基于 Python3 爬虫获取最新上架图书的实现

吴剑冰
（浙江海洋大学石化与能源工程学院，浙江 舟山 316000）

摘　要：在大数据时代，利用网络爬虫自动定向采集多网页有用的信息，并将爬取信息储存至数据库，Excel，Word 等，可以根据网页历史数据来确定爬虫爬取网页更新信息的周期，大大增加了信息的更新及时性。基于 Python3.5 定向爬取当当网最新上架图书，存储图书基本信息到 Word 文档，并且可对比历史爬取记录，更新新书信息。

关键词：Python3 语言；网络爬虫；定向爬虫；主题爬虫

DOI:10.16184/j.cnki.comprg.2018.04.011

1　概述

主题网络爬虫，根据一定的网页分析算法过滤与主题无关的链接保留主题相关的链接并将其放入待抓取的 URL 队列；然后根据一定的搜索策略从队列中选择下一步要抓取的 URL，并重复上述过程，直到达到某一条件时停止[1]。如果采取手动从互联网中获取多网页信息，耗费人力和精力过大，而网络爬虫就能很好地弥补这个缺陷。同时利用网络爬虫可以过滤掉很多手动采集时遇到的无关数据，采集数据更加便捷。伴随着爬虫技术也产生了众多反爬技术，在爬取时候应注意“礼貌”爬行[3]，否则会被对方限制爬行。

Python 语言代码简洁，易学，并且拥有多种发送 HTTP 请求和解析 HTML 源码的库，故该语言十分适合用来编写爬虫程序。基于 Python3.5 爬取当当网最新上架图书，获取图书的基本信息并将获取的信息以 Word 文档的形式展现。并且在二次爬取图书信息时可以对比前一次爬取记录，将更新的图书信息存储而不重复存储已经爬取过的图书信息，方便购书者直观地了解最新上架的图书信息。

2　爬虫的设计

2.1　编程环境

Window10 操作系统，Python3.5，IDE:PyCharm。

2.2　爬取的起始页面

爬取的新书上架包括小说、文学、烹饪、经济、管理等分类栏目，小说栏目的起始采集页面为 http://book.dangdang.com/list/newRelease_C01.03_P1.html，观察各栏目起始页面 URL 规律，可构造 urlStart='http://book.dangdang.com/list/newRelease_C01.{}_P1.html'.format(v)，每个分类栏目对应相应的变量 v 值。

2.3　模拟浏览器访问

一些站点不喜欢非人为访问，因此应该尽量让爬虫模拟人真实访问的样子，网站是通过浏览器发送请求头中的 User-Agent 的值来判断浏览器身份，把 User-Agent 修改成对应的浏览器。当把模拟的访问方式改成移动设备中的浏览器时，则会得到一个没有广告以及其他干扰的简化的更易爬虫采集的网站版本[2]。利用 requests 库发送 GET 请求，返回网页信息，若无法得到页面信息利用 try-except 则会显示“页面获取出错”。

```
def getResponseContent(self, urls):
    try:
                                headers = {"User-Agent":"
Mozilla/5.0 (Windows NT10.0;WOW64)AppleWebKit/
537.36 (KHTML,like Gecko) Chrome/61.0.3163.91 Sa-
fari/537.36"}
      html = requests.get(urls, headers=headers)
    return html.text
    except:
      print("页面获取出错")
```

2.4　获取总页面数

新书上架栏目的需要翻页爬取，利用 BeautifulSoup 库来解析返回的网页信息，用 Chrome 的开发者工具找到最后一个页面对应的标签 find（'div'，{"class" :" fanye_bottom"}）中的‘a’标签。观察翻页后的页面 URL 比较可得第一页为 P1，第二页为 P2，以此类推。可构造出每一页 的 URL，遍历每个页面即可爬取信息。

