动态网页爬取及信息处理模型

（设计）

对动态网页网络爬虫的研究

——以爬取京东商品信息为例

作　　者：姚　　圆

学　　院：信息工程学院

专　　业：数据科学与大数据技术

年　　级：2020

学　　号：2020145128

任课教师：李　京　建

课程名称：爬虫技术和大数据分析

完成日期：2023.06.29

目录

摘要

大数据是互联网发展到一定阶段的必然产物。由于 Internet 整合资源的能力在不断提高，因此 Internet 本身必须通过数据反映其自身的价值，因此，从这个角度来看，数据是 Internet 的价值体现。通过 Python 网络爬虫技术，能在短时间内获取大量可靠信息，结合数据挖掘模型，将互联网中的隐藏价值挖掘，对趋势分析、事务决策、价值评估等提供参考。

关键词：大数据　网络爬虫　数据挖掘　Python

Abstract

Big data is an inevitable product of the development of the Internet to a certain stage. Since the Internet's ability to integrate resources is constantly improving, the Internet itself must reflect its own value through data, so from this point of view, data is the value of the Internet. Through Python web crawler technology, a large amount of reliable information can be obtained in a short time, combined with data mining models, the hidden value mining in the Internet provides reference for trend analysis, transaction decision-making, value evaluation, etc.

Key Words: Big Data　Web Crawlers　Data Mining　Python

引言

网络数据采集是指通过网络爬虫或网站公开 API 等方式从网站上获取数据信息。该方法可以将非结构化数据从网页中抽取出来，将其存储为统一的本地数据文件，并以结构化的方式存储。它支持图片、音频、视频等文件或附件的采集，附件与正文可以自动关联。

在互联网时代，网络爬虫主要是为搜索引擎提供最全面和最新的数据。面对大数据时代的海量数据，网络爬虫更是从互联网上采集数据的有利工具。

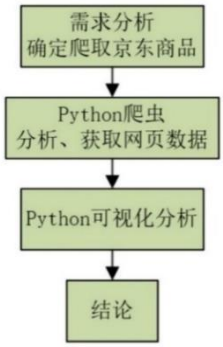
可视化数据挖掘技术通过可视化的方式收集和整理数据,在收集过程中结合规律找出隐含的关系和技术。可视化技术主要通过挖掘过程中获取最终信息。数据挖掘过程中，可视化技术主要体现在可以通过图形的方式表达，人们可以直观看到具体情况。这些图形可以展示数据的来源、数据挖掘过程以及人们想要的结果。当用户需要某些数据时，可以通过可视化的方式进行观察，获取所需信息。

第一章、 业务分析

1.1 业务需求

1.1.1 需求简述

随着电子商务的快速发展，电商网站迅速增加，随之而来网络购物成为趋势。网络购物带给商家和消费者的一个挑战是信息过载问题。面对电商网站上的大量数据，商家和消费者会迷茫，不能做出理性选择。为解决此问题，网络爬取，又称为数据自动采集，被提出来并得到迅速发展。因发布在电商网站上的数据被认为是公开的，所以对其上大量网页信息的爬取是允许的。因此，爬虫在这个信息爆炸的大数据时代是一项必不可少的技能。



图表 1 程序流程图

1.1.2 主要难题

1.1.2.1 页面“懒加载”

页面懒加载减轻了网站服务器的负载压力，但同时也使得爬虫脚本程序的爬取难度增加，为此采用 json 语言脚本，模拟页面滚动，实现对数据的全量采集。

1.1.2.2 页面事件触发

很多商品售卖网站为提供顾客更好使用体验，通常会采用商品信息模块化展示，因此，往往某件商品信息采集过程中需要定位到对应标签，触发点击事件，完成页面跳转。

1.1.2.3 词云绘制

词云图的绘制过程中，涉及自然语言处理和可视化分析，其中词频统计阶段，往往包含了大量语气词和商品品质关联性较弱词汇，在这部分中选用合适的停用词表，语言逻辑分析能提升词云图中关键词可信度。

