Lab1 互联网信息获取与解析

罗思佳2021201679

一、实验目的

从新浪股票(https://finance.sina.com.cn/stock/) 爬取同一家公司的100条公告，以csv格式存储公告内容

•基础要求：

1.csv列属性包括：**标题**、**日期**和**正文**✔

2.100条公告**不重复**✔

3.绘制词云✔

•加分：

4.比较不同html解析方法时间效率✔

5.csv文件增加属性：类别✔

(✔表示已完成)

二、实验内容

实验内容包括以下三个步骤：

1.对网页进行爬取

这里使用的是Scrapy框架进行爬取。首先需要获取网页的url，以及每条公告的标题、类型、日期、正文页的链接在html中的位置，然后逐页进行爬取，对于每一条公告还需要进入详情页链接爬取正文内容，爬取完成后使用scrapy的特定命令将数据保存在csv文件中。

2.处理爬取到的数据

对爬取到的数据进行清洗和处理，包括调整列顺序、删除空行、删除重复值等。

3.绘制词云

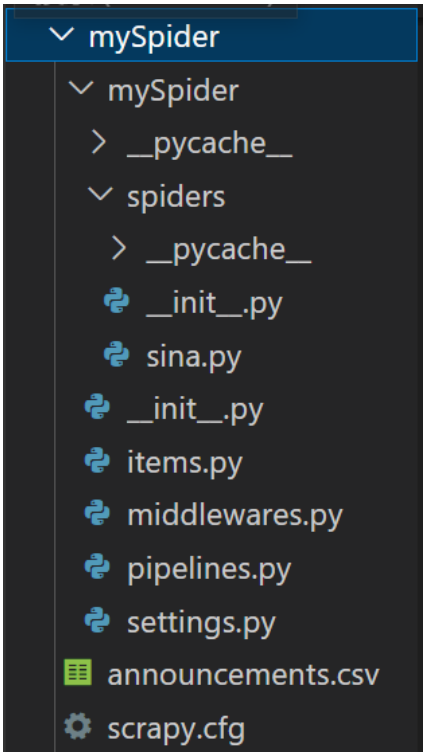
分别对公告的标题和正文进行词云的绘制。

三、实验设计与分析

（一）基于scrapy爬取网页

1.新建项目

使用命令`scrapy startproject mySpider`创建一个新的Scrapy项目，构建scrapy框架工程文件，设置`setting.py`配置信息。



这些文件分别是：

- scrapy.cfg: 项目的配置文件。
- mySpider/: 项目的Python模块，将会从这里引用代码。
- mySpider/items.py: 项目的目标文件。
- mySpider/pipelines.py: 项目的管道文件。
- mySpider/settings.py: 项目的设置文件。
- mySpider/spiders/: 存储爬虫代码目录。

2.明确目标

需要爬取的是上市公司公告*新浪财经*新浪网 (sina.com.cn)里的公告标题、日期、类型和正文。在mySpider/items.py里创建一个`SinaItem`类，并且定义相关的类属性

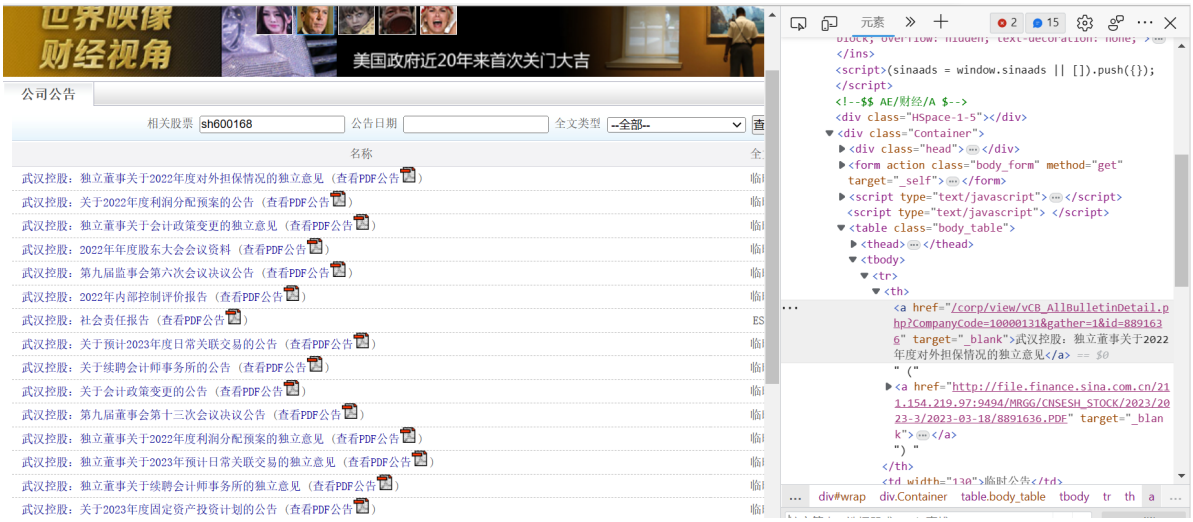
```
class SinaItem(scrapy.Item):
    # define the fields for your item here like:
    title = scrapy.Field()
    date = scrapy.Field()
    type = scrapy.Field()
    detail_page = scrapy.Field()
```

