上讲回顾：文本分类

- 文本分类方法
  - 朴素贝叶斯（Naïve Bayes）
  - 支持向量机（SVM）
  - 分类效果评测
  - 文本分析代码实现

中国人民大学
RENMIN UNIVERSITY OF CHINA

计算传播理论与实务

2019-2020秋季学期

第三讲

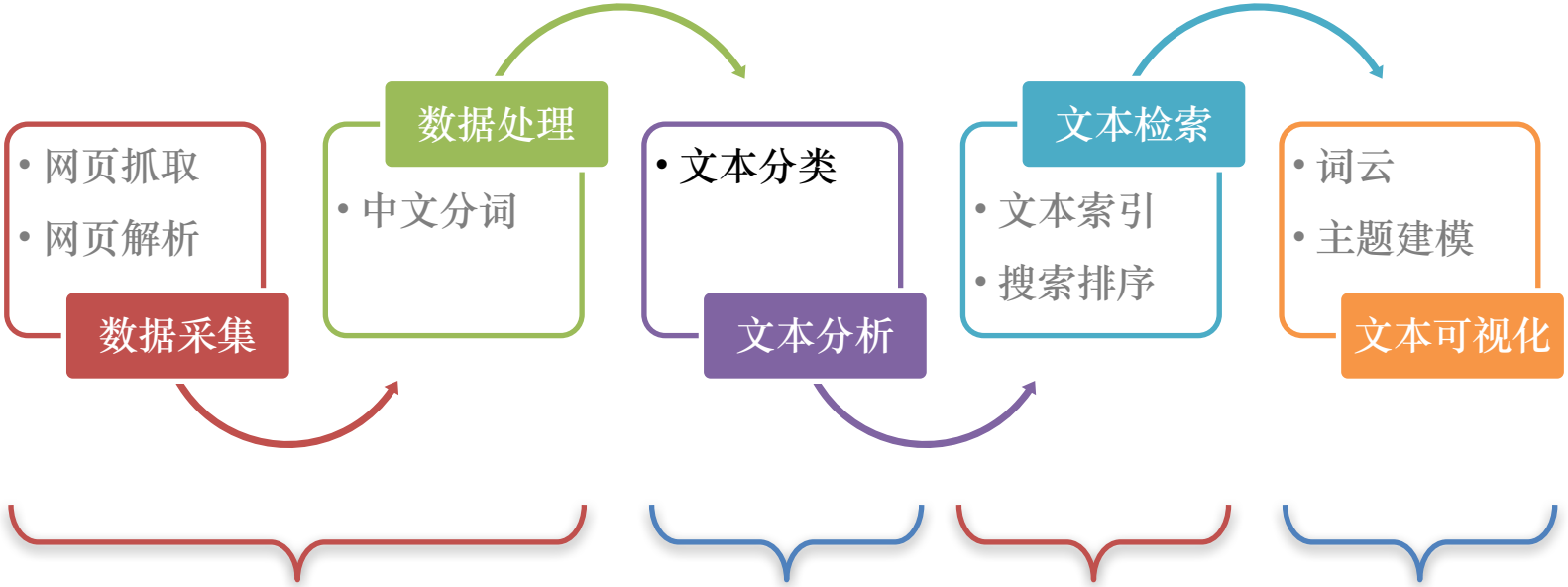
文本分析

授课教师：范举副教授、塔娜讲师

时间：2019年10月21日

文本模块主要内容

- 目标：自动分析出人们在“说”什么
  - 大多新闻与传播领域的数据以文本形态存在



第3.2节

数据采集

网页抓取

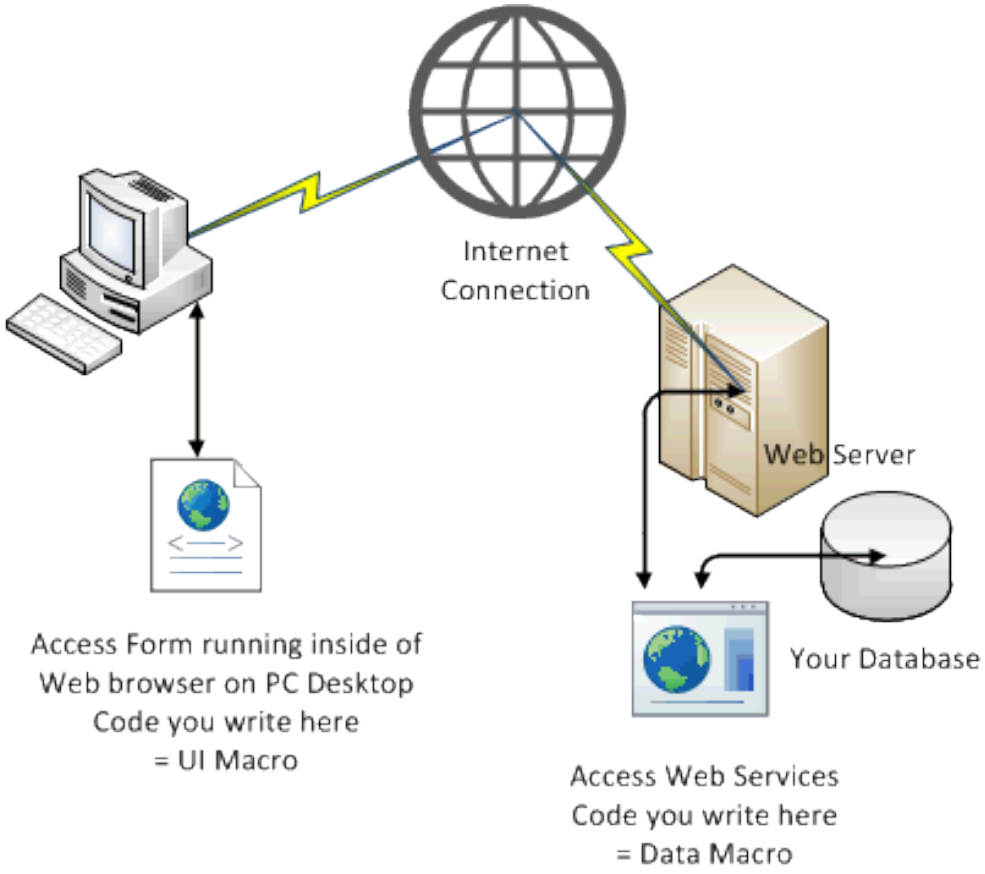
网页解析

SCRAPY实践

第3.2.1节 网页抓取

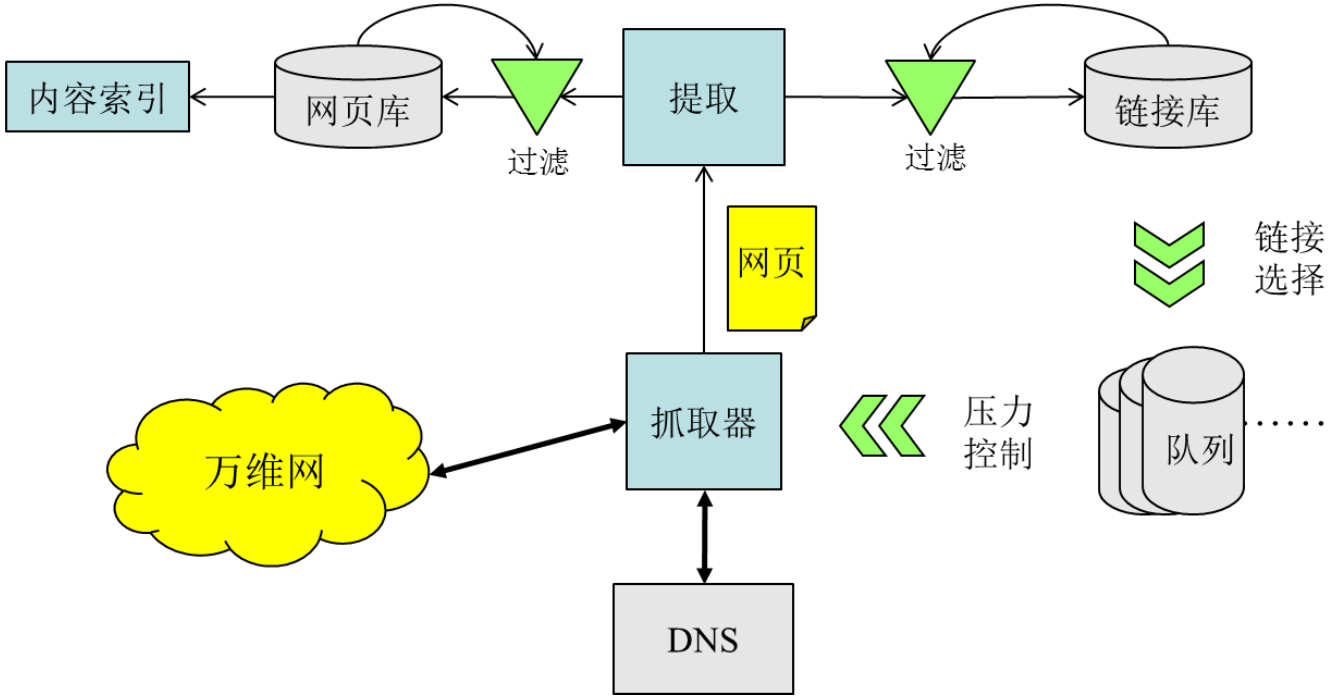
总体结构

互联网应用架构


Internet
Connection
Web Server
Your Database
Access Form running inside of
Web browser on PC Desktop
Code you write here
= UI Macro
Access Web Services
Code you write here
= Data Macro

网页抓取系统

- 典型的抓取系统架构



网页抓取系统

- 抓取系统（抓取前端）
  - 抓取URL并组装plain-html页面
  - 压力控制
- 选取子系统（链接调度）
  - 决策抓取的对象和顺序
- 过滤子系统（重复与低质量检测）
  - 低价值page/link挖掘和控制
  - 资源增长趋势、成本和效益估算

第3.2.1节 网页抓取

抓取前端

互联网应用协议

- 万维网中的各种应用协议
  - 超文本传输协议（HTTP）
    - http://www.w3.org/Protocols/rfc2616/rfc2616.html
    - 会话 = request + response
    - 格式 = header+body
    - 工具 = wget、libcurl、浏览器插件
    - 方法 = GET/HEAD/POST …
  - 其他：https，ftp，Usenet，email

