中国科学院大学
University of Chinese Academy of Sciences

计算机与控制学院
School of Computer and Control Engineering

移动互联网技术

第三章　互联网数据获取技术

简单爬虫（二）

王文杰
wangwj@ucas.ac.cn

本节主要内容

- **其他常用库**
- 爬虫搜索策略

Python爬虫其他常用库

- Requests：
- Selenium：
- Lxml
- Beautifulsoup
- pyquery

Requests

- **Requests是用python语言基于urllib编写的，是比urllib更加方便的库，可以节约大量的开发工作。**
- **默认安装好python之后，requests需要通过pip安装：**
  - **pip install requests**
  - **安装完后，可以验证其是否正确安装：**

```
C:\Users\wangwj>python
Python 3.5.4 (v3.5.4:3f56838, Aug  8 2017, 02:17:05) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> import requests
>>> requests.get("http://www.baidu.com")
<Response [200]>
>>>
```

Requests--总体功能

```
import requests

response  = requests.get("https://www.baidu.com")
print(type(response))
print(response.status_code)

print(type(response.text))
print(response.text,'\n')

print(type(response.content))
print(response.content,'\n')

print(type(response.content.decode("utf-8")))
print(response.content.decode("utf-8"))

print("response.cookies = ",response.cookies,'\n')
```

- 多情况下的网站如果直接response.text会出现乱码的问题，所以这个使用response.content
- 这样返回的数据格式其实是二进制格式，然后通过decode()转换为utf-8，这样就解决了通过response.text直接返回显示乱码的问题.

Requests--请求方式

- **基本GET请求**

```
import requests

response = requests.get('http://httpbin.org/get')
print(response.text)
```

- **带参数的GET请求**

```
response1 = requests.get("http://httpbin.org/get?name=zhaofan&age=23")
print(response1.text)
print(response1.content.decode("utf-8"))
```

抓取并保存的例子

- 抓取百度图标：
  - 在元素里面找到其地址：
    https://www.baidu.com/img/baidu_jgylogo3.gif

```
>>> import requests
>>> resp = requests.get("https://www.baidu.com/img/baidu_jgylogo3.gif")
>>> print(resp.text)
>>> print(resp.content)  #响应体的二进制格式
>>> with open("1.gif","wb") as f:
...         f.write(resp.content)
...         f.close()
```

Selenium

- Selenium是一个用于Web应用程序测试的工具。Selenium测试直接运行在浏览器中，就像真正的用户在操作一样。
- 在爬虫的时候，会遇到一些用js做渲染的网页，这时用requests就无法正常获取请求的内容；用selnium库就可以直接驱动浏览器，用浏览器直接执行js的渲染，得到的结果就是渲染过后的页面，就可以爬取到js渲染后的内容了。

安装Selenium

- 安装chromedriver：
  - 在镜像节点下载chromedriver

    http://npm.taobao.org/mirrors/chromedriver/
  - 将下载解压后的chromedriver.exe拷贝到pip文件所在目录下
- 注意：
  - 需要安装Chrome浏览器，其他浏览器也可以
  - 注意chromedriver支持的Chrome浏览器的版本

安装Selenium

- 默认的python安装中是没有安装selenium模块的，需要单独通过pip安装：
  - 进入cmd，运行pip install selenium
  - 安装完后，可以验证其是否正确安装：

```
C:\Users\wangwj>python
Python 3.5.4 (v3.5.4:3f56838, Aug  8 2017,
Type "help", "copyright", "credits" or "li
>>> import selenium
>>> from selenium import webdriver
>>> driver = webdriver.Chrome()
```