3.制作爬虫

在终端输入命令`scrapy genspider sina "finance.sina.com.cn"`，从而在mySpider/spider目录下创建一个名为sina的爬虫，并指定爬取域的范围。

（1）获取标题目录页面情况

以“武汉控股”公司为例, start_urls="http://vip.stock.finance.sina.com.cn/corp/view/vCB_BulletinGather.php?stock_str=sh600168&page_index=1"

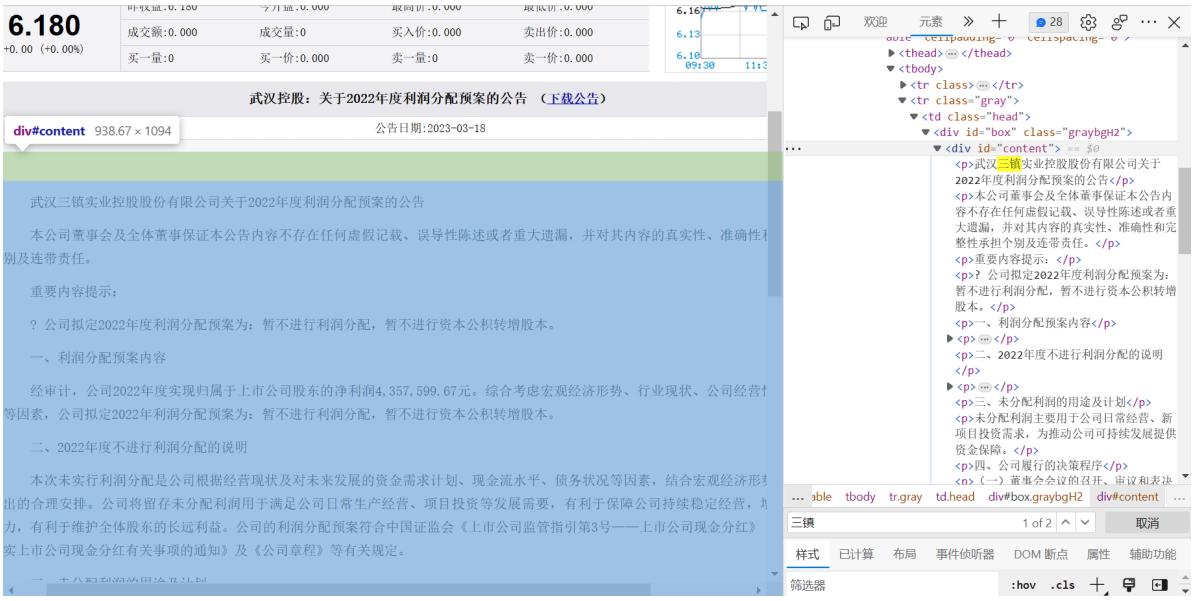


获取各标签下的公告标题，xpath为：`/html/body/div/div[5]/table/tbody/tr/th/a`

获取各标签下的类型：`/html/body/div/div[5]/table/tbody/tr/td[1]`

获取各标签下的公告日期：`/html/body/div/div[5]/table/tbody/tr/td[2]`

点击标题进入详情页，获取正文内容



获取页面内容：
`//*[@id="content"]//text()`

编写代码

`<tbody>`标签下的每一个`<tr>`标签代表一条公告，包括**公告标题、全文类型和日期**，因此需要对其进行遍历，将三个属性存入创建的SinaItem对象中；并且对于每一条公告，需要获取详情页的url，调用`parse_detail`函数进入详情页链接爬取**正文**内容。

```
import scrapy
from mySpider.items import SinaItem


class SinaSpider(scrapy.Spider):
    name = "sina"
    allowed_domains = ["finance.sina.com.cn"]
    start_urls = [
        "http://vip.stock.finance.sina.com.cn/corp/view/vCB_BulletinGather.php?
stock_str=sh600168&page_index=1"
```

```
    ]

    # 回调函数接受item
    def parse_detail(self, response):
        item = response.meta['item']
        detail_page = response.xpath('//*[@id="content"]//text()').extract()
        detail_page = ''.join(detail_page)
        item['detail_page'] = detail_page
        # print(detail_page)
        yield item

    # 解析首页中的标题名称
    def parse(self, response):
        tbody_list = response.xpath('/html/body/div/div[5]/table/tbody')

        for tbody in tbody_list: # 遍历tbody下的tr标签
            tr_list = tbody.xpath('./tr')
            for tr in tr_list:
                item = SinaItem()
                title = tr.xpath('./th/a/text()').extract_first()
                item['title'] = title
                # print(title)
                date = tr.xpath('./td[2]/text()').extract_first()
                item['date'] = date
                # print(date)
                type = tr.xpath('./td[1]/text()').extract_first()
                item['type'] = type
                # print(type)
                detail_url = "http://vip.stock.finance.sina.com.cn" + tr.xpath(
                    './th/a/@href').extract_first()  # 链接里的内容
                # print(detail_url)
                yield scrapy.Request(detail_url,
                                     callback=self.parse_detail,
                                     dont_filter=True,