作者简介：吴剑冰（1997-），女，本科。

收稿日期：2018-01-29

2.5 抓取图书基本信息

2.5.1 图书封面及书名采集

需要抓取的图书信息包括：书名、作者、价格、图片、内容简介。其中书名和图片信息在主页面中获取，查看浏览器开发者工具，图书封面URL地址和书名分别在<img>标签内src属性和title属性利用BeautifulSoup库和正则表达式来获取。正则表达式" /([1-9] [0-9]{4,} -1_p.jpg) ((?! /?) .) *$" 可以从页面的<img>标签中匹配出的为封面缩略图，一般原图和缩略图之间URL有一定规则，知道缩略图的URL就能构造出原图的URL，这里原图的URL正则表达式为" http.*/[1-9][0-9]{4,} -1_w.jpg"。在Python 3以后的版本中，urllib2这个模块已经不存在，合并到urllib中，分为urllib.request和urllib.error 。利用urllib.request.urlopen模块解析图片地址，保存图片。

```
htmlContent = self.getResponseContent(url)
bsObj = BeautifulSoup(htmlContent, "lxml")
for img_link in bsObj.findAll("img",src=re.compile
            ("/([1-9][0-9]{4,}-1_p.jpg)((?! /?).)*$")):
  if img_link.attrs[´src´] not in item.imgs:
    ig = img_link.attrs[´src´]
    item.imgs.append(ig)
    imgurls=re.findall("http.*/[1-9][0-9]{4,}-1_",ig)
    for imgurl in imgurls:
        img_add = imgurl + ´w.jpg´
        item.imgList.append(img_add)
    itle_link = img_link.attrs[´title´]
    if title_link not in item.titles:
      item.titles.append(title_link)
```

2.5.2 获取图书详情页面内链

因在主页图书价格不会根据打折情况发生变动，内容简介也不完整，故需要在图书的详情页面获取，在主页面通过标签定位和正则表达式的配合使用寻找每个商品详情的子页面，在class类NewbookItem中创建列表newpage = []，将每个获取的子页面链接加入列表newpage，将新获取的子页面链接与列表中已有的进行比对，可以避免重复采集。

```
for url in urls:
   htmlContent=self.getResponseContent(url)
   bsObj = BeautifulSoup(htmlContent,"lxml")
for link in bsObj.findAll("a", href=re.compile("/([1-9][0-
9]{4,}.html)((?! /?).)*$")):
if ´href´ in link.attrs:
  if link.attrs[´href´] not in item.newpage:
    item = NewbookItem()
    # 遇到了新页面
    newPage = link.attrs[´href´]
    item.newpage.append(newPage)
```

2.5.3 获取异步加载信息

图书价格和内容简介都是异步加载，不同于爬取普通页面，用普通的requests发出的get请求返回的网页信息内不包含的数据就是异步加载，是一种相对于同步加载的“延迟”的加载方式。一般是由页面滚动条下拉，或者点击触发的加载方式，通过Ajax发出请求，接受JSON格式的字符串。Python爬虫用json库解析请求的url返回了一个字典类型数据，通过浏览器开发者工具查看详细的Ajax请求信息，找到爬取数据对应的键，通过键来索引数据，图书内容简介包含是在返回字典数据中的字典‘data’中键‘html’对应的值。通过正则表达式清洗掉数据中多余的符号，匹配到包含“内容简介”的段落。图书价格则包含在返回数据中字典‘price’中键‘salePrice’对应的数字，若商品打折则为字典‘price’中键‘directPrice’对应的数字，可以使用try—except先获取打折价格若无获取到价格则获取图书正常价格。图书作者信息利用普通获取方式resquests库和BeautifulSope库在" a" 标签中找到 " dd_name" : " 作者"。实验发现中因个别图书原网页并无作者信息，故采取异常处理（try—expect）采集图书详情中若发生异常则输出异常并进行下一本图书的采集，防止爬虫意外中断采集，需重新开始采集的麻烦。

```
pt = "[1-9][0-9]{4,}"
id1 = re.findall(pt, newPage)
for iddd in id1:
        get_url = "http://product.dangdang.com/index.
php?r=callback%2Fdetail&productId={}&templateType
=publish&describeMap =&shopId =0&categoryPath =
01.03.45.00.00.00".format(iddd)
content_data = requests.get(get_url)
requests.encoding = ´gb18030´
json_data = json.loads(content_data.text)
content1 = json_data[´data´]
content2 = content1[´html´]
contents_all = re.sub ("[A-Za-z0-9\\! %[\]\,=/<>:...\"
&_; ()\–\´]", "", content2)# 清洗数据
```

```
contents = re.findall("内容简介.*", contents_all)
```