Selenium

- Selenium在实现爬虫时，有几个关键命令
  - find_element(s)_by_tag_name
  - find_element(s)_by_css_selector
  - find_element_by_id，如
    - driver.get("http://www.baidu.com")
      driver.find_element_by_id('kw').send_keys('selenium')
      - #selenium 元素查找find_element_by_id方法，找到元素后输入信息

      driver.find_element_by_id('su').click()
      - #selenium 元素查找find_element_by_id方法，找到元素后进行点击

lxml

- **https://lxml.de/**
- **lxml is the most feature-rich and easy-to-use library for processing XML and HTML in the Python language.**
  - **提供了一些xpath解析方法**
- **安装： pip install lxml**
- lxml还支持css语法的选择方式，需要安装cssselect
  - **Pip install cssselect**
- Selenium官网的Document里极力推荐使用CSS locator，而不是XPath来定位元素，原因是CSS locator比XPath locator速度快

beautifulsoup

- **Beautiful Soup**
  - **python的一个第三方库，也是网页解释库，最主要的功能是从HTML或XML网页中抓取数据**
- Beautiful Soup依赖于lxml库，安装Beautiful Soup之前，需要安装好lxml
- 需要安装Beautiful Soup第四个版本
  - Pip install BeautifulSoup4

```
>>> import bs4
>>> print(bs4)
<module 'bs4' from 'C:\\Python36\\lib\\site-packages\\bs4\\__init__.py'>
>>>
```

beautifulsoup

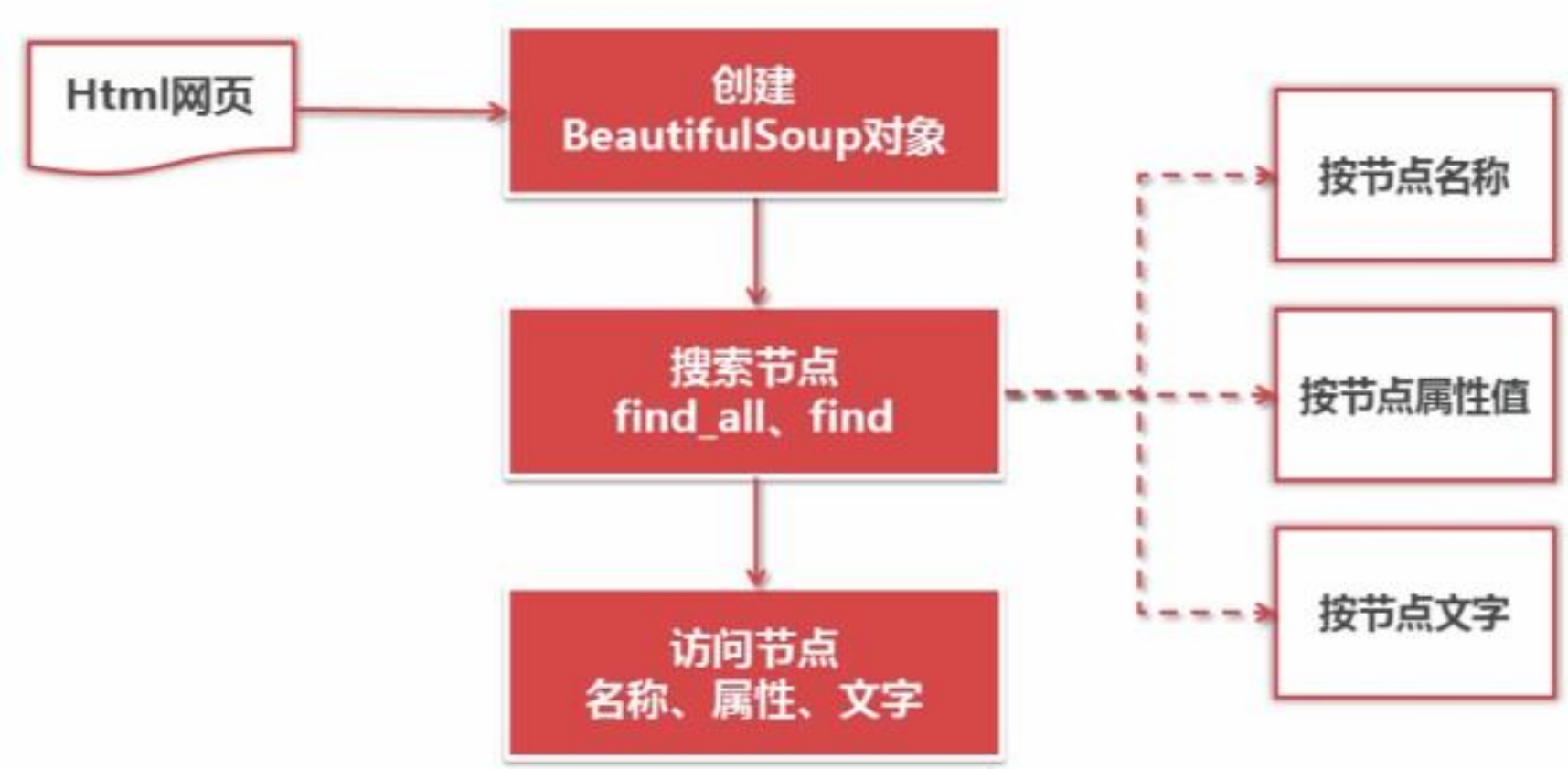
- Beautiful Soup提供一些简单的、python式的函数用来处理导航、搜索、修改分析树等功能。它是一个工具箱，通过解析文档为用户提供需要抓取的数据，因为简单，所以不需要多少代码就可以写出一个完整的应用程序。
- Beautiful Soup自动将输入文档转换为Unicode编码，输出文档转换为utf-8编码。你不需要考虑编码方式，除非文档没有指定一个编码方式，这时，Beautiful Soup就不能自动识别编码方式了。