                                     meta={'item': item})
```

（2）爬取多页面

因为每页只有30条公告，需要爬取100条，所以需要逐页进行爬取。

实现方法：添加一个变量`index`记录页数，每爬完一个页面就更新url

加入了分页操作的代码：

```
import scrapy
from mySpider.items import SinaItem


class SinaSpider(scrapy.Spider):
    name = "sina"
    allowed_domains = ["finance.sina.com.cn"]
    start_urls = [
        "http://vip.stock.finance.sina.com.cn/corp/view/vCB_BulletinGather.php?
stock_str=sh600168&page_index=1"
    ]
```

```
    url =
"http://vip.stock.finance.sina.com.cn/corp/view/vCB_BulletinGather.php?
stock_str=sh600168&page_index=%d"
    index = 1

    # 回调函数接受item
    def parse_detail(self, response):
        item = response.meta['item']
        detail_page = response.xpath('//*[@id="content"]//text()').extract()
        detail_page = ''.join(detail_page)
        item['detail_page'] = detail_page

        # print(detail_page)

        yield item

    # 解析首页中的标题名称
    def parse(self, response):
        tbody_list = response.xpath('/html/body/div/div[5]/table/tbody')

        for tbody in tbody_list:
            tr_list = tbody.xpath('./tr')
            for tr in tr_list:
                item = SinaItem()
                title = tr.xpath('./th/a/text()').extract_first()
                item['title'] = title
                # print(title)
                date = tr.xpath('./td[2]/text()').extract_first()
                item['date'] = date
                # print(date)
                type = tr.xpath('./td[1]/text()').extract_first()
                item['type'] = type
                # print(type)
                detail_url = "http://vip.stock.finance.sina.com.cn" + tr.xpath(
                    './th/a/@href').extract_first()  # 链接里的内容
                # print(detail_url)
                yield scrapy.Request(detail_url,
                                     callback=self.parse_detail,
                                     dont_filter=True,
                                     meta={'item': item})

                # 分页操作
                if self.index <= 5: # 爬取前五页
                    new_url = format(self.url % self.index)
                    self.index += 1

                    yield scrapy.Request(new_url,
                                         callback=self.parse,
                                         dont_filter=True)
```