2.6 图书信息的储存

2.6.1 基本信息写入 Word 文档

引入 Python 的 docx 库，将获取的信息写入 Word 文档方便查看。设置变量 i，每爬取一本书自增 1，创建类 class NewbookItem: titles = [] ,author = [] ,price = [] , content= [] ,newpage = []。爬取每本书的对应信息加入对应列表，存储时按次序提取对应列表的信息。

```
from docx import Document
from docx.shared import Inches
if (i == 1):
    document = Document()  # 创建空 Word 文档
else:    # 追加写入“新书上架.dox”文档
    document = Document("新书上架.docx")
# 文档标题
document.add_heading("新书上架", 0)
# 文档段落
p=document.add_paragraph("第{}本书".format(i))
p=document.add_paragraph("网址:"+item.newpag[i-1])
p=document.add_paragraph("书名:"+item.titles[i-1])
p=document.add_paragraph("作者:"+item.author[i-1])
p=document.add_paragraph("价格:"+item.price[i-1])
document.add_picture(img_name,width=Inches(2.5))
try:
    p=document.add_paragraph(item.content[0])
except:
    p=document.add_paragraph("暂时没有获取到内容")
item.content.clear()
document.add_page_break()# 插入空页面
document.save("新书上架.docx") # 保存文档
```

2.6.2 图书书名写入 Excel

Python 没有直接操作 Excel 的模块，从第三方库引入 xlwt 模块。xlwt 模块作用是将数据写入 Excel 中。因是 xlwt 以只读方式打开，后再写入数据会覆盖原始保存数据。应引入 os.path 模块打开文件，xlrd.open_workbook 打开已有的 xsl 文件，引入 xlutils 用来修改 Excel 文档，复制文件内容循环追加写入每次抓取信息。

```
import xlwt
from os.path import join
from xlrd import open_workbook
from xlutils.copy import copy
if(i==1):
  workbook=xlwt.Workbook(encoding='ascii')
  # 新建工作表
  worksheet=workbook.add_sheet("my worksheet")
  worksheet.write(0, 0, ' 书名 ')
  worksheet.write(i, 0, item.titles[i-1])
  workbook.save("1.xls")
else:
  rb = open_workbook(join('1.xls'))
  # 复制工作簿内容并追加写入
  wb = copy(rb)
  wb.get_sheet(0).write(i,0,item.titles[i-1])
  wb.save(join('1.xls'))
```

2.7 图书信息的历史比对

打开前一次储存的的工作簿，xlrd 模块读取内容若书本已经被采集过，则不会执行存储书本模块。

```
data = xlrd.open_workbook('1.xls')
table = data.sheets()[0]
table.col_values(0)
print(item.titles[i - 1])t
if (item.titles[i - 1] in table.col_values(0)):
    print("本书已经存在")
    item.content.clear()
else:
    print("本书不存在")
    self.pipelines()
```

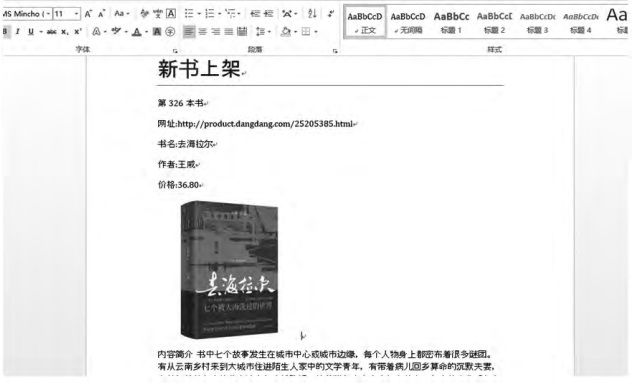

图 1

3 结语

通过网络定向爬虫实现了对最新上架图书信息的爬取，实验表明本程序可以有效实现定向爬取功能，并将爬取数据保存，以 Word 文档形式直观展现，其中爬取方法对爬虫初学者有借鉴意义。因爬取的图书的基本信息较多，爬取速度有待加快，可以使用多线程、多进程的方式加快爬取速度。但也应在不对爬取网站后台正常运行造成影响的情况下，适量地进行爬取。随着大数据