beautifulsoup

序号	解析库	使用方法	优势	劣势
1	Python标准库	BeautifulSoup(html,'html.parser')	Python内置标准库；执行速度快	中文容错能力较差
2	lxml HTML解析库	BeautifulSoup(html,'lxml')	速度快；容错能力强	需要安装，需要C语言库
3	lxml XML解析库	BeautifulSoup(html,['lxml','xml'])	速度快；容错能力强；支持XML格式	需要C语言库
4	htm5lib解析库	BeautifulSoup(html,'htm5llib')	以浏览器方式解析，最好的容错性	速度慢

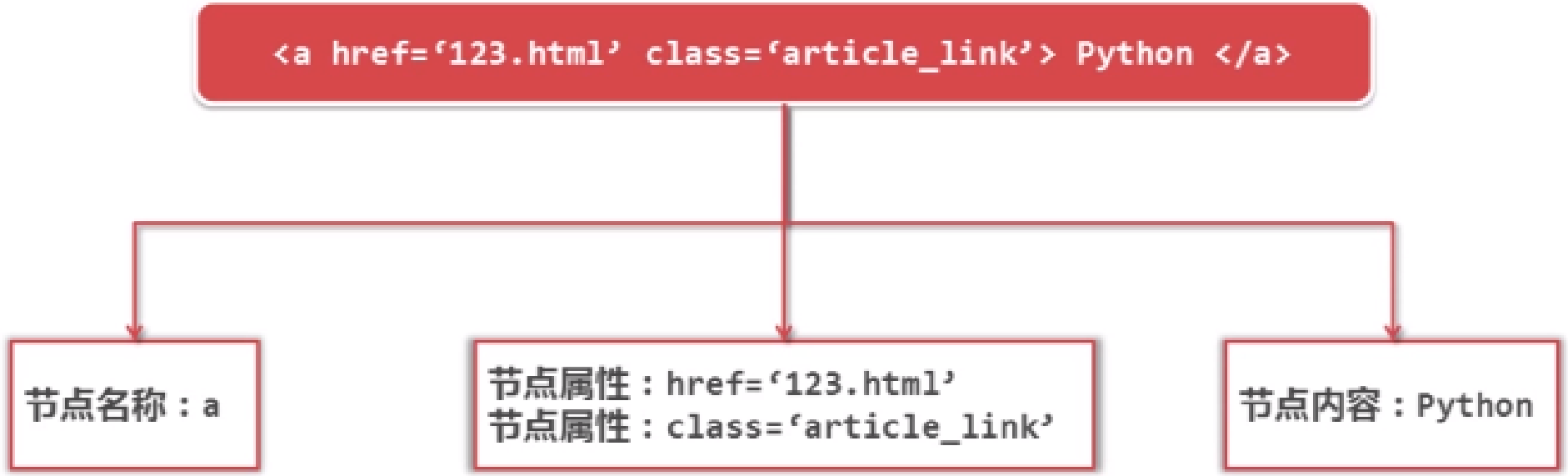
beautifulsoup

- 语法



beautifulsoup

- 例子



beautifulsoup

- 创建beautifulsoup对象

```
from bs4 import BeautifulSoup

#根据HTML网页字符串创建beautifulSoup对象
soup = BeautifulSoup(
    html_doc,                    #HTML文档字符串
    ‘html.parser’ ,              #HTML解释器
    from_encoding=‘utf8’         #指定HTML文档的编码
    )
```