4.保存数据

在终端输入命令`scrapy crawl sina -o anouncements.csv`,将爬取到的数据存入csv文件

（二）数据清洗

对爬取到的数据进行清洗和词云的绘制

导入相关包

```
import pandas as pd
import numpy as np
from PIL import Image
import matplotlib
import re
import csv
```

读取csv文件，调整列的顺序

```
df=pd.read_csv('./mySpider/announcements.csv')
order=['title','date','type','detail_page']
df=df[order] # 调整列顺序
print(df.shape) #打印形状
df.head()
```

运行结果：包括了标题、日期、类型和正文

```
(180, 4)
```

	title	date	type	detail_page
0	武汉控股：独立董事关于会计差错更正事项的独立意见	2023-03-15	临时公告	\r\n\t\t\t\t\t\t公告内容详见附件\t\t\t\t\t\t
1	武汉控股：审计机构出具的关于武汉控股前期会计差错更正专项说明的审核报告	2023-03-15	更正或补充	\r\n\t\t\t\t\t\t公告内容详见附件\t\t\t\t\t\t
2	武汉控股：第九届董事会第十二次会议决议公告	2023-03-15	临时公告	\r\n\t\t\t\t\t\t武汉三镇实业控股股份有限公司第九届董事会第十二次会议决议公告...
3	武汉控股：重大事项进展公告	2022-12-31	临时公告	\r\n\t\t\t\t\t\t武汉三镇实业控股股份有限公司重大事项进展公告本公司董事会及全...
4	武汉控股：2022年第三次临时股东大会决议	2022-12-31	临时公告	\r\n\t\t\t\t\t\t公告内容详见附件\t\t\t\t\t\t

```
df.isnull().sum() # 判断是否有空值
```

```
title          0
date           0
type           0
detail_page    0
dtype: int64
```

```
# 删除正文为“公告内容详见附件”的行
df2=df[df.detail_page.str.len() > 100]
print(df2.shape)
```

```
# 判断是否有重复
df3=df2.groupby(["title"]).size()
col=df3[df3>1].reset_index()[["title"]]
pd.merge(col,df2,on=["title"])
```

```
#去除重复值
df_clean=df2.drop_duplicates(subset=['title'],keep='first',inplace=False)
print(df_clean.shape)
```

清洗后的`df_clean.shape`如下

```
(108, 4)
```

```
# 按日期排序
df_clean['date'] = pd.to_datetime(df_clean['date'])
df_clean = df_clean.sort_values(by='date',ascending=False)
```

保存

```
# 取前100行
df_clean=df_clean.iloc[:100]
# 保存到csv
df_clean.to_csv('./announce_clean.csv', index=0,encoding='utf-8')
```

将标题和正文写入txt文件，用于词云图的绘制

```
# 将标题写入txt
df_clean['title'].to_csv('title.txt', sep=' ',
index=False,header=False,quoting=csv.QUOTE_NONE,escapechar=' ')
```

```
# 将正文写入txt
df_clean['detail_page'].to_csv('detail_page.txt', sep=' ',
index=False,header=False,quoting=csv.QUOTE_NONE,escapechar=' ')
```

（三）绘制词云

导入相关包

```
import jieba
from wordcloud import WordCloud
from wordcloud import STOPWORDS
import matplotlib.pyplot as plt
```

1.绘制公告标题的词云

导入文本

```
title_text = open("./title.txt",encoding='utf8').read() # 导入title文本
title_text = title_text.replace('\n',"").replace("\u3000","") # 去除换行符和空格
```

分词

```
# 分词，返回结果为词的列表
title_cut = jieba.lcut(title_text)
# 将分好的词用空格分割开连成字符串
title_cut = ' '.join(title_cut)
```

导入停词

```
# 导入停词
stop_words = open("./hit_stopwords.txt",encoding="utf8").read().split("\n")
```

绘制词云

```
# 背景
background = Image.open("./diamond.jpg")
graph = np.array(background)
# 绘制词云
word_cloud = WordCloud(font_path="./simsun.ttc",
                       background_color="white",
                       mask=graph,
                       stopwords=stop_words)

word_cloud.generate(title_cut)
plt.subplots(figsize=(12,8))
plt.imshow(word_cloud)
plt.axis("off")
```

