慧科
只为赋能人才

第1讲 网络爬虫基础

课程介绍、基本概念、基本原理

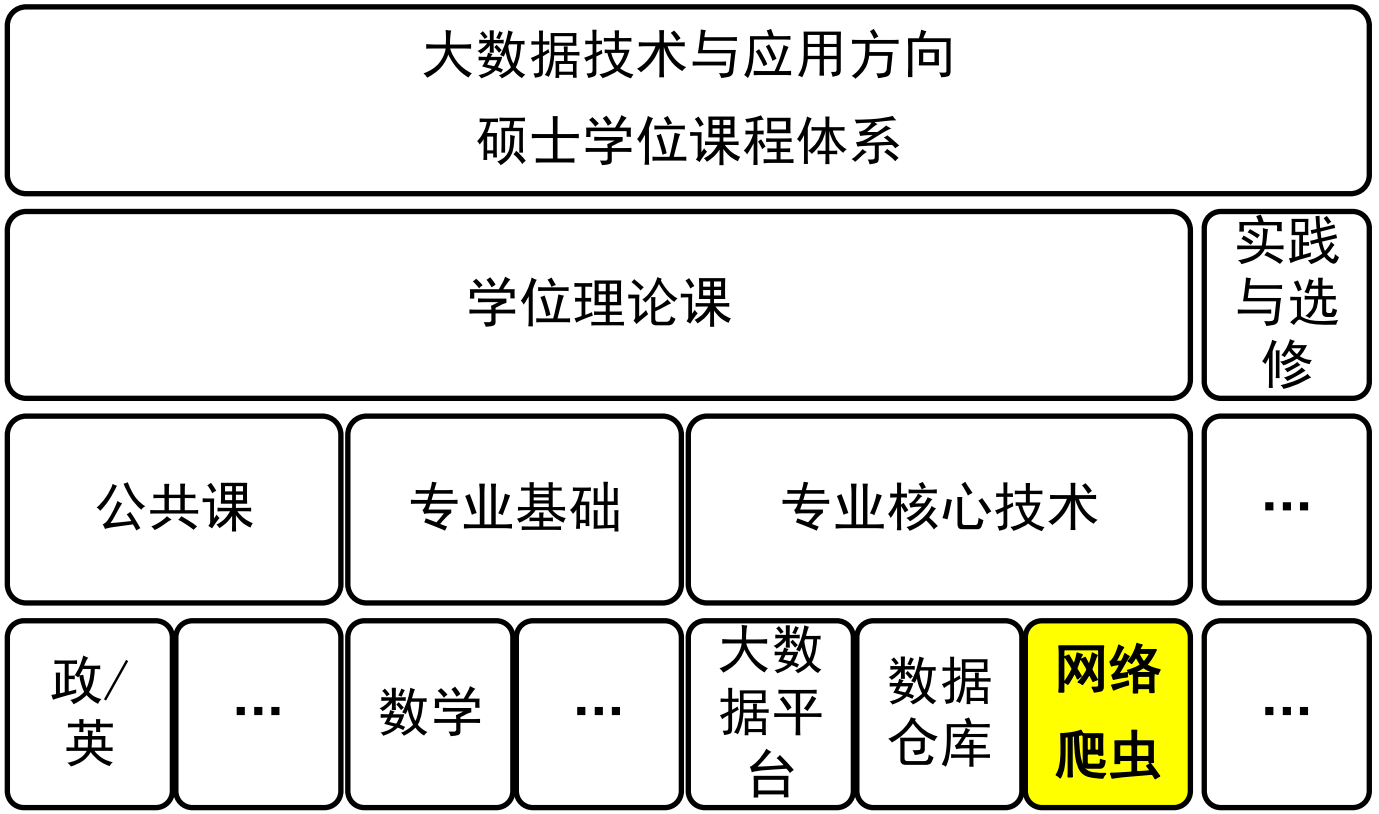
慧科集团 李皓

1

课程简介

课程性质

- “网络爬虫与数据采集”是面向大数据技术与应用方向硕士生的一门专业技术课程



课程目标

了解网络数据获取的基本方式、基本要求和技术要点；

掌握定向网络爬取、内容解析、数据存储的基本能力；

理解website is the API…这一理念。

课程内容

序号	教学内容	基本要求及重点和难点
1	网络爬虫基础	爬虫概念，爬虫的功能与性能需求，爬虫面临的挑战与发展历程。
2	网络爬虫初步实现	模拟用户行为:发起请求，获得响应，定位区域，解析内容，避免陷阱，存储数据的理论和实现。
3	复杂数据提取与整合	复杂异构数据源的信息提取与整合，掌握应对一般反爬虫策略的方法。
4	高性能分布式爬虫原理与实践	设计分布式爬虫，设计实现并发、非阻塞I/O的网络爬取系统。
5	数据采集与清洗技术	数据预处理的基本思路，数据清洗的基本方法