beautifulsoup

- 搜索节点（find_all,find）

```
#方法： find(name, attrs, string)

#查找所以标签为a的节点
soup.find_all('a')

#查找所有标签为a，连接符合/view/123.htm形式的节点
soup.find_all('a',href='/view/123.htm')
soup.find_all(‘a’,href=re.compile(r‘/view/\d+\.htm’))  #bs支持正则表达式匹配

#查找所有标签为div，class为abc，文字为Python的节点
soup.find_all(‘div’,class_=‘abc’, string=‘Python’)
#class增加下划线，是bs为了与python的关键字class区分
```

beautifulsoup

- 访问节点信息

```
#假设，得到了节点：<a href='1.html'>Python</a>

#获取查找到的节点的标签名称
node.name

#获取查找到的a节点的href属性
node['href']

#获取查找到的a节点的链接文字
node.get_text()
```

pyquery

- PyQuery库也是一个非常强大又灵活的网页解析库，PyQuery 是 Python 仿照 jQuery 的严格实现。语法与 jQuery 几乎完全相同。
- 官网地址：http://pyquery.readthedocs.io/en/latest/
- jQuery参考文档： http://jquery.cuishifeng.cn/
- 安装： pip install pyquery

pyquery

- CSS选择器

.class	.color	选择 class="color"的所有元素
#id	#info	选择 id="info"的所有元素
*	*	选择所有元素
element	p	选择所有的 p 元素
element,element	div,p	选择所有 div 元素和所有 p 元素
element element	div p	选择 div 标签内部的所有 p 元素
[attribute]	[target]	选择带有 targe 属性的所有元素
[arrtibute=value]	[target=_blank]	选择 target="_blank"的所有元素

本节主要内容

- 其他常用库
- **爬虫搜索策略**

爬行策略搜索

- 对于互联网大数据的获取，通常需要面对海量的WEB页面
- 页面数量很大时，决定爬行顺序对于提高爬虫效率就变得很关键。
  - 如何对这些页面进行高效的获取是爬虫遇到的主要难点之一。

- 抓取策略
  - 深度优先
  - 宽度优先
  - PageRank
  - 合作抓取的策略

宽度与深度优先抓取

- 深度优先是指网络爬虫会从起始页开始，一个链接一个链接跟踪下去，处理完这条线路之后再转入下一个起始页，继续追踪链接

```
def depth_tree(tree_node):
    if tree_node is not None :
        print (tree_node._data)
        if tree_node._left is not None:
            return depth_tree(tree_node._left)
        if tree_node._right is not None:
            return depth_tree(tree_node._right)
```

宽度与深度优先抓取

- 宽度优先，是指将新下载网页发现的链接直接插入到待抓取URL队列的末尾，也就是指网络爬虫会先抓取起始页中的所有网页，然后在选择其中的一个连接网页，继续抓取在此网页中链接的所有网页

```
def level_queue(root):
    """利用队列实现树的广度优先遍历"""
    if root is None:
        return
    my_queue = []
    node = root
    my_queue.append(node)
    while my_queue:
        node = my_queue.pop(0)
        print (node.elem)
        if node.lchild is not None:
            my_queue.append(node.lchild)
        if node.rchild is not None:
            my_queue.append(node.rchild)
```

PageRank—背景

- 分类目录：
  - 早期的搜索引擎采用的方法，即通过人工进行网页分类并整理出高质量的网站
  - Yahoo和hao123
- 文本检索
  - 根据用户查询关键字与网页内容的相关程度返回搜素结果
  - 优点是突破数量的限制
  - 但是结果不是太好，总是有些网页来回倒腾某些关键词使自己的排名靠前