(下转第 39 页)

表 2　各个特征的性能测试结果

特征	正确率	累计收益率	夏普比率
收盘价	51.05%	47.97%	1.1
20 日均价	59.79%	59.79%	1.34
MACD	55.56%	35.19%	0.83
ATR	71.63%	47.03%	1.4
BR	51.58%	37.01%	1.24
MFI	44.54%	18.96%	0.95

综上，选取 20 日均价作为特征，进行后续的实验。

3.3　训练与验证模型

将数据集分为 2 个部分，选取 2000 年到 2002 年的数据为训练集，选取 2002 年到 2004 年为测试集，并通过特征向量利用隐马尔可夫模型构建预测涨跌模型，其预测结果如图 3 所示。

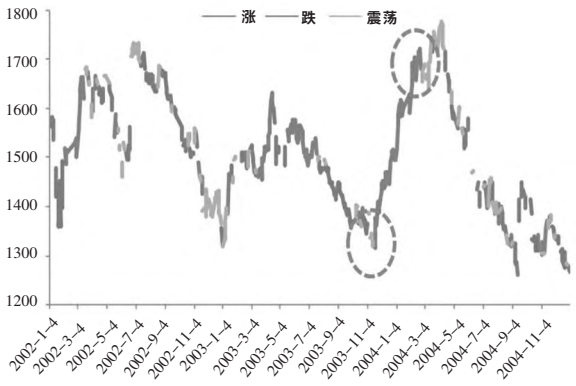

图 3　预测结果

从图 3 中，可以发现本文的马尔可夫模型在预测股票价格走势时，具有很高的准确，其准确率达到 75.11%。其也存在一些缺点，在预测股票市场价格波动较小的震荡行情时，很容易出现误判，这也会是后续研究的重点。

4　结语

隐马尔可夫模型在一定程度识别市场的状态具有很高的准确性。运用真实数据实验，建立了一个基于马尔可夫模型的量化择时系统，并证明该方法的可行性。

对于后续的研究，将引入更多的智能动态选择机制来提高模型在整体准确率。同时也在考虑融入其他模型建立混合模型。

参考文献

[1] Guidolin, M. and A. Timmermann. (2007) Asset allocation under multivariate regime switching. Journal of Economic Dynamics & Control, 31 (11) , 3503–3544.

[2] De Angelis, L. and L.J. Paas. (2013) A dynamic analysis of stock markets using a hidden Markov model. Journal of Applied Statistics, 40 (8) , 1682–1700.

[3] Salhi, K., et al. (2016) Regime switching model for financial data: Empirical risk analysis. Physica aStatistical Mechanics and Its Applications, 461, 148–157.

[4] 丁鹏. 量化投资——策略与技术 [M] . 北京: 电子工业出版社, 2004.

[5] 曾劲松. 技术分析与中国股票市场有效性 [J] . 财经问题研究, 2005, (08) : 27–30.

(上接第 33 页)

时代的来临，未来爬虫技术的应用将会越来越广泛。

参考文献

[1] 刘金红，陆余良. 主题网络爬虫研究综述 [J] . 计算机应用研究, 2007, 24 (10) : 26–29+47.

[2] <美> Ryan Mitchell. Python 网络数据采集. 北京：人民邮电出版社, 2016.

[3] 周德懋，李舟军. 高性能网络爬虫：研究综述. 计算机科学, 2009, 36 (8) : 26–29+53.

(上接第 36 页)

取分析 [D] . 吉林大学, 2017.

[3] 刘金红，陆余良. 主题网络爬虫研究综述 [J] . 计算机应用研究, 2007, (10) : 26–29+47.

[4] 李文龙. 基于 Docker 集群的分布式爬虫研究与设计 [D] . 浙江理工大学, 2017.

[5] 蔡学锋. 基于 Solr 的搜索引擎核心技术研究与应用 [D] . 武汉理工大学, 2013.