HTTP

- HTTP实现资源的订购和传送。其工作方式类似于快餐点单。
  - 请求(request): 顾客向服务员提出请求
  - 回复(response):服务员根据情况，回应顾客的请求

HTTP工作流程

- 用户在浏览器中单击或输入某个超级链接，HTTP的工作开始
- 一次HTTP操作称为一个事务，其工作过程分为四步：
  1. 客户机与服务器**建立连接**
  2. 客户机向服务器**发送请求**，基本信息包括：统一资源标识符（URL）、协议版本号、MIME信息（请求修饰符、客户机信息等）
  3. 服务器**响应**，基本信息包括：状态行（协议版本号、成功/错误代码)、MIME信息（服务器信息、实体信息等）
  4. 客户端**接收信息**，通过浏览器显示在用户的显示屏上，断开连接释放资源
- 上述过程任意一步出错
  - 错误信息将返回客户端并通过显示屏输出

HTTP对话

- 一个典型的HTTP协议对话

GET / HTTP/1.1
Host: www.abc.com
User-Agent: Baiduspider
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Referer: http://www.baidu.com/s?wd=hello&rsv_bp=0&rsv_spt=3&inputT=6828
Cookie: USERID=e39f70632f09c323dc64; MCITY=-131%3A
Connection: keep-alive
Keep-Alive: 115

发起请求

HTTP/1.1 **200 OK**
Date: Sun, 01 Apr 2012 01:46:00 GMT
Server: Apache
Content-Type: text/html; charset=utf-8
Cache-Control: **max-age**=315360000
Expires: Wed, 30 Mar 2022 01:46:00 GMT
Last-Modified: Thu, 28 Sep 2006 03:50:43 GMT
Etag: "1d40226-0-451b4693"
Accept-Ranges: bytes
Content-Length: 34315
Connection: Keep-Alive