1.1.2.4 商品选购参考

在此部分，本文所选用的是获取商品好评率来对比商品的购买权重，此外还可以以好评数量、电脑配置、价格等多方面入手。

1.2 功能实现

1.2.1 模块选用

本文所选用 python 语言中的模块如下：

```
import re
# 网页解析
from lxml import html
etree = html.etree
# 异步操作
import asyncio
# 网页请求及交互模块
from pyppeteer import launch
# 时间获取
import time
# 数据打印
from pprint import pprint
# 中文分词库
import jieba
# 元素对象频率统计
import collections
# 词云绘制模块
from wordcloud import WordCloud
# python 绘画模块
import matplotlib.pyplot as plt
```

1.2.2 功能实现

本文实现了以下功能：

1.2.2.1 爬取机制设定和爬虫伪装

在爬虫脚本程序爬取网页信息时，目标网站为及时响应爬虫脚本程序可能会引起资源占用，服务器负载加重，因此很多网站会采取反爬虫措施。为实现爬虫脚本程序信息采集工作，设置合理的爬取机制，并通过参数设定伪装爬虫，能极大促进爬虫脚本程序工作效率和网络资源合理化使用。

1.2.2.2 页面滚动

动态网页基本功能是通过 json 语言实现，在爬虫脚本程序中注入 js 语言，能够实现爬虫脚本程序和网页之间的交互，对目标信息进行全量采集。

1.2.2.3 页面跳转

网页结构经常是模块化、分区化以及合理化制定，而在信息爬取进程中，对某些指定信息的爬取需要通过网页元素定位，事件触发来完成页面跳转，以达成对指定网页信息爬取。

1.2.2.4 信息采集

本文所采用 CSS 网页结构定位，并通过 json 语法对网页元素中的信息提取、存储。

1.2.2.5 页面自动化及爬取机制设置

Asyncio 为 python 语言程序中可实现异步操作的函数库，通过 asyncio 函数的运用，完成对页面的异步操作，保证了页面信息采集流程中流向的稳定。

页面访问或页面跳转过程中往往会出现连接不稳定或反爬虫机制的监视，本文通过等待响应参数设置，即保障了页面请求的间歇性，也减少了网页资源占用，避免了反爬虫机制的限制。

1.2.2.6 数据可视化处理

本文为了使用户能够瞬间理解弹幕的关键信息,利用越高频越突出的视觉效果,将分析出来的高频词汇用词云图进行展示。。频率越高的词汇,其尺寸越大,反之,越小。

第二章、 网络爬虫设计

2. 1 网站结构和机制解析

2.1.1 网站构成

网站构成：域名+页面

域名一般是不会改变的，因此我们爬虫所需要解析的就是网站自己所编写的不同页面的入口 url，只有解析出来各个页面的入口，我们才能开始我们的爬虫。

2.1.2 网页加载方法

同步加载：改变网址上的某些参数会导致网页发生改变

异步加载：改变网址上的参数不会使网页发生改变

2.1.3 网页源码

源代码一般由三个部分组成，分别是：

html：描述网页的内容结构

css：描述网页的排版布局

JavaScript：描述网页的事件处理，即鼠标或键盘在网页元素上的动作后的程序

2.2 网页请求

2.2.1 请求头（Request Headers）信息详解：

```
1.  Accept: text/html,image/*(浏览器可以接收的类型)
2.  Accept-Charset: ISO-8859-1(浏览器可以接收的编码类型)
3.  Accept-Encoding: gzip,compress(浏览器可以接收压缩编码类型)
4.  Accept-Language: en-us,zh-cn(浏览器可以接收的语言和国家类型)
5.  Host: www.it315.org:80(浏览器请求的主机和端口)
6.  If-Modified-Since: Tue, 11 Jul 2000 18:23:51 GMT(某个页面缓存时间)
7.  Referer: http://www.it315.org/index.jsp(请求来自于哪个页面)
8.  User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)(浏览器相关信息)
9.  Cookie：(浏览器暂存服务器发送的信息)
10. Connection: close(1.0)/Keep-Alive(1.1)(HTTP 请求的版本的特点)
11. Date: Tue, 11 Jul 2000 18:23:51 GMT(请求网站的时间)
```

