前言

关于Python的可视化我们已经陆陆续续的出了10期，包括饼图、条形图、直方图、箱线图、散点图、折线图、面积图、雷达图、热力图和树地图。如果你跟着我的文章作了一一的学习之后，是不是想动动手来做一个可视化的综合项目呢？不错，如果想的的话，一定要坚持到底哦~

如果你需要绘图的话，先实现这期的数据收集，即如何从豌豆荚的官网中抓取到"网上购物"类APP的所有信息，包括APP对应的下载人数、好评率、评价人数、安装包大小、所属公司等详细信息。关于这一部分的数据抓取，我们仍然使用爬虫利器Python来完成，对于每一个感兴趣的朋友，希望能够详细的安装流程走一遍，既有助于你对Python爬虫的认识，也有助于你知道爬虫的逻辑。

爬虫步骤

- 爬下" 网上购物"类APP下的5种子分类链接

由上图可知，由于" 网上购物"类APP主页下的全部APP只有42页，一共只有1008个APP，如果想获得更多APP的信息，可以将这个大类别拆分到商城、团购、优惠、快递和全球导购5个子分类，发现每个子类别下又有42页（其实网站的数据只能到41页），这样不就可以获得更多APP信息了嘛~~所以，第一步需要做的上就是如何将5个子分类的链接获取到，然后基于子分类生成规律的url（即含page的url）。

```
# ========== Python3.X Jupyter ==========
# ========== 步骤一、抓取每一个子分类的URL ==========

# 导入第三方包
import requests
from bs4 import BeautifulSoup
import numpy as np
import time
import pandas as pd

# 设置请求头
headers = {'User-Agent':'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/61.0.3163.79 Safari/537.36'}

# 豌豆荚应用首页 > 安卓软件分类 > 网上购物 > 商城下载的链接
url = 'http://www.wandoujia.com/category/5017_591'
# 发送请求
res = requests.get(url, headers = headers).text
# 解析HTML
soup = BeautifulSoup(res, 'html.parser')

# 商城类app的5个分类链接及名称
category_urls = [i.findAll('a')[0]['href'] for i in soup.findAll('ul',{'class':'switch-tab cate-tab'})
[0].findAll('li')[1:]]
category_names = [i.text.strip() for i in soup.findAll('ul',{'class':'switch-tab cate-tab'})[0].findAll('li')
[1:]]
```

- 生成规律的urls

你会发现，在某个子分类下，虽然都有42页，但你去点击第2页、第3页...时，你会发现这些子分类下的页数链接是存在规律的，即：

http://www.wandoujia.com/category/5017_591/1

http://www.wandoujia.com/category/5017_591/2

这就是"商城"类APP的前两页URL。所以只需要对每一个类别APP链接的url后跟上这些page数值，就可以得到所有子分类下的所有页数的链接。如果把每个分类下的40页所有APP都遍历的抓一次，会比较耗费时间，故这里就暂时针对每个分类的前10页APP数据进行遍历，生成如下的链接。

```
# ========== 步骤二、生成所有子分类及页码的URL ==========

# 各类别app的前10页urls
```

```
names = []
urls = []

for url,name in zip(category_urls,category_names):
    for i in range(1,11):
        names.append(name)
        urls.append(url+'/'+str(i))
```

- 根据规律的url抓取出APP的名称及其超链接

由于我们需要的详细信息，是通过点击每一个APP之后才能得到的，所以首先需要获得这些通过点击APP之后的超链接，而这些超链接正如下图所示，存放在HTML的h2标签下：

```
# ========== 步骤三、抓取子分类页下各APP对应的URL ==========

# 根据每一页的url抓出app对应的链接
app_urls = []

for url in urls:
    res = requests.get(url, headers = headers).text
    soup = BeautifulSoup(res,'html.parser')

    # 返回每个页面中app的名称及对应的链接
    # 为防止报错，这里做了异常处理
    try:
        app_lists = soup.findAll('ul',{'id':'j-tag-list'})[0]
        app_urls.extend([i.findAll('a')[0]['href'] for i in app_lists.findAll('h2',{'class':'app-title-h2'})])
    except:
        pass
```

第二步的目的就是找到每一个APP背后的超链接是什么，故根据上面的代码，可以得到1204个APP对应的超链接。当点击子分类页下的APP后，就来到了APP的详情页，这些信息都规律的存放在一些标记底下，如APP名称在p标记下、APP分类在dl标记下等。那我们就可以这些监控出来的标记进行数据的抓取了：

```
# ========== 步骤四、爬虫抓取各APP的详细信息 ==========

# 构建空的列表，用于数据的存储
appname = []
appcategory = []
install = []
love = []
comments = []
size = []
update = []
version = []
platform = []
company = []

for url in app_urls:
    res = requests.get(url, headers = headers).text
    soup = BeautifulSoup(res,'html.parser')

    try:
        # 抓取的信息
        appname.append(soup.find('p',{'class':'app-name'}).text.strip())
        appcategory.append('-'.join(soup.find('dl',{'class':'infos-list'}).findAll('dd')
[1].text.strip().split('\n')))
        install.append(soup.find('span',{'class':'item install'}).find('i').text)
        love.append(soup.find('span',{'class':'item love'}).find('i').text)
        comments.append(soup.find('div',{'class':'comment-area'}).find('i').text)
        size.append(soup.find('dl',{'class':'infos-list'}).findAll('dd')[0].text.strip())
        update.append(soup.find('dl',{'class':'infos-list'}).findAll('dd')[3].text.strip())
        version.append(soup.find('dl',{'class':'infos-list'}).findAll('dd')[4].text.strip())
        platform.append(soup.find('dl',{'class':'infos-list'}).findAll('dd')[5].text.strip().split('\n')[0])
        company.append(soup.find('dl',{'class':'infos-list'}).findAll('dd')[6].text.strip())
```

```
        except:
            pass
```

- 根据以上的url抓取出APP的详细信息

上面的一段代码会耗费一些时间，会根据不同的电脑有所差异，我的电脑跑了约20分钟。虽然等待了一些时间，但当结果正常的存储后，心里还是有一些兴奋的。接下来就是要将这些列表的存储结果输出到本地的Excel表格中，这样便于后期的分析使用。

```
# ========== 步骤五、数据存储 ==========

# 将存储的列表值写入到字典，并进行数据框的转换
apps = pd.DataFrame({'appname':appname,'appcategory':appcategory,
                     'install':install,'love':love,'comments':comments,'size':size,
                     'update':update,'version':version,'platform':platform,'company':company})

# 数据导出
apps.to_excel('apps.xlsx', index = False)
```

回到你Python的工作空间，打开apps的Excel文件，如果上面你的运行没有任何问题的话，存储的数据将会是如下的结果：

结语

OK，这期关于豌豆荚APP应用的爬虫案例就讲到这里，下一期我们将基于这个数据进行数据的可视化分析。如果你有问题，欢迎在公众号的留言区域表达你的疑问。同时，也欢迎各位朋友继续转发与分享文中的内容，让跟多的人学习和操作。最后，本文相关的Python脚本和PDF版本已存放到百度云盘，可以通过下面的链接获取：

链接: https://pan.baidu.com/s/1geDRddT 密码: cud2