公告标题的词云图如下

2.绘制正文的词云

导入文本

```
detail_text = open("./detail_page.txt",encoding='utf8').read() # 导入文本
detail_text = detail_text.replace('\n',"").replace("\u3000","") # 去除换行符和空格
```

分词

```
# 分词，返回结果为词的列表
detail_cut = jieba.lcut(detail_text)
# 将分好的词用某个符号分割开连成字符串
detail_cut = ' '.join(detail_cut)
```

绘制词云

```
background2 = Image.open("./pikaqiu.jpg")
graph2 = np.array(background2)

word_cloud2 = WordCloud(font_path="./新蒂黑板报体.ttf",
                        background_color="white",
                        mask=graph2,
                         stopwords=stop_words )

word_cloud2.generate(detail_cut)
plt.subplots(figsize=(12,8))
plt.imshow(word_cloud2)
plt.axis("off")
```

正文词云图如下（背景是一只皮卡丘🐱）

扩展延伸

使用Selenium框架爬虫

我最开始用的是Scrapy框架（见上述过程），虽然实验没有要求用不同的框架，我后来又尝试了Selenium框架来进行爬取，代码如下：

```
from selenium import webdriver
```

```
import time
from selenium.webdriver.common.by import By
import datetime
import pandas as pd

starttime = datetime.datetime.now()
data = {'title': [1] * 30, 'date': [2] * 30, 'content': [3] * 30}

df = pd.DataFrame(data)
driver = webdriver.Chrome()
driver.get(
    "http://vip.stock.finance.sina.com.cn/corp/view/vCB_BulletinGather.php?
stock_str=sh600168&page_index=1"
)
time.sleep(3)  # 加载时间

links = driver.find_elements(
    By.XPATH, "/html/body/div/div[5]/table/tbody/tr/th/a[1]")  # 获取所有a标签组成列
表
length = len(links)  # 一共有多少个标签

for i in range(0, length):  # 遍历列表的循环，逐一点击链接
    links = driver.find_elements(
        By.XPATH,
        "/html/body/div/div[5]/table/tbody/tr/th/a[1]")  # 每次循环时都重新获取a标
签，组成列表
    link = links[i]  # 逐一将列表里的a标签赋给link
    url = link.get_attribute('href')  # 提取a标签内的链接（type:str）
    driver.get(url)
    time.sleep(1)  # 加载时间

    title = driver.find_element(
        By.XPATH, "//*[@id='allbulletin']/thead/tr/th").text  # 纯文本内容
    # print(title)
    df.loc[i, 'title'] = title

    date = driver.find_element(By.XPATH,
                               "//*[@id='allbulletin']/tbody/tr[1]/td").text
    df.loc[i, 'date'] = date

    content = driver.find_element(By.XPATH, "//*[@id='content']").text
    # print(content)
    df.loc[i, 'content'] = content
    # print("\n")
    driver.back()  # 后退，回到原始页面目录页
    time.sleep(1)  # 留出加载时间

print("count:" + str(length))  # 打印列表长度，即有多少篇公告

df.to_csv('./crawler2.csv', index=0, encoding='utf-8')
endtime = datetime.datetime.now()
print("running time:" + str((endtime - starttime).seconds) + "s")
```

两者的使用感受是：Scrapy是一个完整的Python框架，提供了大量的内置功能和代码，因此编写少量代码便可实现复杂的任务，但是不太容易上手，最开始的学习成本较高；Selenium实际上是一个网络自动化库，使用时可以看到爬取的过程（打开浏览器，然后模拟点击网页），上手较容易，但爬取速度较慢。