2.2.2 响应头（Response Headers）信息详解：

```
1.  Location: http://www.it315.org/index.jsp(控制浏览器显示哪个页面)
2.  Server:apache tomcat(服务器的类型)
3.  Content-Encoding: gzip(服务器发送的压缩编码方式)
4.  Content-Length: 80(服务器发送显示的字节码长度)
5.  Content-Language: zh-cn(服务器发送内容的语言和国家名)
6.  Content-Type: image/jpeg; charset=UTF-8(服务器发送内容的类型和编码类型)
7.  Last-Modified: Tue, 11 Jul 2000 18:23:51 GMT(服务器最后一次修改的时间)
8.  Refresh: 1;url=http://www.it315.org(控制浏览器 1 秒钟后转发 URL 所指向的页面)
9.  Content-Disposition: attachment; filename=aaa.jpg(服务器控制浏览器发下载方式打开文件)
10. Transfer-Encoding: chunked(服务器分块传递数据到客户端)
11. Set-Cookie:SS=Q0=5Lb_nQ; path=/search(服务器发送 Cookie 相关的信息)
12. Expires: -1(服务器控制浏览器不要缓存网页，默认是缓存)
13. Cache-Control: no-cache(服务器控制浏览器不要缓存网页)
```

```
Pragma: no-cache(服务器控制浏览器不要缓存网页)
Connection: close/Keep-Alive(HTTP 请求的版本的特点)
Date: Tue, 11 Jul 2000 18:23:51 GMT(响应网站的时间)
```

2.2.3 网页请求过程

从浏览器输入网址、回车后，到用户看到网页内容，经过的步骤如下：

```
（1）dns 解析，获取 ip 地址；
（2）建立 TCP 连接，3 次握手；
（3）发送 HTTP 请求报文；
（4）服务器接收请求并作处理；
（5）服务器发送 HTTP 响应报文；
（6）断开 TCP 连接，4 次挥手。
```

2.3 环境准备

Anaconda 安装， Python 环境配置&Pycharm 安装；

pip 安装，pip 是 Python 的包管理器，现在的 Python 安装包一般都会自带 pip。

第三方库：

Re、asynio、pyppeter、jiaba、wordcloud 库的安装

2.4 网络爬虫

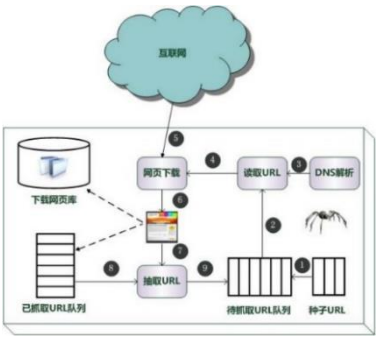
2.4.1 网页爬虫介绍

网络爬虫（又称为网页蜘蛛，网络机器人，在 FOAF 社区中间，更经常的称为网页追逐者），是一种按照一定的规则，自动地抓取万维网信息的程序或者脚本。

网络爬虫 robots 协议是一个业内的约定，虽然不具有法律意义的，但是里面写明了网站里面哪些内容可以抓取，哪些不允许，我们爬取网页时应遵循网络协议规范。

网络爬虫按照系统结构和实现技术，大致可以分为以下几种类型：通用网络爬虫、聚焦网络爬虫、增量式网络爬虫、深层网络爬虫，实际的网络爬虫系统通

常是多种爬虫技术相结合实现的。



图表 2 网络爬虫原理

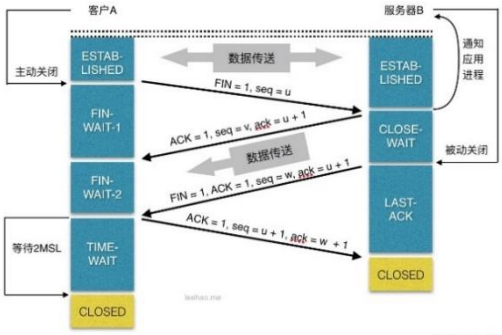
2.4.2 域名解析

统一资源定位符 URL 通过域名解析后返回对应 IP 地址，我们可以通过该 IP 地址访问网络资源。

2.4.3 建立连接

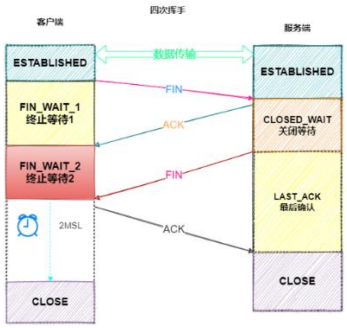
页面跳转、事件触发、与服务器建立连接等过程中， 为保障请求数据的正确返回，我们先要了解建立连接，并在代码模块中加入等待机制。