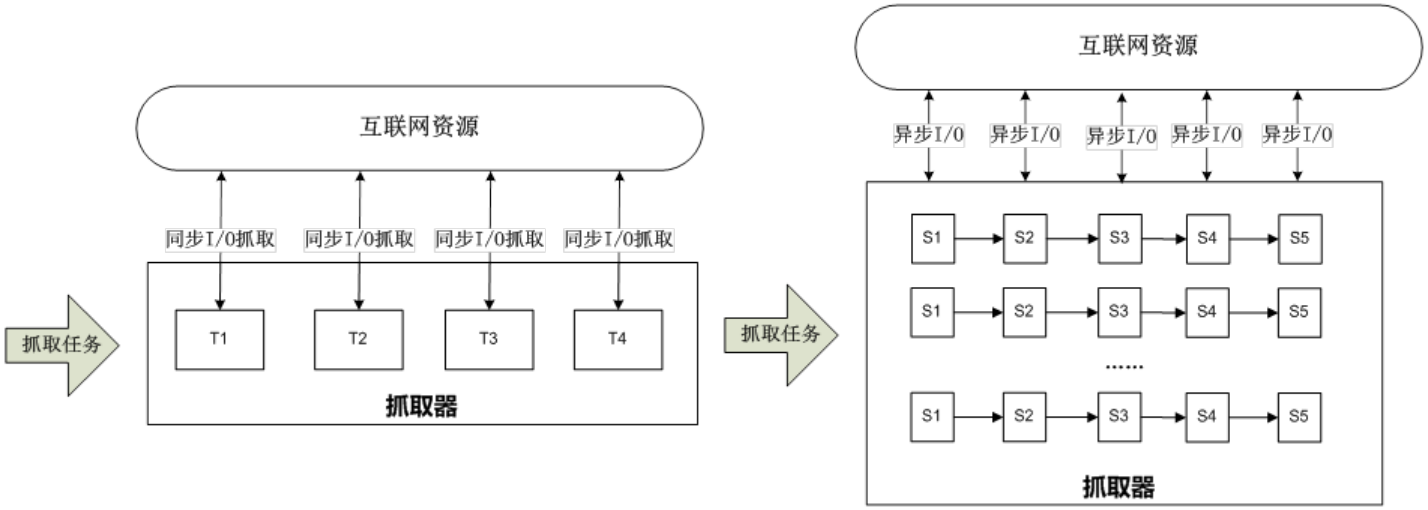
接受应答

```
import urllib.request
url = r'http://www.baidu.com'
res = urllib.request.urlopen(url)
html = res.read().decode('utf-8’)

print(html)
```

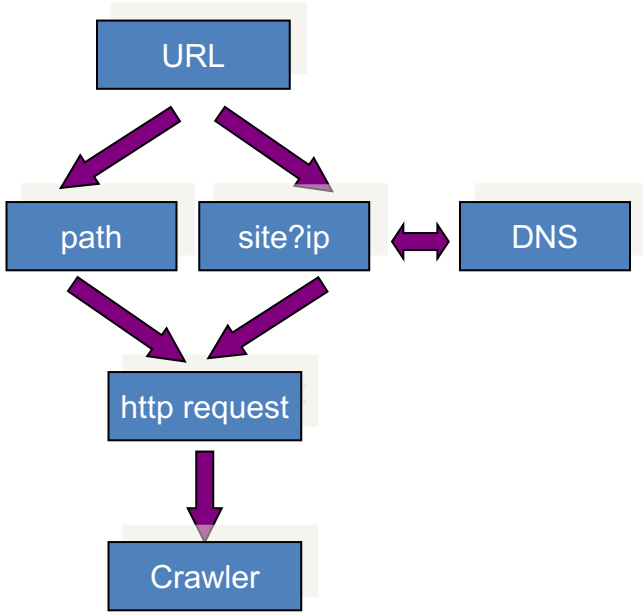
提升抓取性能

- 同步抓取 → 异步抓取
  - 异步I/O模型并不能减少单个页面的抓取速度，但是大大提高了抓取器的并发能力。



提升抓取性能

- DNS：名称 --> IP
- DNS预解析
  - DNS解析是重量级的操作
  - 预存全部站点的IP记录
  - 独立维护和更新



$ ping www.ruc.edu.cn

PING rucweb.appchizi.com (14.116.224.35): 56 data bytes
64 bytes from 14.116.224.35: icmp_seq=0 ttl=3 time=41.806 ms

友好的访问（Politeness）

- 抓取控制：合作共赢
  - 搜索引擎同时具有网站流量的生产者和消费者身份
    - 生产者：为网站带来用户，提升影响和价值
    - 消费者：消耗网站流量，带来带宽、服务器开销

```
<html>
<head>
<meta name="robots" content="noindex,nofollow">
<title>...</title>
</head>
<body>...
```

友好的访问（Politeness）

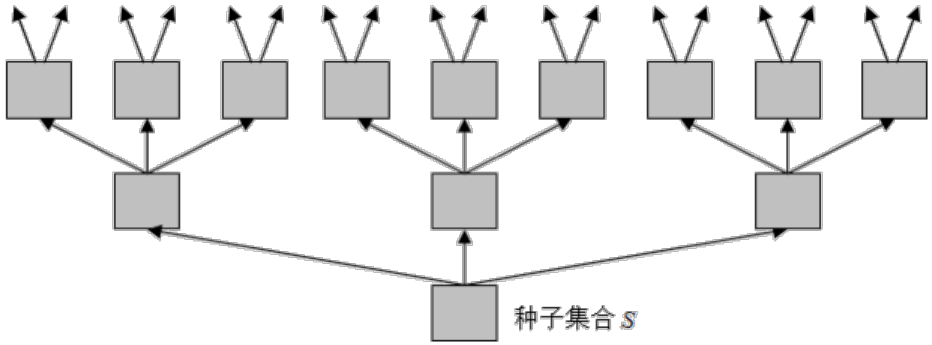
- 抓取控制：合作共赢（续）
  - Meta robots
    - 可以提供页面级更精细的控制
    - 支持多种语义，但不支持针对不同UA分别设置
    - noindex – 禁止检索这张网页
    - nofollow – 禁止抓取该网页的出链
    - noarchive – 禁止保存快照
    - nosnippet – 禁止展示摘要
    - noodp – 禁止展示第三方(DMOZ)摘要
  - 仅是建议标准，各大搜索引擎支持程度不同
- 一对矛盾
  - 友好原则要求我们不能过于频繁地访问某个网站。
  - 由于网页不断更新，我们需要频繁地爬取才能保证搜索结果的新鲜度

第3.2.1节 网页抓取

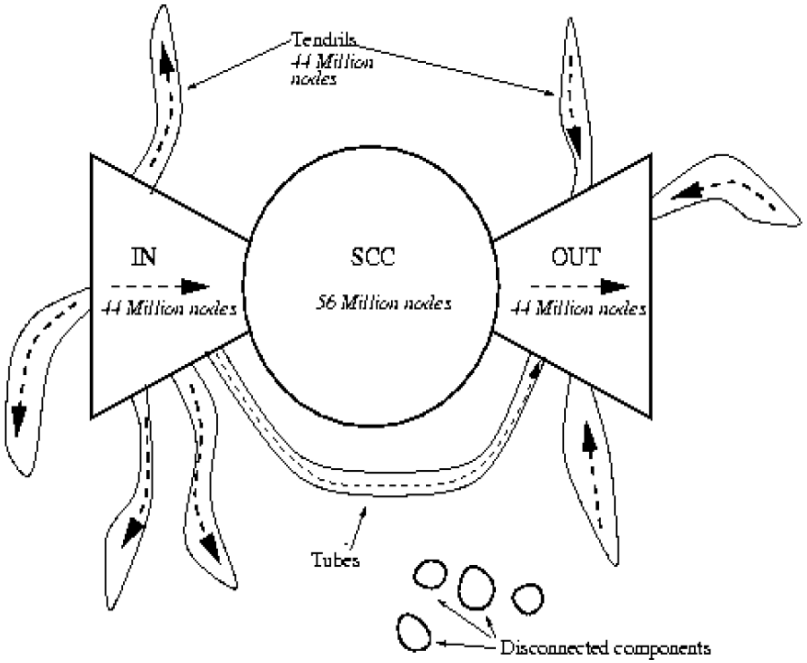
链接调度

链接调度

- 链接调度设计思路
  - 链式反应：以种子站点作为链接自我膨胀的起点
    - 知名站点的首页
    - 网页目录资源：http://www.dmoz.org/；http://www.alexa.com/