PageRank—提出

- 网页数量急剧膨胀，用户需要有效搜索出有用的信息
- **Google的创始人Larry Page提出的PageRank（网页排名/网页级别）算法是一种网页级别的算法，可用于计算互联网里网站的重要性，以对搜索进行排名**
- **PageRank算法计算量大，因此诞生了Map-Reduce，可分布式计算PageRank**
- **PageRank和BigTable是Google早期的核心**

PageRank—基本思想

- 假定用户一开始随机地访问网页集合中的一个网页，以后跟随网页的向外链接向前浏览网页，而不回退浏览，浏览下一个网页的概率就是被浏览网页的**PageRank值**
- 页面重要性的计算问题，在搜索引擎系统中是非常重要的。
  - 一方面用于搜索结果的排序
  - 另一方面在爬虫爬行的过程中用来对页面的优先级估算。

PageRank—基本思想

- Google的PageRank根据网站的外部链接和内部链接的数量和质量来衡量网站的价值。
  - 数量假设：在Web图模型中，一个页面节点接收到的其他网页指向的入链数量越多，那么这个页面越重要
  - 质量假设：指向页面A的入链质量不同，质量高的页面会通过链接向其他页面传递更多的权重。所以越是质量高的页面指向A，则页面A越重要。

PageRank

- PageRank背后的概念是，每个到页面的链接都是对该页面的一次投票，被链接的越多，就意味着被其他网站投票越多。
- 这个就是所谓的“链接流行度”——衡量多少人愿意将他们的网站和你的网站挂钩。
  - PageRank这个概念引自学术中一篇论文的被引述的频度——即被别人引述的次数越多，一般判断这篇论文的权威性就越高。

- 一个网页多次被引用，则可能是很重要的；
- 如果一个网页没有被多次引用，但是如果被重要的网页引用，也有可能是重要的网页。
- 网页重要性的评价主要有三种：

（1）认可度越高的网页越重要，即反向链接越多的网页越重要。

（2）反向链接的源网页质量越高，被这些高质量网页的链接指向的网页越重要。

（3）链接数越少的网页越重要。

PageRank—模型

- PageRank算法计算每一个网页的PageRank值，然后根据这个值的大小对网页的重要性进行排序。
- 它的思想是模拟一个悠闲的上网者，上网者首先随机选择一个网页打开，然后在这个网页上浏览了几分钟，跳转到该网页所指向的链接，这样漫无目的地在网页上跳来跳去，

  PageRank就是估计这个悠闲的上网者进入各个网页上的概率

PageRank--算法1

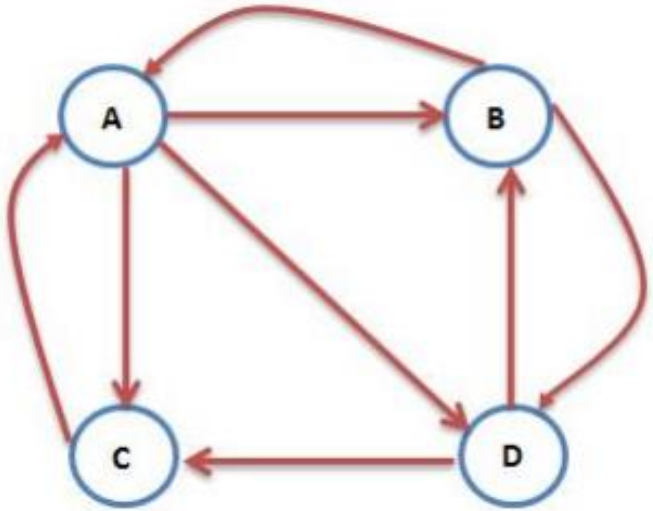
- 是根据网页之间相互的超链接计算页面级别的方法：
  - 每多出一个指向自己的链接，就加上该指向节点的PR值

A的PR值就可以表示为

PR(A)=PR(B)+PR(C)

