​

**1.定义：**

代替人们自动地在互联网中进行数据信息的采集与整理。

**2.分类**

（1）通用网络爬虫

访问全互联网资源，

将互联网中的网页下载到本地，形成一个互联网内容的镜像备份。

（2）聚焦网络爬虫

有选择性地访问与目标主题相关的网页，获取所需要的数据。

（3）增量式网络爬虫

只抓取新产生或者已经发生变化的网页的网络爬虫。

（4）深层网络爬虫

是指抓取深层网页的网络爬虫。

**3.网络**

表层网络

是指传统搜索引擎可以索引的页面，主要以超链接可以到达的静态网页构成的网页。

深层网络

是指大部分内容无法通过静态链接获取的，只能通过用户提交一些关键词才能获取的网页，如用户注册后内容才可见的网页。

**3.robots 协议**

告知网络爬虫哪些网页是允许被抓取的，哪些网页是禁止被抓取的。

**（1）User-agent：**用于指定网络爬虫的名称。

若该选项的值为“\*”，则说明robots.txt文件对任何网络爬虫均有效。

带有“\*”号的User-agent选项只能出现一次。

**（2）Disallow：**用于指定网络爬虫禁止访问的目录。

若Disallow选项的内容为空，说明网站的任何内容都是被允许访问的。

在robots.txt文件中，至少要有一个包含Disallow选项的语句。

**（3）Allow：**用于指定网络爬虫允许访问的目录。

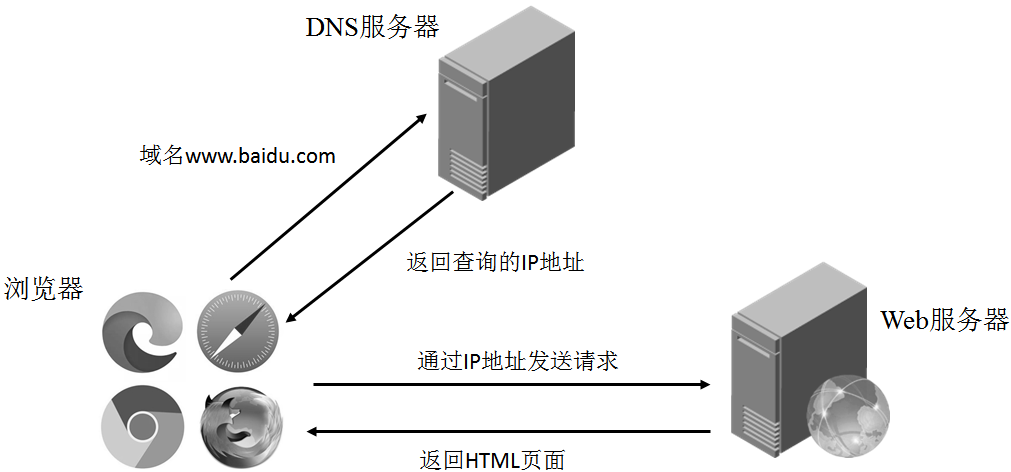
**（4）Sitemap：**用于告知网络爬虫网站地图的路径。

主要说明网站更新时间、更新频率、网址重要程度等信息。

**4.防范**

添加User-Agent字段、降低访问频率、设置代理服务、识别验证码

**5.浏览器加载网页过程**

​

编辑

 （图1 浏览器加载网页过程）

**6.抓取静态网页技术**

**6.1 概念**

  静态网页是HTML格式的网页，这种网页在浏览器中呈现的内容都会体现在源代码中

**6.2request请求**

步骤：  
 （1）获取url url = 'http.....'

（2）发送请求 response = request.get(headers = headers,url = url,params = params)

post请求+data，json

param = {

'query':kw

}

form\_data={

'参数1': '参数值',

'参数2':'参数值',

}

  (3) 获取响应 名字 = response.text/json（）

（4）持久化存储

fp = open('./文件名.json','w',encoding = 'utf-8') #json

json.dump(名字,fp = fp,ensure\_ascii=False)

with open(filename,'w',encoding = 'utf-8')as fp: #text

fp.write(名字)