互联网的蝴蝶结构



- 蝴蝶结中部（SCC, Strongly Connected Component），网页彼此相连
- 蝴蝶结左部（IN）：导航页居多，通过这类网页可正向链接到SCC
- 蝴蝶结右部（OUT）：权威性网页，大多数SCC都链接向了这些站点
- 蝴蝶结的须脚（Tendrils）：无论采用何种方法都只能遍历有限的网页
- Disconnected components：孤岛

链接调度

- 累积式抓取
  - 定期从种子出发开始扩散、完全替换
  - 优点：实现简单，广度优先或者深度优先即可
  - 缺点：周期长，代价大；时效性、更新率问题
  - 适用范围：
    - 网页快照库的初始化和重建
    - 定向抓取站点的档案收录
  - 典型抓取策略
    - 广度优先、深度优先

链接调度

- 增量式抓取
  - 动态维护现有网页集合，在原来的基础上对新增和变化网页进行收集
  - 优点：节约带宽资源
  - 缺点：
    - 策略复杂，难以确定理想的抓取顺序。
    - 抓取目标多样，需要兼顾覆盖率，更新率，时效性需求。
  - 适用范围：搜索引擎日常抓取

第3.2.1节 网页抓取

重复与低质量检测

重复与低质量检测

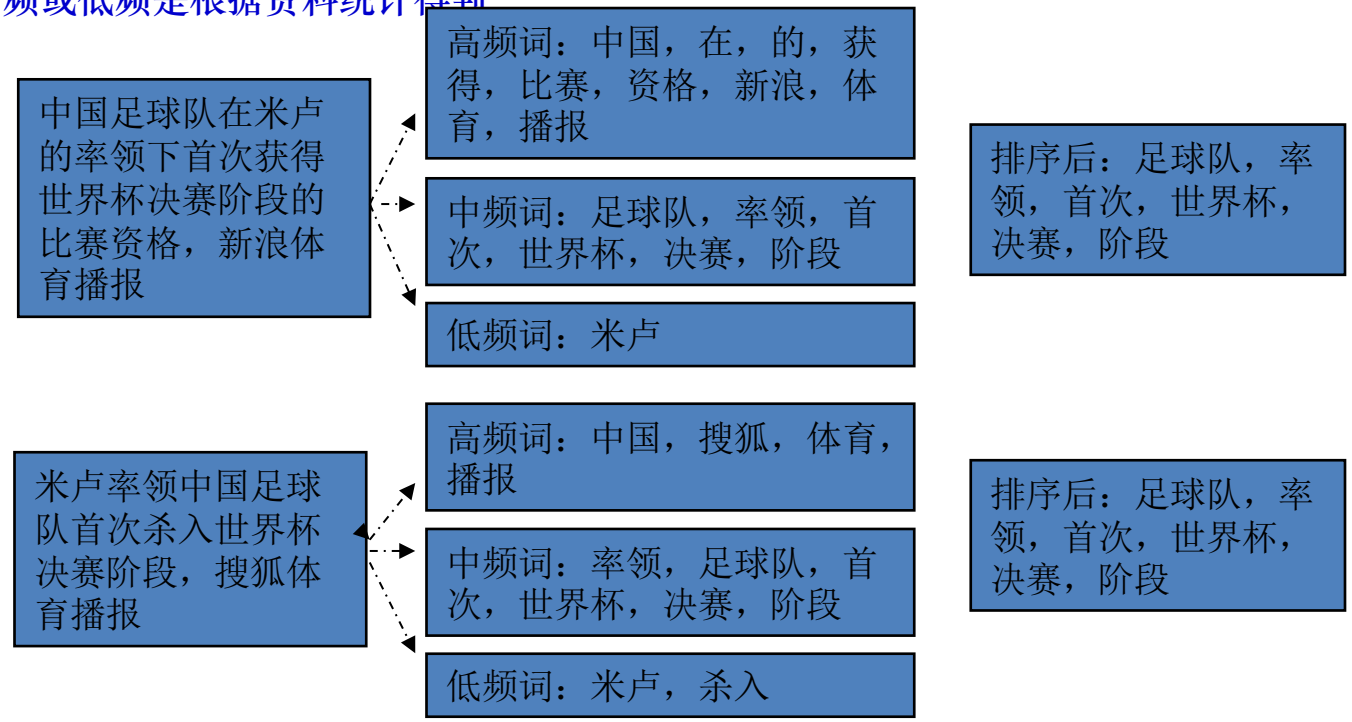
- 低质量网页（如：垃圾网页）

重复与低质量检测

- 重复页面内容检测
  - 有规律重复：站内副本，站点镜像。
  - 无规律重复：转载（允许）、采集（打压）。

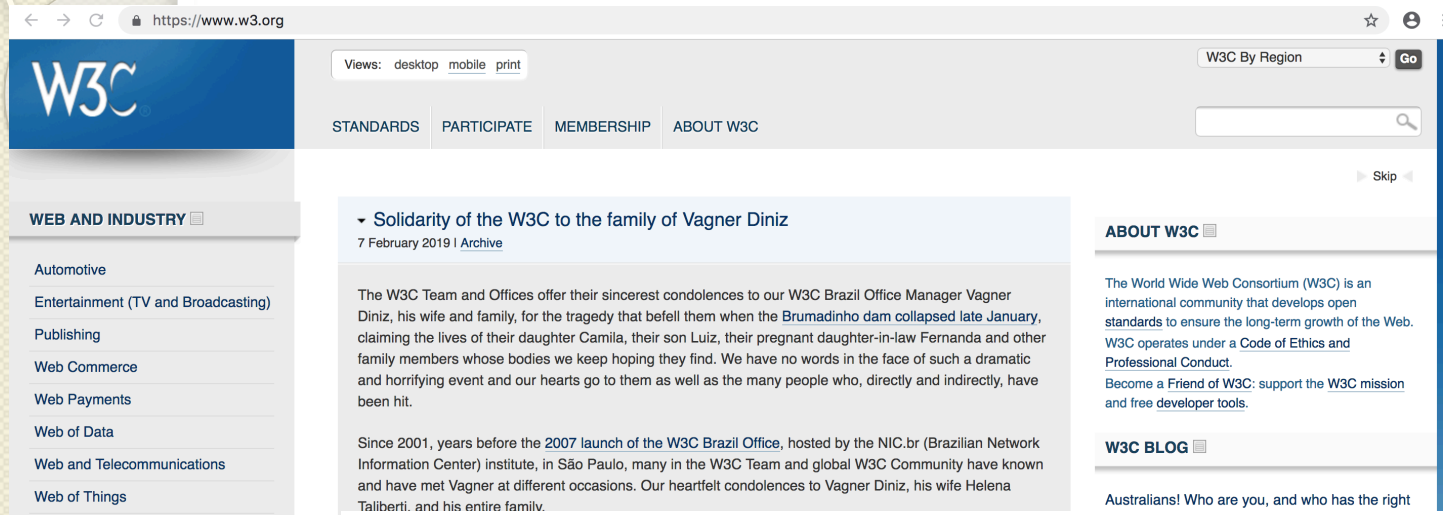
重复检测

- I-Match算法
  - 基本假设：在文档中高频词和低频词不太会影响文章语义，特别高频和特别低频词无法反映文档的真是内容，就像比赛中去掉最高分和最低分
  - 高频或低频是根据资料统计得到



第3.2.2节 网页解析

浏览器显示内容与HTML源代码




```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<!-- Generated from data/head-home.php, ../../smarty/{head.tpl} -->
<head>
<title>World Wide Web Consortium (W3C)</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link rel="Help" href="/Help/" />
<link rel="stylesheet" href="/2008/site/css/minimum" type="text/css" media="all" />
<style type="text/css" media="print, screen and (min-width: 481px)">
/*<![CDATA[*/
@import url("/2008/site/css/advanced");
/*]]>*/
</style>
<link href="/2008/site/css/minimum" rel="stylesheet" type="text/css" media="only screen and (max-width: 480px)" />
<meta name="viewport" content="width=device-width" />
<link rel="stylesheet" href="/2008/site/css/print" type="text/css" media="print" />
<link rel="shortcut icon" href="/2008/site/images/favicon.ico" type="image/x-icon" />
<meta name="description" content="The World Wide Web Consortium (W3C) is an international community where Member organizations, a full-time staff, and the public work together to
develop Web standards." />
<link rel="alternate" type="application/atom+xml" title="W3C News" href="/blog/news/feed/atom" />
</head>