建立 TCP 连接，达成三次握手协议，建立数据传输基础。



图表 3 “三次握手”

当数据传送完毕，断开 TCP 连接。那对于 TCP 的断开连接，这里就有了神秘的“四次分手”。



图表 4 “四次挥手”

2.4.4 HTTP 协议

爬虫脚本程序通自定义请求头和在请求头 js 注入防止被页面认为自动化执行脚本的爬虫软件等多种方式进行伪装，对网页数据爬取。

TCP/IP： Http 使用 TCP 作为它的支撑运输协议。HTTP 客户机发起一个与服务器的 TCP 连接，一旦连接建立，浏览器（客户机）和服务器进程就可以通过套接字接口访问 TCP。

HTTP 请求报文：一个 HTTP 请求报文由请求行（request line）、请求头部（header）、空行和请求数据 4 个部分组成，下图给出了请求报文的一般格式。

图表 5 HTTP 请求头

2.4.5 请求行

在该部分，只讨论请求方式的选择，而 pyppeter 库是通过操控浏览器行为进行爬取，请求方式与调用浏览器有关。

请求行分为三个部分：请求方法、请求地址和协议版本。

请求方法：

HTTP/1.1 定义的请求方法有 8 种：GET、POST、PUT、DELETE、PATCH、HEAD、OPTIONS、TRACE。最常的两种 GET 和 POST，如果是 RESTful 接口的话一般会用到 GET、POST、DELETE、PUT。

2.5 代码实现

2.5.1 导入第三方库

```
import re
# 网页解析
from lxml import html
etree = html.etree
# 异步操作
import asyncio
# 网页请求及交互模块
from pyppeteer import launch
# 时间获取
```

```
import time
# 数据打印
from pprint import pprint
# 中文分词库
import jieba
# 元素对象频率统计
import collections
# 词云绘制模块
from wordcloud import WordCloud
# python 绘画模块
import matplotlib.pyplot as plt
```

2.5.2 网页请求对象

```
async def switch(pages, start_url,Info):
    start_time = time.time()
    #'headless': False 如果想要浏览器隐藏更改 False 为 True
    browser = await launch(options={
        'headless': False, # 设置无头浏览器
        'userDataDir': r'D:\temporary', # 自定义用户目录
        'args': ['--no-sandbox', # 无沙盒模式运行
        '--window-size=1440,900',       # 自定义屏占比
        '--disable-infobars'],    # 禁用浏览器正在被自动化的提示
        'dumpio': True,     # 限制内存使用，防止页面卡住
        'fullPage': True, # 展开页面
        'waitUntil': 'networkidle2'# 网络资源等待机制})
    page = await browser.newPage()# 创建一个新的浏览页面
    await page.setJavaScriptEnabled(enabled=True) # 开启 js
    await page.setViewport({'width':1440,'height':900})# 渲染屏占比
     # 设置请求头伪装爬虫
    await page.setUserAgent(
        'Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.67 Safari/537.36')
    # js 注入防止被页面让位自动化执行脚本的爬虫软件
    await page.evaluate('''() =>{ Object.defineProperties(navigator,{ webdriver:{ get: () => false } }) }''')
    await page.goto(start_url,options={'waitUntil': 'networkidle2','fullPage':True})  # 前往目标网站
    await page.waitFor(2500); # 等待机制
```

2.5.3 网页中转

在此部分至关重要是实现网页滑动的模拟行为，解决网页“懒加载”，方可获得正确商品详情页 url。

```
    # 滑动 js  动态加载
        await page.evaluate('''''async () => {
                            await new
                        Promise((resolve, reject) => {
                            var
                        totalHeight = 0;
                        var
                        distance = 100;
                        var
                        timer = setInterval(() => {
                            var
                        scrollHeight = document.body.scrollHeight;
                        window.scrollBy(0, distance);
                        totalHeight += distance;

                        if (totalHeight >= scrollHeight){
                        clearInterval(timer);
                        resolve();
                        }
                        }, 100);
                        });
                }''')
        await asyncio.sleep(3) # 设置等待加载，防止页面加载失败
        # 根据页数轮转爬取第 61-i 个商品
        if pages%2:
            num = 36
        else:
            num = 25
        # 返回目标商品 url
        link = await page.Jeval('#J_goodsList > ul > li:nth-
child({}) > div > div.p-img > a'.format(num),'el => el.href')
        # 跳转函数，前往商品详情页
        await page.goto(link)
        await page.waitFor(2500)
```