**7.解析网页数据技术**

**7.1 正则表达式**

正则表达式是一种文本模式，这种模式描述了匹配字符串的规则，用于检索字符串中是否有符合该模式的子串，或者对匹配到的子串进行替换。

|  |  |
| --- | --- |
| 元字符 | 说明 |
| . | 匹配任何一个字符（除换行符外） |
| ^ | 匹配字符串的开头 |
| $ | 匹配字符串的末尾 |
| | | 连接多个子表达式，匹配与任意子表达式模式相同的字符串 |
| [] | 字符组，匹配其中的出现的任意一个字符 |
| - | 连字符，匹配指定范围内的任意一个字符 |
| ? | 匹配其前导字符0次或1次 |
| \* | 匹配其前导字符0次或多次 |
| + | 匹配其前导字符1次或多次 |
| {n} | 匹配其前导字符n次 |
| {m,n} | 匹配其前导字符m~n次 |
| () | 分组，匹配子组 |

|  |  |
| --- | --- |
| 预定义字符集 | 说明 |
| \w | 匹配下画线“\_”或任何字母（a~z，A~Z）与数字（0~9） |
| \s | 匹配任意的空白字符，等价于[<空格>\t\r\n\f\v] |
| \d | 匹配任意数字，等价于[0-9] |
| \b | 匹配单词的边界 |
| \W | 与\w相反，匹配非字母或数字或下画线的字符 |
| \S | 与\s相反，匹配任意非空白字符的字符，等价于[^\s] |
| \D | 与\d相反，匹配任意非数字的字符，等价于[^\d] |
| \B | 与\b相反，匹配不出现在单词边界的元素 |
| \A | 仅匹配字符串开头，等价于^ |
| \Z | 仅匹配字符串结尾，等价于$ |

**7.2 xpath**

7.2.1介绍

XPath是XML路径语言，用于从HTML或XML格式的数据中提取所需的数据。

|  |  |
| --- | --- |
| 表达式 | 说明 |
| 节点名称 | 选取此节点的所有子节点 |
| / | 从根节点选取直接子节点，相当于绝对路径 |
| // | 从当前节点选取后代节点，相当于相对路径 |
| . | 选取当前节点 |
| .. | 选取当前节点的父节点 |
| @ | 选取属性节点 |

|  |  |
| --- | --- |
| 通配符/函数 | 说明 |
| \* | 匹配任何元素节点 |
| @\* | 匹配任何属性节点 |
| node() | 匹配任何类型的节点 |

（ 在XPath中，可以使用“|”运算符连接多个路径表达式）

7.2.2 lxml库

from lxml import etree   #例子

# 从bookstore.xml文件中解析，返回ElementTree类的对象

ele\_tree = etree.parse(r'bookstore.xml')

print(type(ele\_tree))

|  |  |
| --- | --- |
| 属性 | 说明 |
| tag | 获取节点的名称 |
| text | 获取第一个子元素之前的文本。若没有文本，则获得的结果可以是字符串或None |
| tail | 获取当前元素的结束标记之后，下一个同级元素的开始标记之前的文本。若没有文本，则获得的结果可以是字符串或None |
| attrib | 获取属性节点的字典 |

**8.抓取动态网页数据**

**8.1 selenium概念**

selenium可以直接运行在浏览器中，模拟用户使用浏览器完成一些动作，包括自动加载页面、输入文本、选择下拉框、单击按钮、单击超链接等。需要通过浏览器驱动程序WebDriver才能与所选浏览器进行交互。

|  |  |
| --- | --- |
| 属性 | 说明 |
| title | 获取当前页面的标题 |
| current\_url | 获取当前页面的URL地址 |

|  |  |
| --- | --- |
| 方法 | 说明 |
| get() | 根据指定URL地址访问页面 |
| maximize\_window() | 设置浏览器窗口最大化 |
| forward() | 页面前进 |
| back() | 页面后退 |
| refresh() | 刷新当前页面 |
| save\_screenshot() | 对当前浏览器窗口进行截图 |
| quit() | 会话结束时退出浏览器 |
| close() | 关闭当前窗口 |

 例

driver = webdriver.Chrome()

driver.get('http://news.baidu.com/')

# 通过id属性定位元素

element = driver.find\_element\_by\_id('header-wrapper')

# 访问text属性输出元素的文本内容

print(element.text)

**9.借助第三方网站实现验证码识别**

​