<body id="www-w3-org" class="w3c_public w3c_home">
<div id="w3c_container">
<!-- Generated from data/mast-home.php, ../../smarty/{mast.tpl} -->
<div id="w3c_mast"><!-- #w3c_mast / Page top header -->
<h1 class="logo"><a tabindex="2" accesskey="1" href="/"><img src="/2008/site/images/logo-w3c-mobile-lg" width="90" height="53" alt="W3C" /></a> <span class="alt-logo">W3C</span>
</h1>
<div id="w3c_nav">
<form id="region_form" action="http://www.w3.org/Consortium/contact">
<div><select name="region">
<option selected="selected" title="Get Information about W3C By Region">W3C By Region</option>
<option value="all">All</option>
<option value="au">Australia</option>
<option xml:lang="de" lang="de" value="de">&#xD6;sterreich
(Austria)</option>
<option lang="nl" xml:lang="nl" value="nl">Belgi&#xEB;
(Belgium)</option>
<option value="za">Botswana</option>
```

HTML网页结构

- **Doctype**：文档类型说明，主要告诉浏览器所查看的文件类型
- **<html>标签**：文档开始和结束，是一个双标签，头尾呼应，包含HTML主体内容
- **<head>标签**：含元数据内容，元数据包括：<link>、<meta>、<noscript>、<script>、<style>、<title>，这些内容为浏览器提供信息
  - <meta>：提供关于文档的信息
  - **<title>:** 浏览器左上角的标题
- **<body>标签**：文档内容
- **<a>标签**：超链接

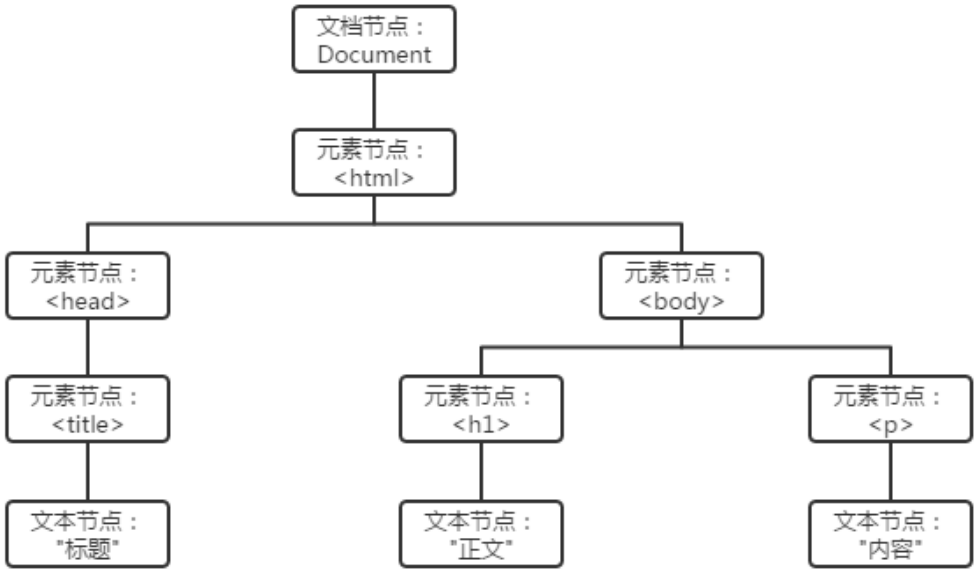
```
1  <!DOCTYPE html>                                  <!--文档类型数据-->
2  <html lang="zh-cn">                              <!--表示HTML文档开始-->
3  <head>                                           <!--包含文档元数据开始-->
4      <meta charset="utf-8">                       <!--声明字符编码-->
5      <title>文档结构</title>                      <!--设置文档标题-->
6  </head>                                          <!--包含文档元数据结束-->
7
8  <body>                                           <!--表示HTML文档的内容-->
9
10 <a href="http://www.baidu.com">百度</a>          <!--一个超链接元素（标签）-->
11
12 </body>                                          <!--表示HTML文档的内容结束 -->
13
14 </html>                                          <!--表示HTML文档结束-->
```

文档内容的组成

- 当前HTML文件本身所携带的内容
  - URL
  - Title
  - Body中的文本
- 当前HTML文件之外的内容
  - Anchor文本
  - Query-click文本
  - ……