四、实验结果

4.1 爬取内容

选取了“武汉控股”公司进行爬取，爬取的列属性为 **公告标、日期、类型、正文**

爬取的同一家公司的100条公告保存在`mySpider/announcements.csv`文件里，如下图（包括了类型）

```
mySpider > announcements.csv
date,detail_page,title,type
2023-03-15,"
            公告内容详见附件                ",武汉控股：独立董事关于会计差错更正事项的独立意
            见,临时公告
2023-03-15,"
            公告内容详见附件                ",武汉控股：审计机构出具的关于武汉控股前期会计差错
            更正专项说明的审核报告,更正或补充
2023-03-15,"
            武汉三镇实业控股股份有限公司第九届董事会第十二次会议决议公告本公司董事会及全体董事保证本公
            告内容不存在任何虚假记载、误导性陈述或者重大遗漏，并对其内容的真实性、准确性和完整性承担个
            别及连带责任。一、董事会会议召开情况武汉三镇实业控股股份有限公司（以下简称“公司”）第九届
            董事会第十二次会议通知于2023年3月7日以书面方式通知各位董事，会议于2023年3月14日上午以通
            讯表决的形式召开。本次会议应参加表决董事10人，实际参加表决董事10人。本次会议的召开符合
            《公司法》和《公司章程》及其他有关法律法规的规定，会议合法有效。二、董事会会议审议情况会议
            通过认真审议，采取记名投票方式逐项表决，通过了如下议案：（一）关于会计差错更正的议案武汉汉
            西污水处理有限公司（以下简称“汉西污水”）系公司子公司武汉市城市排水发展有限公司（以下简称
            “排水公司”）的联营企业，截止至2022年12月31日排水公司持有其20%的股权。公司于2022年12月收
            到汉西污水调整后财务报表，根据调整后的财务报表显示，汉西污水对营业收入进行了调整，主要原因
            系汉西污水于2019年向政府部门提出污水处理价格调价申请，并于2021年7月与武汉市水务局及武汉市
            东西湖区人民政府签订的《武汉市汉西污水处理厂特许经营协议补充协议（二）》，根据审批后的协议
            约定，自2019年5月起汉西污水的污水处理服务价格由1.139元/立方米调整为1.527元/立方米，汉西
            污水根据上述协议及文件调整了财务报表。本公司收到汉西污水调整后的财务报表，依据《企业会计准
            则第28号—会计政策、会计估计变更和会计差错更正》及《公开发行证券的公司信息披露编报规则第19
```

清洗后的数据存在`announce_clean.csv`里

```
announce_clean.csv
title,date,type,detail_page
武汉控股：第九届董事会第十二次会议决议公告,2023-03-15,临时公告,"
            武汉三镇实业控股股份有限公司第九届董事会第十二次会议决议公告本公司董事会及全体董事保证本公
            告内容不存在任何虚假记载、误导性陈述或者重大遗漏，并对其内容的真实性、准确性和完整性承担个
            别及连带责任。一、董事会会议召开情况武汉三镇实业控股股份有限公司（以下简称“公司”）第九届
            董事会第十二次会议通知于2023年3月7日以书面方式通知各位董事，会议于2023年3月14日上午以通
            讯表决的形式召开。本次会议应参加表决董事10人，实际参加表决董事10人。本次会议的召开符合
            《公司法》和《公司章程》及其他有关法律法规的规定，会议合法有效。二、董事会会议审议情况会议
            通过认真审议，采取记名投票方式逐项表决，通过了如下议案：（一）关于会计差错更正的议案武汉汉
            西污水处理有限公司（以下简称“汉西污水”）系公司子公司武汉市城市排水发展有限公司（以下简称
            “排水公司”）的联营企业，截止至2022年12月31日排水公司持有其20%的股权。公司于2022年12月收
            到汉西污水调整后财务报表，根据调整后的财务报表显示，汉西污水对营业收入进行了调整，主要原因
            系汉西污水于2019年向政府部门提出污水处理价格调价申请，并于2021年7月与武汉市水务局及武汉市
            东西湖区人民政府签订的《武汉市汉西污水处理厂特许经营协议补充协议（二）》，根据审批后的协议
            约定，自2019年5月起汉西污水的污水处理服务价格由1.139元/立方米调整为1.527元/立方米，汉西
            污水根据上述协议及文件调整了财务报表。本公司收到汉西污水调整后的财务报表，依据《企业会计准
            则第28号—会计政策、会计估计变更和会计差错更正》及《公开发行证券的公司信息披露编报规则第19
            号—财务信息的更正及相关披露》等有关规定，相应调整了2019年度-2021年度及2022年一季度-三季
            度的长期股权投资及投资收益，并追溯调整前期已披露的财务报表。（详见公司2023年3月14日临
            2023-005号公告）（10票同意，0票反对，0票弃权）本公司独立董事张司飞、杨小俊、吴立、廖琨就
            会计差错更正事项出具了独立意见，认为：公司本次会计差错更正符合《企业会计准则第28号—会计
            政策、会计估计变更及差错更正》和《公开发行证券的公司信息披露编报规则第19号—财务信息的更
            正及相关披露》等相关文件的规定，更正后的信息能够更加客观、公允地反映公司的财务状况和经营成
```

