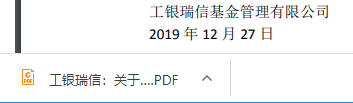
from selenium import webdriver

browser = webdriver.Chrome()

browser.get('http://www.cninfo.com.cn/new/disclosure/detail?orgId=jjjl0000090&announcementId=1207206145&announcementTime=2019-12-27')

browser.find\_element\_by\_xpath('/html/body/div/div[1]/div[2]/div[1]/a[3]').click()

运行结果如下，可以看到已经可以实现PDF的自动下载了：



那如何来实现批量下载呢，其实只要写一个for循环即可。我们之前已经获得了一个下载链接的列表href，利用for循环遍历该列表，就能实现把列表里所有的内容都下载下来了，代码如下：

for i in range(len(href)):

browser = webdriver.Chrome()

browser.get(href[i])

browser.find\_element\_by\_xpath('/html/body/div/div[1]/div[2]/div[1]/a[3]').click()

time.sleep(3)

browser.quit()

因为是批量的下载并且PDF的下载是需要时间的，所以这里得加一个time.sleep(3)来让它每次点击完下载按钮后有3秒的下载时间（如果文件大的话，则需要更长的下载时间），然后再利用browser.quit()退出模拟浏览器执行下一轮循环。

这边还有个需要注意的细节，有的页面里，它其实并没有PDF，只有几行字的告示（这个很少见，但的确存在），那么也就没有所谓的PDF下载按钮了。此时如果执行上面的代码就会出现错误，就得补写一个try except函数来避免程序报错，代码如下：

for i in range(len(href)):

browser = webdriver.Chrome()

browser.get(href[i])

try:

browser.find\_element\_by\_xpath('/html/body/div/div[1]/div[2]/div[1]/a[3]').click()

time.sleep(3) # 这个一定要加，因为下载需要一点时间

browser.quit()

print(str(i+1) + '.' + title[i] + '下载完毕)

except:

print(title[i] + '不是PDF文件')

只需要把这几行代码放到上一节讲过的代码后面，就可以实现一个理财报告PDF的批量自动下载了。其完整代码如下所示：

from selenium import webdriver

import re

import time

browser = webdriver.Chrome()

url = 'http://www.cninfo.com.cn/new/fulltextSearch?notautosubmit=&keyWord=理财'

browser.get(url)

time.sleep(3)

data = browser.page\_source

p\_count = '<span class="total-box" style="">约(.\*?)条'

count = re.findall(p\_count, data)[0] # 获取公告个数，注意这里要加一个[0],因为findall返回的是一个列表

pages = int(int(count)/10)

# 1.自动翻页获取源码源代码

datas = []

datas.append(data) # 这边是把第一页源代码先放到datas这个列表里

for i in range(3): # 这边为了演示改成了range(3)，想爬全部的话改成range(pages)

browser.find\_element\_by\_xpath('//\*[@id="fulltext-search"]/div/div[1]/div[2]/div[4]/div[2]/div/button[2]').click()

time.sleep(2)

data = browser.page\_source

datas.append(data)

time.sleep(1)

alldata = "".join(datas)

browser.quit()

# 2.编写正则表达式

p\_title = '<span title="" class="r-title">(.\*?)</span>'

p\_href = '<a target="\_blank" href="(.\*?)data-id='

p\_date = '<span class="time">(.\*?)</span>'

title = re.findall(p\_title, alldata)

href = re.findall(p\_href, alldata)

date = re.findall(p\_date, alldata, re.S)

# 3.清洗数据

for i in range(len(title)):

title[i] = re.sub('<.\*?>', '', title[i])

href[i] = 'http://www.cninfo.com.cn' + href[i]

href[i] = re.sub('amp;', '', href[i])

date[i] = date[i].strip()

date[i] = date[i].split(' ')[0]

print(str(i + 1) + '.' + title[i] + ' - ' + date[i])

print(href[i])

# 4.自动筛选

for i in range(len(title)):

if '2019' in date[i] or '2020' in date[i]: # 筛选2019和2020年的，可以自己调节

title[i] = title[i]

href[i] = href[i]

date[i] = date[i]

else:

title[i] = ''

href[i] = ''

date[i] = ''

while '' in title:

title.remove('')

while '' in href:

href.remove('')

while '' in date:

date.remove('')

# 5.自动批量爬取PDF - 选择默认储存位置

for i in range(len(href)):

browser = webdriver.Chrome()

browser.get(href[i])

try:

browser.find\_element\_by\_xpath('/html/body/div/div[1]/div[2]/div[1]/a[3]').click()

time.sleep(3) # 这个一定要加，因为下载需要一点时间

browser.quit()

print(str(i+1) + '.' + title[i] + '下载完毕')

except:

print(title[i] + '不是PDF文件')

10.2　PDF文本解析基础

代码文件：10.2　PDF解析文本基础.py

在上一小节，我们已经可以通过公告标题来做一些初步的筛选，那么如果想做一些更深层次的筛选的话，比如想要深入到PDF正文内容进行分析的话，就需要用到PDF文本解析技术。Python中有多个可以用于解析PDF文本的库，如pdfplumber库、pdfminer库、tabula库等，经笔者测试，pdfplumber库是目前使用最方便的库，而且pdfplumber库不仅可以解析文字，还可以方便的解析PDF文件中的表格。

这里推荐通过pip安装法安装pdfplumber库，具体方法见1.4.4节，要输入的命令是pip install pdfplumber，如果直接pip安装有问题的话，可以参考1.4.4节中的补充知识点采用清华镜像pip安装法。