文本抽取方法——HTML DOM树

- DOM：W3C文档对象模型
- DOM树定义了HTML（或XML）的逻辑结构
- DOM中定义的方法可以操作html dom，包括抽取其包含的文本内容



文本抽取方法——正则表达式

- **正则表达式**（Regular Expression）：使用单个字符串来描述、匹配一系列匹配某个句法规则的字符串，正则表达式通常被用来检索、替换那些**匹配某个模式的文本**

```
#!/usr/bin/python
import re
line = "Cats are smarter than dogs";
searchObj = re.search( r'(.*) are (.*?) .*', line, re.M|re.I)
if searchObj:
          print "searchObj.group() : ", searchObj.group()
          print "searchObj.group(1) : ", searchObj.group(1)
          print "searchObj.group(2) : ", searchObj.group(2)
else:
          print "Nothing found!!"
```

```
searchObj.group() : Cats are smarter than
dogs
searchObj.group(1) : Cats
searchObj.group(2) : smarter
```

HTML中基本内容抽取

- 标题： r'<title>(.*?)</title>'

```
import re
content = ’<title>this is title</title> other content’
title = re.findall(r'<title>(.*?)</title>', content)
print(title[0])
```

```
>>> content = '<title>this is the title</title> and other content'
>>> title = re.findall(r'<title>(.*?)</title>', content)
>>> print(title[0])
this is the title
>>>
```

HTML中基本内容抽取

- HTML文件中的链接

  锚文本

  - 获取锚文本

```
res = r'<a .*?>(.*?)</a>'
anchor =  re.findall(res, content, re.S|re.M)
for value in anchor:
    print value
```

```
>>> content = 'other text <a href = "http://www.url.com">this is anchor</a>'
>>> anchor = re.findall(r'<a.*?>(.*?)</a>', content)
>>> print(anchor[0])
this is anchor
>>>
```

  - 获取完整链接部分

```
urls=re.findall(r'<a.*?href=(.*?)>.*?<\/a>', content, re.I|re.S|re.M)
```

```
>>> content = 'other text <a href="http://www.url.com">this is anchor</a>'
>>> urls = re.findall(r'<a.*?href=(.*?)>.*?</a>', content)
>>> print(urls[0])
"http://www.url.com"
>>>
```

- 去除所有HTML标签

```
reg = re.compile('<[^>]*>')
reg.sub('',html)
```

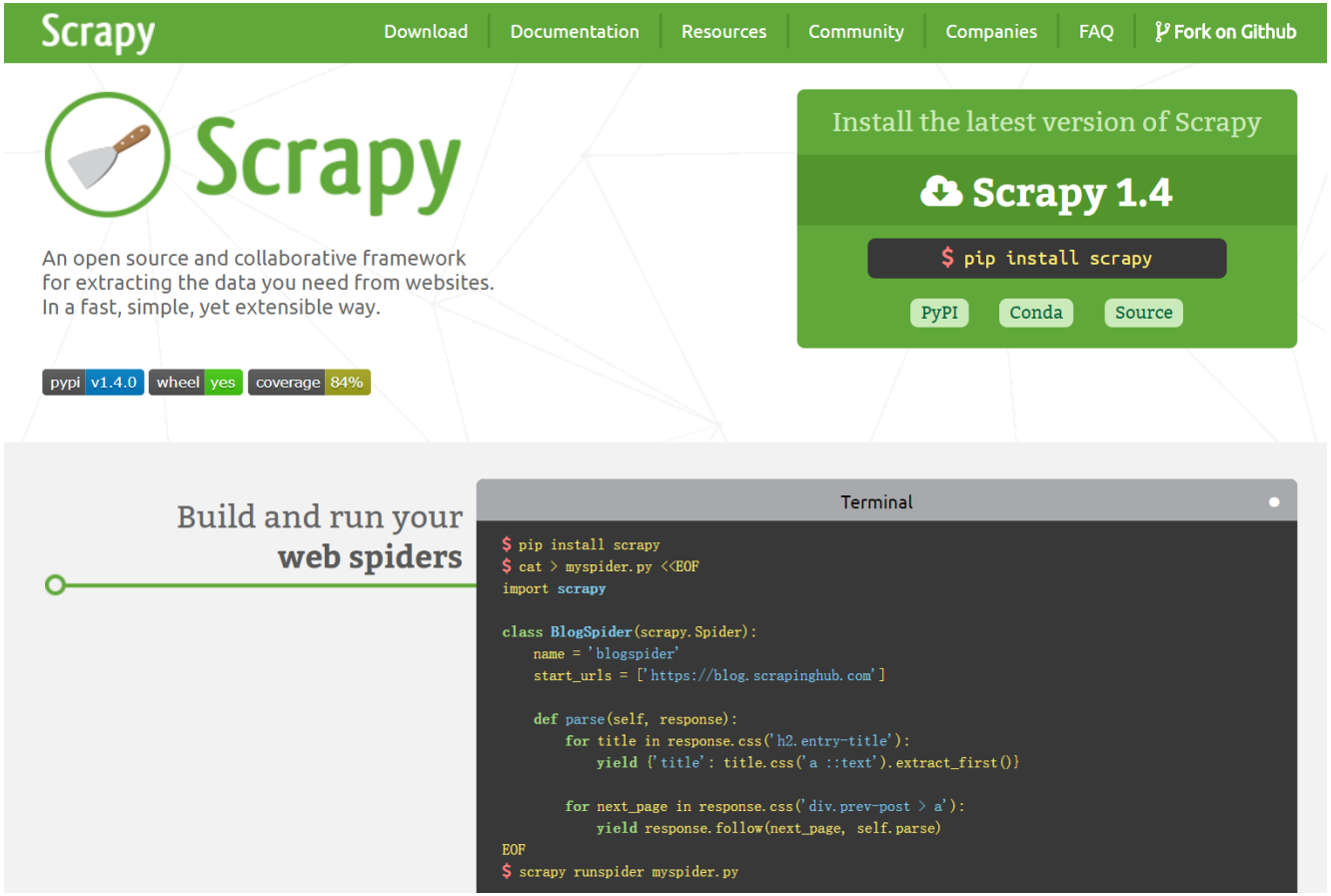
```
>>> content
'other text <a href="http://www.url.com">this is anchor</a>'
>>> reg = re.compile('<[^>]*>')
>>> print(reg.sub('', content))
other text this is anchor
>>>
```