4.2 词云

标题词云

正文词云

4.3 解析方法效率比较

因为Selenium的代码修改起来较为容易，所以我选择在Selenium的框架上，进行xpath和BeautifulSoup解析网页的比较，以下是使用了BeautifulSoup方法的代码：

```
from selenium import webdriver
import time
from selenium.webdriver.common.by import By
import datetime
import pandas as pd
from bs4 import BeautifulSoup

starttime = datetime.datetime.now()
data = {'title': [1] * 30, 'date': [2] * 30, 'content': [3] * 30}

df = pd.DataFrame(data)
driver = webdriver.Chrome()
driver.get(
    "http://vip.stock.finance.sina.com.cn/corp/view/vCB_BulletinGather.php?
stock_str=sh600168&page_index=1"
```

```
)
time.sleep(3)  # 加载时间

links = driver.find_elements(
    By.XPATH, "/html/body/div/div[5]/table/tbody/tr/th/a[1]")  # 获取所有a标签组成列表
length = len(links)  # 一共有多少个标签

for i in range(0, length):  # 遍历列表的循环，逐一点击链接
    links = driver.find_elements(
        By.XPATH,
        "/html/body/div/div[5]/table/tbody/tr/th/a[1]")  # 每次循环时都重新获取a标签，组成列表
    link = links[i]  # 逐一将列表里的a标签赋给link
    url = link.get_attribute('href')  # 提取a标签内的链接（type:str）
    driver.get(url)
    time.sleep(1)  # 加载时间

    # 解析网页，第二个参数为解析器
    soup = BeautifulSoup(driver.page_source, "lxml")
    title = soup.find("th", class_="head").get_text()
    print(title)
    df.loc[i, 'title'] = str(title)

    date = soup.find("td", class_="head").get_text()
    print(date)
    df.loc[i, 'date'] = str(date)

    content = soup.find("div", id="content").contents
    # print(text)
    text = str()
    for p in content:
        text = text + "\n" + p.text
    df.loc[i, 'content'] = text

    driver.back()  # 后退，回到原始页面目录页
    time.sleep(1)  # 留出加载时间

print("count:" + str(length))  # 打印列表长度，即有多少篇公告

df.to_csv('./crawler3.csv', index=0, encoding='utf-8')
endtime = datetime.datetime.now()
print("running time:" + str((endtime - starttime).seconds) + "s")
```

爬取时间比较（以爬取一页为例）：

使用xpath：103s左右

```
count:30
running time:103s
```

使用BeautifulSoup：128s左右

```
count:30
running time:128s
```

可见xpath的爬取效率略高于BeautifulSoup，查阅资料知，BeautifulSoup是基于DOM的，会载入整个文档，解析整个DOM树，因此时间和内存开销都会大很多。而lxml只会局部遍历，并且lxml是用c写的，而BeautifulSoup是用python写的，因此性能方面自然会差很多。

五、实验总结

在爬虫的过程中，我遇到过一些问题和困难，比如没有找到正确的xpath、详情页链接的url不正确、分页操作不知道在哪一步进行等等，但通过上网搜索和debug，都很快地找到了解决方法，总体来说较为顺利。

通过这次实验，我收获了:

1. 使用Scrapy和Selenium框架爬取多页面信息
2. 了解了网页的html组成和结构
3. 使用xpath和BeautifulSoup解析网页，感受到了两者在使用上和效率上的差别
4. 学会了如何处理爬取到的数据，包括清洗、过滤、去重、转换数据类型等操作
5. 锻炼了解决问题的能力，如反爬虫机制、网站结构变化等问题的解决方法
6. 掌握了如何利用爬虫和数据分析技术获取和分析互联网上的数据，从而发现信息和商业价值。

总之，在这次实验中我学会了基本的爬虫技能和数据处理的基本操作，是一次难得的经历。