2.5.4 获取商品信息

请求函数所得网页源码解析后，通过 css 或 xpath 定位，存储在字典后转储在一个列表，最后数据处理模块将所得数据存储在一个列表。

```
    Price = await page.Jeval('.itemInfo-wrap .summary-price-wrap .summary-price .p-price .price','el => el.innerText')
        Name = await page.Jeval('#detail > div.tab-con > div:nth-child(1) > div.p-parameter > ul.parameter2.p-parameter-list > li:nth-child(1)','el => el.title')
        wt = await page.Jeval('#detail > div.tab-con > div:nth-child(1) > div.p-parameter > ul.parameter2.p-parameter-list > li:nth-child(3)','el => el.title')
        Weight = re.sub('kg','',wt) # 正则化匹配
        Screen_color_gamut = await page.Jeval('#detail > div.tab-con > div:nth-child(1) > div.p-parameter > ul.parameter2.p-parameter-list > li:nth-child(5)','el => el.title')
        Type = await page.Jeval('#detail > div.tab-con > div:nth-child(1) > div.p-parameter > ul.parameter2.p-parameter-list > li:nth-child(6)','el => el.title')
        Thickness = await page.Jeval('#detail > div.tab-con > div:nth-child(1) > div.p-parameter > ul.parameter2.p-parameter-list > li:nth-child(8)','el => el.title')
        it3 = await page.Jx('//div[@data-tab="item"]/div[1]/ul[2]/li[contains(string(),"内存容量")]')
        for i in it3:
            Memory=await (await i.getProperty('title')).jsonValue()
        it4 = await page.Jx('//div[@data-tab="item"]/div[1]/ul[2]/li[contains(string(),"支持 IPv6")]')
        for i in it4:
            IP=await (await i.getProperty('title')).jsonValue()
        it5 = await page.Jx('//div[@data-tab="item"]/div[1]/ul[2]/li[contains(string(),"颜色")]')
        for i in it5:
            Color=await (await i.getProperty('title')).jsonValue()
        Processor = await page.Jeval('#detail > div.tab-con > div:nth-child(1) > div.p-parameter > ul.parameter2.p-parameter-list > li:nth-child(19)','el => el.title')
        Screen_refresh_rate = await page.Jeval('#detail > div.tab-con > div:nth-child(1) > div.p-parameter > ul.parameter2.p-parameter-list > li:nth-child(17)','el => el.title')
        Graphics_card_model = await page.Jeval('#detail > div.tab-con > div:nth-child(1) > div.p-parameter > ul.parameter2.p-parameter-list > li:nth-child(20)','el => el.title')
        Screen_size = await page.Jeval('#detail > div.tab-con > div:nth-child(1) > div.p-parameter > ul.parameter2.p-parameter-list > li:nth-child(21)','el => el.title')
```

```
        it1= await page.Jx('//div[@data-tab="item"]/div[1]/ul[2]/li[contains(string(),"固态硬盘")]')
        for i in it1:
            SDD=await (await i.getProperty('title')).jsonValue()
        it2 = await page.Jx('//div[@data-tab="item"]/div[1]/ul[2]/li[contains(string(),"机械硬盘")]')
        for i in it2:
            HDD=await (await i.getProperty('title')).jsonValue()
        print(Price,Name,Weight,Screen_color_gamut,Type,Thickness,Memory,IP,Color,Processor,Screen_refresh_rate,Graphics_card_model,Screen_size,SDD,HDD)
        # 数据存储
        Info.append([Price,Name,Weight,Screen_color_gamut,Type,Thickness,Memory,IP,Color,Processor,Screen_refresh_rate,Graphics_card_model,Screen_size,SDD,HDD])
```