第3.2.2节 Scrap实践

数据采集方法和工具

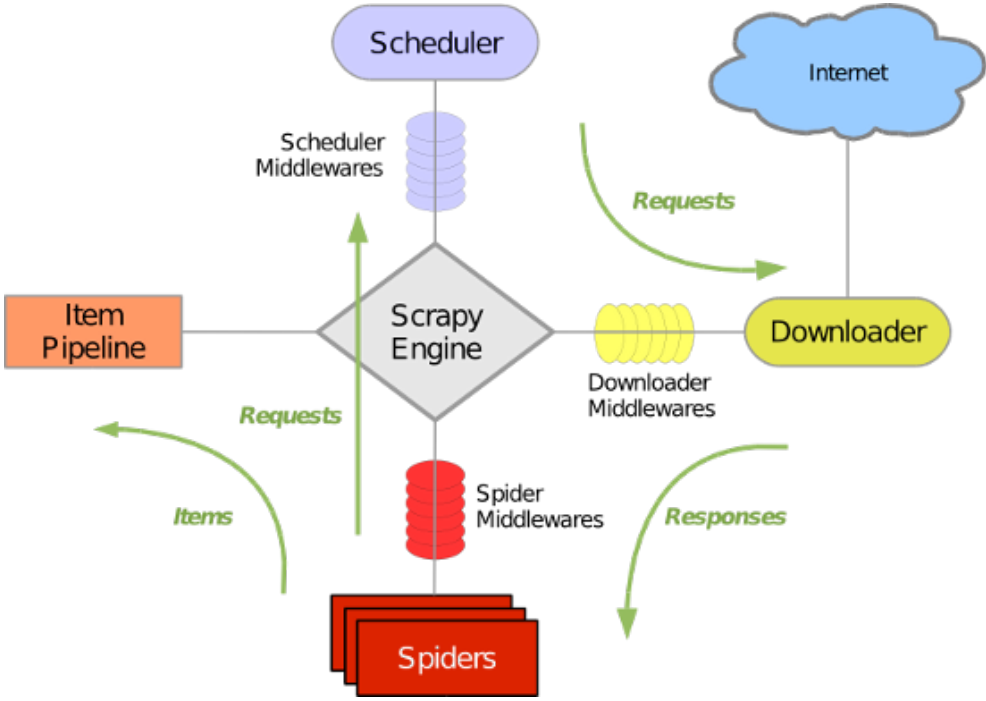
- 人工
  - 浏览器
  - Copy + Paste
- 爬虫工具
  - 自动化程序向网络服务器请求数据，然后对数据进行解析和提取
  - Excel、八爪鱼……
- API接口调用
  - E.g., 新浪微博API
- 可编程工具包
  - Scrapy

https://scrapy.org/



Scrapy

- Scrapy是一个为了爬取网站数据，提取结构性数据而编写的应用框架。 可以应用在包括数据挖掘，信息处理或存储历史数据等一系列的程序中。
- Python语言实现

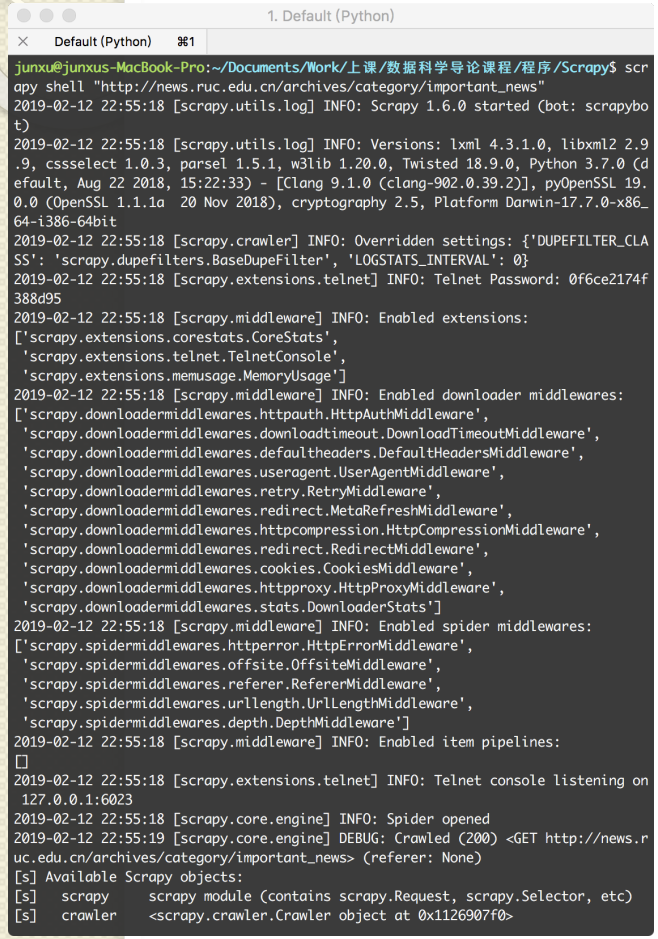


安装

- Windows下：
  - 运行：conda install -c conda-forge scrapy
- 其他：
  - pip install Scrapy

- 参考：
  https://docs.scrapy.org/en/latest/intro/install.html

先试一试(Scrapy shell)




```
>>> response.css('title')
[<Selector xpath='descendant-or-self::title' data='<title>\r\n人大要闻 - 中国人民大学新闻网 | NEWS of RUC<'>]
>>> response.css('title::text').get()
'\r\n人大要闻 - 中国人民大学新闻网 | NEWS of RUC'
>>>
```

```
>>> response.css('a::attr(href)')[1].get()
'http://www.ruc.edu.cn'
>>> response.css('a::attr(href)')[2].get()
'http://portal.ruc.edu.cn'
>>> response.css('a::attr(href)')[3].get()
'mailto:leader@ruc.edu.cn'
>>> response.css('a::attr(href)')[4].get()
'/archives/category/camp_news/view'
>>> response.css('a::attr(href)')[5].get()
'/archives/category/camp_news/department_news'
>>> response.css('a::attr(href)')[6].get()
'/archives/category/camp_news/institute_news'
>>> response.css('a::attr(href)')[7].get()
'/archives/category/important_news/campus'
>>>
```

创建一个爬取项目

- #scrapy startproject rucnews

```
junxu@junxus-MacBook-Pro:~/Documents/Work/上课/数据科学导论课程/程序/Scrapy$ scrapy startproject rucnews
New Scrapy project 'rucnews', using template directory '/usr/local/lib/python3.7/site-packages/scrapy/templates/project', created in:
    /Users/junxu/Documents/Work/上课/数据科学导论课程/程序/Scrapy/rucnews

You can start your first spider with:
    cd rucnews
    scrapy genspider example example.com
junxu@junxus-MacBook-Pro:~/Documents/Work/上课/数据科学导论课程/程序/Scrapy$
```

- 生成如下文件

```
rucnews/
        scrapy.cfg # deploy configuration file
        rucnews/ # project's Python module, you'll import your code from here
        __init__.py items.py # project items definition file
        middlewares.py # project middlewares file
        pipelines.py # project pipelines file
        settings.py # project settings file
        spiders/ # a directory where you'll later put your spiders
                __init__.py
```