B和D都不止有一条出链，例如从B网页打开A和C是同概率：

$$PR(A) = \frac{PR(B)}{2} + \frac{PR(C)}{1}$$



PageRank--算法1

- 对于某个页面u，B_u表示指向u的所有页面的集合，那么u的PR值为：

$$PR(u) = \sum_{v \in B_u} \frac{PR(v)}{L(v)}$$

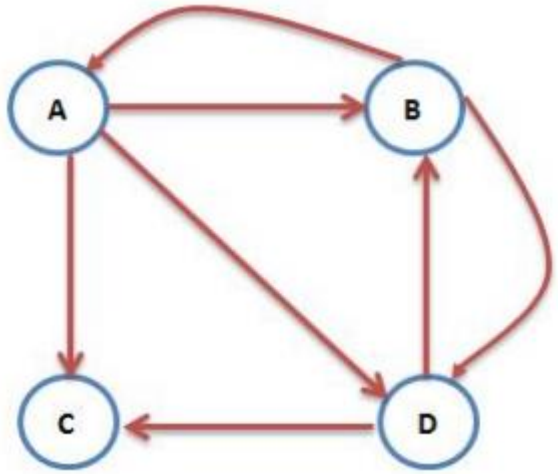
L(v)为页面v所指向的页面个数，即v的出度之和

PageRank—其他考虑

- 实际上，任何节点，即使有出度，也可能会产生随机的跳转，比如d，虽然没有到a的出度，但是也可能随机跳转到a。
- 这种情况，也可以只考虑没有出度的节点，即类似下面的算法

对于没有出链的网页（例如C），设定它对所有网页都有出链

$$PR(A) = \frac{PR(B)}{2} + \frac{PR(C)}{4}$$



- 这里，C因为没有出度，所以它可以随机地跳转到其他网页（包括它自己）。这样就可以增加了上面等式右边第二项。

PageRank--算法2

- 某个页面除了可以通过指向它的超链接进入该页面外，还有一些其他的途径访问该页面，如通过收藏夹或直接输入网址等。因此页面的重要性数值还需要有一部分留给这些直接访问方式
- 引入参数d，称为阻尼因子，指通过链接点击进入该网页的概率，那么通过地址栏输入而跳转的概率就是1-d
- 对页面u，PR值为：

$$PR(u) = 1 - d + d \sum_{v \in B_u} \frac{PR(v)}{L(v)}$$

d取值一般为0.85，这样其他途径访问占0.15

PageRank算法3

- 有人认为：一个页面被访问的随机性应当来自其他所有页面。因此PR值为：

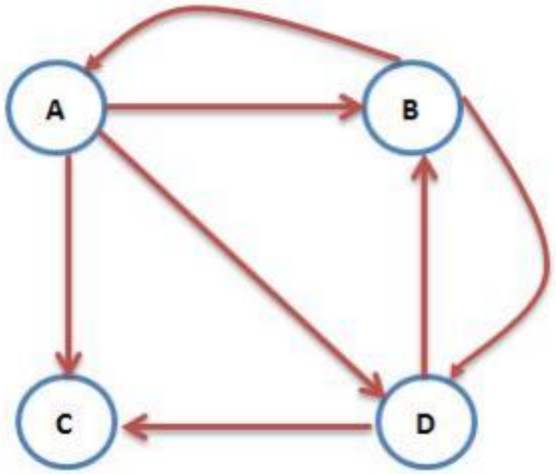
$$PR(u) = \frac{1-d}{N} + d\sum_{v \in B_u} \frac{PR(v)}{L(v)}$$

- 其中N为搜素引擎收录的页面总数
- 上述式子是一种迭代计算，已经证明该计算过程的收敛性，与每个阶段的初始PR值没有关系，收敛时的PR值即为该页面的PR值

PageRank算法3

- 这相当于用户在浏览网页时，停止内链跳转，而是直接在地址栏输入新的地址跳转。
- 假设d为整体的概率，则所有的PR值都需要乘以d，同时任意网页也存在随机跳转到这个网页的可能性。
- 这和算法2一样，主要解决了某些网页没有出链的情况
- 例如：

$$PR(A) = \frac{1-d}{4} + d\frac{PR(B)}{2}$$



PageRank—特点

- 优点：
  - 一个与查询无关的静态算法，所有网页的PageRank值通过离线计算获得，有效减少在线查询时的计算量，极大降低了查询响应时间
- 缺点：
  - 查询一般都具有主题特征，PageRank忽略了主题相关性，导致了结果的相关性与主题性降低
  - 旧页面的等级会比新页面高。因为即使是非常好的新页面也不会有很多上游链接，除非它是某个站点的子站点