2.5.5 商品评价爬取

该部分涉及页面点击事件的触发，通过 css 选择器定位事件入口，调用 click 方法点击，完成相应的触发事件，此时爬虫脚本可获取目标内容。

```
    # 点击评价按钮，爬取评论
        await page.click('#detail > div.tab-main.large > ul > li:nth-child(5)')
        await page.waitFor(2500)
        # 爬取好评率
        rate = await page.Jeval('#comment > div.mc > div.comment-info.J-comment-info > div.comment-percent > div','el => el.innerText')
        pprint(rate)
        pprint(pages)
        # 点击好评
        await page.click('#comment > div.mc > div.J-comments-list.comments-list.ETab > div.tab-main.small > ul > li:nth-child(5) > a')
        await page.waitFor(2500)
        c1 = await page.content()
        comm_1 = etree.HTML(c1)
        comment_1 = comm_1.xpath('//*[@id="comment-4"]/div/div[2]/p/text()')
        # 点击中评
        await page.click('#comment > div.mc > div.J-comments-list.comments-list.ETab > div.tab-main.small > ul > li:nth-child(6) > a')
        await page.waitFor(2500)
        c2 = await page.content()
```

```
        comm_2 = etree.HTML(c2)
        comment_2 = comm_2.xpath('//*[@id="comment-5"]/div/div[2]/p/text()')
        # 点击差评
        await page.click('#comment > div.mc > div.J-comments-list.comments-list.ETab > div.tab-main.small > ul > li:nth-child(7) > a')
        await page.waitFor(2500)
        c3 = await page.content()
        comm_3 = etree.HTML(c3)
        comment_3 = comm_3.xpath('//*[@id="comment-6"]/div/div[2]/p/text()')
        await browser.close()    # 关闭浏览器
    print('爬取信息耗时：',time.time() - start_time )
```

2.5.6 网页输入和主函数入口

```
def start():
    Info = [] # 存储列表
    number = 2 # 网页数量输入
    for pages in range(number):
        start_url = 'https://search.jd.com/Search?keyword=%E7%AC%94%E8%AE%B0%E6%9C%AC%E7%94%B5%E8%84%91&qrst=1&psort=3&suggest=1.def.0.SAK7%7CMIXTAG_SAK7R%2CSAK7_M_AM_L5381%2CSAK7_S_AM_R%2CSAK7_D_HSP_R%2CSAK7_SC_PD_R%2CSAK7_SM_PB_R%2CSAK7_SM_PRK_R%2CSAK7_SM_PRC_R%2CSAK7_SM_PRR_R%2CSAK7_SS_PM_R%7C&wq=%E7%AC%94%E8%AE%B0%E6%9C%AC%E7%94%B5%E8%84%91&shop=1&psort=3&pvid=9682685401bf42fa81d3d8fe09e1b0d7&page={0}&s={1}&click=0'.format(pages * 2 + 1, pages * 60 + 1)
    asyncio.get_event_loop().run_until_complete(switch(pages,start_url,Info))
     # 执行数据采集模块，创建异步事件
    print(Info)
if __name__ == '__main__':
    start()
```

第三章、 爬虫数据可视化

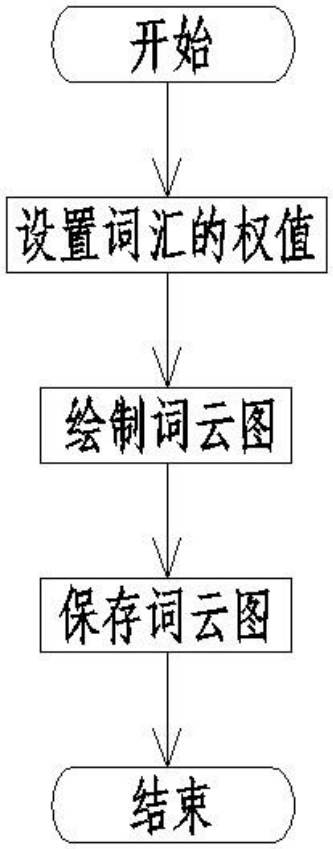
3.1 评论分词

Jieba是目前最好的Python中文分词组件,主要支持3种分词模式;精确模式、搜索引擎模式和全模式。精确模式适合文本分析,主要将句子最精确地切开,只输出最大概率组合。搜索引擎模式适合搜索引擎分词,为了提高召回率,主要在精确模式的基础上对长词再次加工。全模式是指句子存在冗余,将句子中所有可组词