撰写爬虫程序

- 在目录rucnews/spiders/目录下创建ruc.py文件

```python
import scrapy


class QuotesSpider(scrapy.Spider):
    name = "ruc"

    def start_requests(self):
        urls = [
            'http://news.ruc.edu.cn/archives/category/important_news/students/page/1',
            'http://news.ruc.edu.cn/archives/category/important_news/students/page/2',
        ]
        for url in urls:
            yield scrapy.Request(url=url, callback=self.parse)

    def parse(self, response):
        page = response.url.split("/")[-2]
        filename = 'rucnews-%s.html' % page
        with open(filename, 'wb') as f:
            f.write(response.body)
        self.log('Saved file %s' % filename)
```

运行爬虫及爬取结果

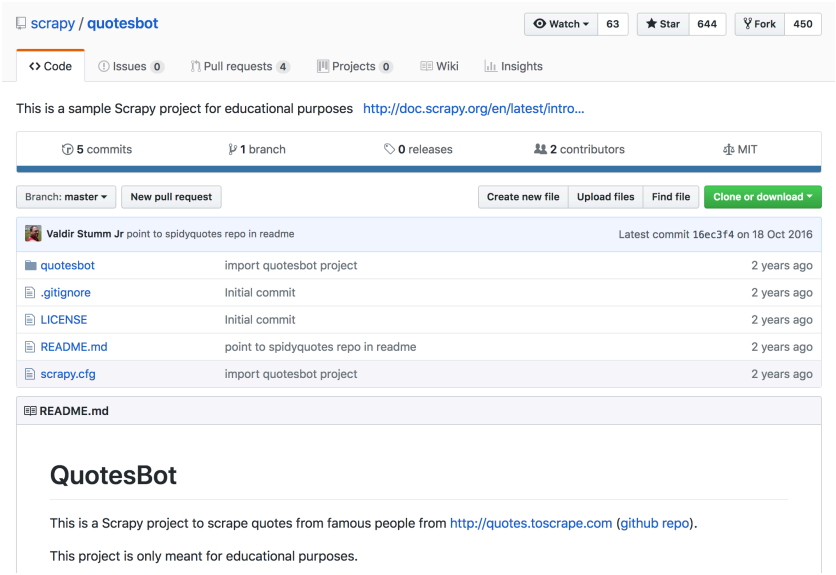
```
junxu@junxus-MacBook-Pro:~/Documents/Work/上课/数据科学导论课程/程序/Scrapy/rucnews$ scrapy crawl ruc
2019-02-26 21:42:39 [scrapy.utils.log] INFO: Scrapy 1.6.0 started (bot: rucnews)
2019-02-26 21:42:39 [scrapy.utils.log] INFO: Versions: lxml 4.3.1.0, libxml2 2.9.9, cssselect 1.0.3, parsel 1.5.1, w3lib 1.20.0, Twisted 18.9.0, Python 3.7.0
(default, Aug 22 2018, 15:22:33) - [Clang 9.1.0 (clang-902.0.39.2)], pyOpenSSL 19.0.0 (OpenSSL 1.1.1a  20 Nov 2018), cryptography 2.5, Platform Darwin-17.7.0-
x86_64-i386-64bit
2019-02-26 21:42:39 [scrapy.crawler] INFO: Overridden settings: {'BOT_NAME': 'rucnews', 'NEWSPIDER_MODULE': 'rucnews.spiders', 'ROBOTSTXT_OBEY': True, 'SPIDER
_MODULES': ['rucnews.spiders']}
2019-02-26 21:42:39 [scrapy.extensions.telnet] INFO: Telnet Password: f4f6875aff1231d0
2019-02-26 21:42:39 [scrapy.middleware] INFO: Enabled extensions:
['scrapy.extensions.corestats.CoreStats',
 'scrapy.extensions.telnet.TelnetConsole',
 'scrapy.extensions.memusage.MemoryUsage',
 'scrapy.extensions.logstats.LogStats']
2019-02-26 21:42:39 [scrapy.middleware] INFO: Enabled downloader middlewares:
['scrapy.downloadermiddlewares.robotstxt.RobotsTxtMiddleware',
 'scrapy.downloadermiddlewares.httpauth.HttpAuthMiddleware',
 'scrapy.downloadermiddlewares.downloadtimeout.DownloadTimeoutMiddleware',
 'scrapy.downloadermiddlewares.defaultheaders.DefaultHeadersMiddleware',
 'scrapy.downloadermiddlewares.useragent.UserAgentMiddleware',
 'scrapy.downloadermiddlewares.retry.RetryMiddleware',
 'scrapy.downloadermiddlewares.redirect.MetaRefreshMiddleware',
 'scrapy.downloadermiddlewares.httpcompression.HttpCompressionMiddleware',
 'scrapy.downloadermiddlewares.redirect.RedirectMiddleware',
 'scrapy.downloadermiddlewares.cookies.CookiesMiddleware',
 'scrapy.downloadermiddlewares.httpproxy.HttpProxyMiddleware',
 'scrapy.downloadermiddlewares.stats.DownloaderStats']
2019-02-26 21:42:39 [scrapy.middleware] INFO: Enabled spider middlewares:
['scrapy.spidermiddlewares.httperror.HttpErrorMiddleware',
 'scrapy.spidermiddlewares.offsite.OffsiteMiddleware',
 'scrapy.spidermiddlewares.referer.RefererMiddleware',
 'scrapy.spidermiddlewares.urllength.UrlLengthMiddleware',
 'scrapy.spidermiddlewares.depth.DepthMiddleware']
2019-02-26 21:42:39 [scrapy.middleware] INFO: Enabled item pipelines:
[]
2019-02-26 21:42:39 [scrapy.core.engine] INFO: Spider opened
2019-02-26 21:42:39 [scrapy.extensions.logstats] INFO: Crawled 0 pages (at 0 pages/min), scraped 0 items (at 0 items/min)
2019-02-26 21:42:39 [scrapy.extensions.telnet] INFO: Telnet console listening on 127.0.0.1:6023
```

▼ rucnews	今天 21:51	--	文件夹
rucnews-students.html	今天 21:51	35 KB	HTML...ocumer
rucnews-2.html	今天 21:51	35 KB	HTML...ocumer
▼ rucnews	今天 21:39	--	文件夹

```
2019-02-26 21:42:40 [scrapy.core.engine] INFO: Closing spider (finished)
```

Scrapy更多资料

- https://docs.scrapy.org/en/latest/
- 教程：http://doc.scrapy.org/en/latest/intro/tutorial.html
- 从案例中学习
  - https://docs.scrapy.org/en/latest/intro/examples.html



Thanks!