Python的PageRank--NetworkX

- networkx在02年5月产生，是用python语言编写的软件包，便于用户对复杂网络进行创建、操作和学习。
- 利用networkx可以生成多种随机网络和经典网络、分析网络结构、建立网络模型、设计新的网络算法、进行网络绘制等。
- http://networkx.github.io/
- https://networkx.github.io/documentation/stable/tutorial.html
- 安装

```
pip install networkx
```

Python的PageRank--NetworkX

- Networkx的主要功能：

```
g= nx.DiGraph()  #构造有向图

g.add_node(url)          #添加节点
 g.add_edge(src, dest)  #添加边
                          #如果有A→B的链接，则A作为src，B作为dest
nx.pagerank(g, 0.9)  #计算PageRank，g为有向图
                          #0.9是PR的随机跳转概率，也称为阻尼系数
```

合作抓取的策略

- 这种方法是由站点主动向爬虫提供站点内各个页面的重要性信息
  - 通过这种信息，网站中的新增加或经常更新的页面内容能及时地被前来爬行的爬虫所获取，以提高爬虫的执行效率
- 主要涉及的协议包括：**Robots**协议

Robots协议

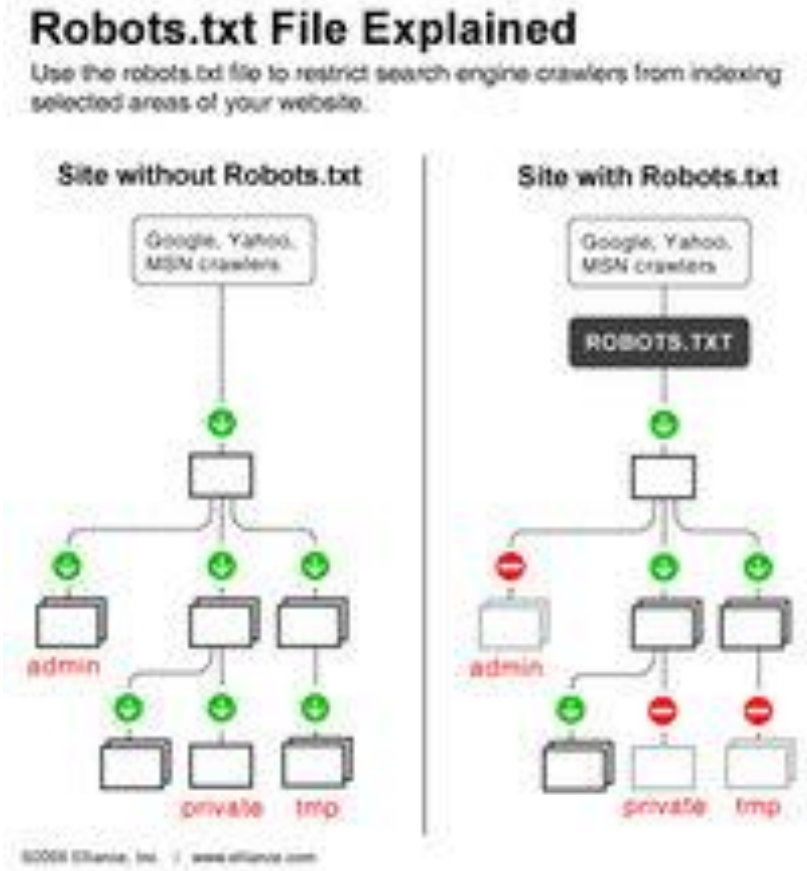
- Robots协议（也称为爬虫协议、机器人协议等）的全称是“网络爬虫排除标准”（Robots Exclusion Protocol）
- 网站管理员可通过**robots.txt**来定义能够被网络爬虫所访问的权限，告诉搜索引擎哪些页面可以抓取，哪些页面不能抓取。
- Robots协议的本质是网站和搜索引擎爬虫的沟通方式，用来指导搜索引擎更好地抓取网站内容，而不是作为搜索引擎之间互相限制和不正当竞争的工具。