的词语的都扫描处理。三种模式的函数表如下表所示。

模式	函数
精确模式	lcut(s):返回列表类型
	cut(s):返回可迭代的列表类型
搜索引擎模式	lcut__for__search(s):返回列表类型
	cut__for__search(s):返回分词结果
全模式	lcut(s,cut__all = Ture):返回列表类型
	cut(s, cut__all = Ture):返回有概率的单词

表格 1 jieba 分词模式展示表



3.2 正则化

在自然语言处理中，我们通常把停用词、出现频率很低的词汇过滤掉。这个过程其实类似于特征筛选的过程。当然停用词过滤，是文本分析中一个预处理方法。它的功能是过滤分词结果中的噪声。

re 模块中提供了一些方法，可以方便在 python 语言中使用正则表达式，re 模块使 python 语言拥有全部的正则表达式功能。

3.3 词频统计

Collection 包可以为我们进行统计分析提供方便，其中的 Counter()方法使用场景较多，可以与字符串数据配套使用，在数据预处理及探索阶段发挥作用。使用 Counter 方法来统计每个单词的频数。

3.4 数据可视化

词云以词语为基本单位,更加直观和艺术的展示文本。wordcloud 库是 Python 中的词云展示第三方库,规定特定文本词在文本数据源中出现的次数绘制词云的形状、尺寸和颜色。本文为了使用户能够瞬间理解弹幕的关键信息,利用越高频越突出的视觉效果,将分析出来的高频词汇用词云图进行展示。频率越高的词汇,其尺寸越大,反之,越小。

词云图展示弹幕高频词汇的流程是首先配置词云图的对象参数,加载通过数据分析得到满足条件的弹幕数据，将查询到的所有词汇根据出现频率与权重,绘制出词云图并且进行展示,最后将词云图保存本地电脑。词云图展示流程图如下图所示。

图表 6 词云绘制流程

通过展示词出现的次数,展示词在词云中所占的区域以及词汇的形状,从视觉上达到更加抢眼直观的效果,可以使乏味的文本数据散发活力，用户可立即理解关键信息,词云图的视觉效果如图所示。

图表 7 词云效果展示图

3.5 代码实现

```
# 数据可视化(词云绘制)
    Comment = []
    for i in comment_1:
        Comment.append(i)
    for i in comment_2:
        Comment.append(i)
    for i in comment_3:
        Comment.append(i)
    print(Comment)
    text = []
    for i in Comment:
        t = re.sub('([^\u4e00-\u9fa5])', '', i)
        ty = jieba.lcut(t)
        for j in ty:
```

```
            text.append(j)
        print(text)
        words = Counter(text)
        print(words.most_common(10))
        wordscloud = WordCloud(
            background_color='white',  # 设置背景颜色 默认是 black
            width = 900,
            height = 600,
            max_words = 100,  # 词云显示的最大词语数量
            font_path ='simhei.ttf',  # 设置字体 显示中文
            max_font_size = 99,  # 设置字体最大值
            min_font_size = 16,  # 设置子图最小值
            random_state = 50  # 设置随机生成状态，即多少种配色方案
        ).generate_from_frequencies(words)
        # 显示生成的词云图片
        fig = plt.figure()
        plt.imshow(wordscloud, interpolation='bilinear')
        plt.axis('off')
        plt.savefig('{}.png'.format(pages))
```

第四章、总结

本文所介绍的网络爬虫脚本程序，通过 python 语言中 pyeppter 库完成网页请求、网页跳转，事件触发、数据采集等，满足了对京东售卖网站信息爬取要求。

在后续步骤中对爬取的数据进行处理优化，最后使用 wordcloud 库和 Matplotlib 库等第三方库进行词云图绘制，通过商品的好评率给出购买意见。有效提高对商品数据的检索效率，在商品售卖网站海量数据里快速找到所需信息，能够为购买者提供参考。

完整代码展示存放在我个人 GitHub 仓库：

https://github.com/Y-John-17/crawler.git