原则

- **Robots协议是国际互联网界通行的道德规范，基于以下原则建立：**

  1、搜索技术应服务于人类，同时尊重信息提供者的意愿，并维护其隐私权；

  2、网站有义务保护其使用者的个人信息和隐私不被侵犯。



- **Robots.txt** 是存放在站点根目录下的一个纯文本文件。
  - 例如：https://www.baidu.com/robots.txt

    https://www.taobao.com/robots.txt
  - 当一个搜索引擎爬虫访问一个站点时，它会首先检查该站点根目录下是否存在**robots.txt**，
    - 如果存在，搜索引擎爬虫就会按照该文件中的内容来确定访问的范围，
    - 如果不存在，爬虫就可沿着链接抓取

Robots协议使用格式

（1）User-agent：

- 用于描述搜索引擎爬虫的名字，在“Robots.txt”文件中，如果有多条User-agent记录说明有多个搜索引擎爬虫会受到该协议的限制，对该文件来说，至少要有一条User-agent记录。如果该项的值设为*，则该协议对任何搜索引擎爬虫均有效。

（2）Disallow：

- 用于描述不希望被访问到的一个URL，这个URL可以是一条完整的路径，也可以是部分的，任何以Disallow开头的URL均不会被Robot访问到。

robots.txt用法举例

- 通过“**/robots.txt**”禁止所有搜索引擎爬虫抓取“**/bin/cgi/**”目录，以及“**/tmp/**”目录和**/foo.html**文件，设置方法如下：

  User-agent： *

  Disallow： /bin/cgi/

  Disallow： /tmp/

  Disallow： /foo.html

- 通过“**/robots.txt**”只允许某个搜索引擎抓取，而禁止其他的搜索引擎抓取。
- 如：只允许名为“**slurp**”的搜索引擎爬虫抓取，而拒绝其他的搜索引擎爬虫抓取“**/cgi/**”目录下的内容，设置方法如下：

  User-agent： *

  Disallow： /cgi/

  User-agent： slurp

  Disallow：

- 禁止任何搜索引擎抓取我的网站，方法如下：

  User-agent： *

  Disallow： /

- 只禁止某个搜索引擎抓取我的网站。如：只禁止名为“**slurp**”的搜索引擎蜘蛛抓取，方法如下：

  User-agent： slurp

  Disallow： /

urllib.robotparser` --- Parser for robots.txt

- https://docs.python.org/3.0/library/urllib.robotparser.html
- This module provides a single class, RobotFileParser.
  - This class provides a set of methods to read, parse and answer questions about a single :`robots.txt` file.
- set_url(url) :
  - Sets the URL referring to a robots.txt file.
- read()
  - Reads the robots.txt URL and feeds it to the parser.
- can_fetch(useragent, url)
  - Returns True if the useragent is allowed to fetch the url according to the rules contained in the parsed robots.txt file.

```
>>> import urllib.robotparser
>>> rp = urllib.robotparser.RobotFileParser()
>>> rp.set_url('http://www.baidu.com/robots.txt')
>>> rp.read()
>>> url = 'http://www.baidu.com'
>>> user_agent = 'Baiduspider'
>>> rp.can_fetch(user_agent, url)#是否允许指定的用户代理访问网页
True
>>> url = 'http://www.baidu.com/baidu'
>>> rp.can_fetch(user_agent, url)
False
>>> url = 'http://www.baidu.com/ulink?'
>>> rp.can_fetch(user_agent, url)
False
>>> url = 'http://www.baidu.com/ulink'
>>> rp.can_fetch(user_agent, url)
False
>>> url = 'http://www.baidu.com/tmp'
>>> rp.can_fetch(user_agent, url)
True
```

- http://www.sohu.com/robots.txt

```
User-agent: Baiduspider
Disallow: /*?*
```

```
>>> rp.set_url("http://www.sohu.com/robots.txt")
>>> rp.read()
>>> rp.can_fetch("Baiduspider", "http://www.sohu.com/")
True
>>> rp.can_fetch("Baiduspider", "http://www.sohu.com/*?*")
False
```

技术不会停下脚步，学习永无止境。