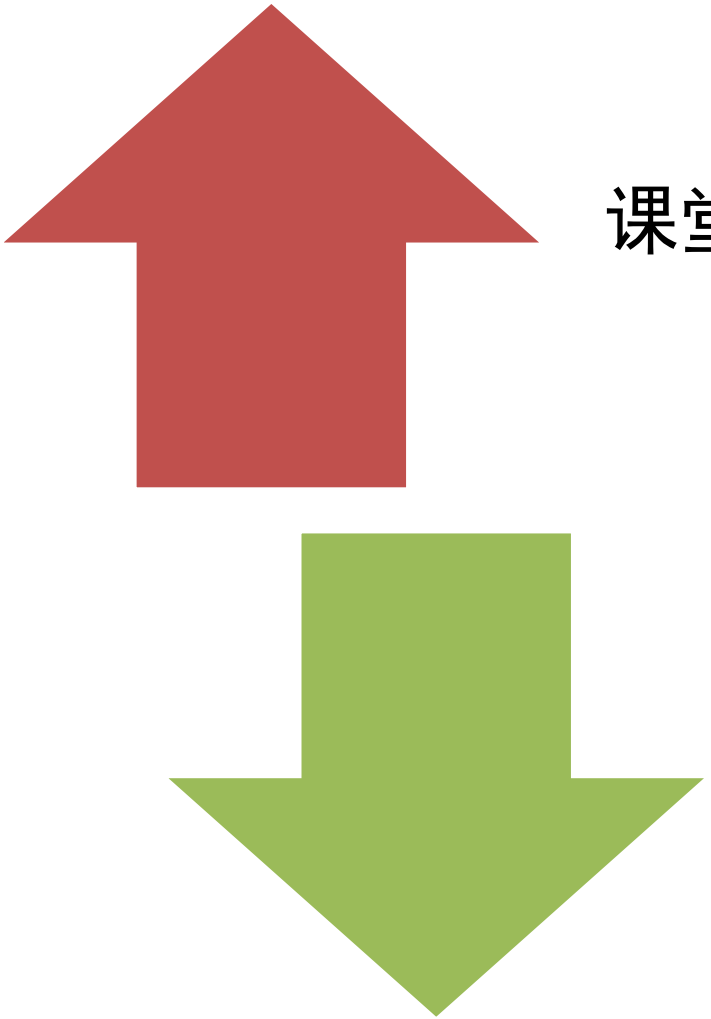
课次安排

讲次	主题	理论课时	实践课时
1	网络爬虫概念、原理、挑战	2	1
2	基础Web页面获取技术	1	2
3	Web页面内容的解析与选择提取	1	2
4	爬取数据存储	1.5	1.5
5	完整爬虫系统的组成与构建	1	2
6	浏览器操作模拟与Ajax抓取	1	2
7	分布式爬虫的设计与实现	1	2
8	爬虫系统性能调优	2	1
9	数据清洗过程与实践	1	2
10	自主项目实践	0	3
11	结课考试	2	0
小结	32学时	14	18

参考资料

1. 崔庆才，《Python 3网络爬虫开发实战》，人民邮电出版社，2018.4.
2. 迪米特里奥斯 考奇斯-劳卡斯 《精通Python爬虫框架Scrapy》，人民邮电出版社，2018.2.
3. Cho, Junghoo, "Web Crawling Project", UCLA Computer Science Department.
4. A History of Search Engines, from Wiley
5. A tutorial for creating basic crawlers.
6. WIVET is a benchmarking project by OWASP, which aims to measure if a web crawler can identify all the hyperlinks in a target website.
7. Shestakov, Denis, "Current Challenges in Web Crawling" and "Intelligent Web Crawling", slides for tutorials given at ICWE'13 and WI-IAT'13.
8. Guide: How to Make a Web Crawler in Under 50 Lines of Code (Python) – Saint
9. https://en.wikipedia.org/wiki/Web_crawler中的References papers
10. 中国大学mooc
11. github

考核方法



课堂出勤与表现（占30%）

期末试卷成绩（70%）

2

网络爬虫基本概念

为什么要学大数据?

- 2010年前后，以云计算、大数据、物联网的普及为标志迎来了第三次信息化浪潮。

为什么要学大数据?

- 成为数据科学家 or 数据工程师

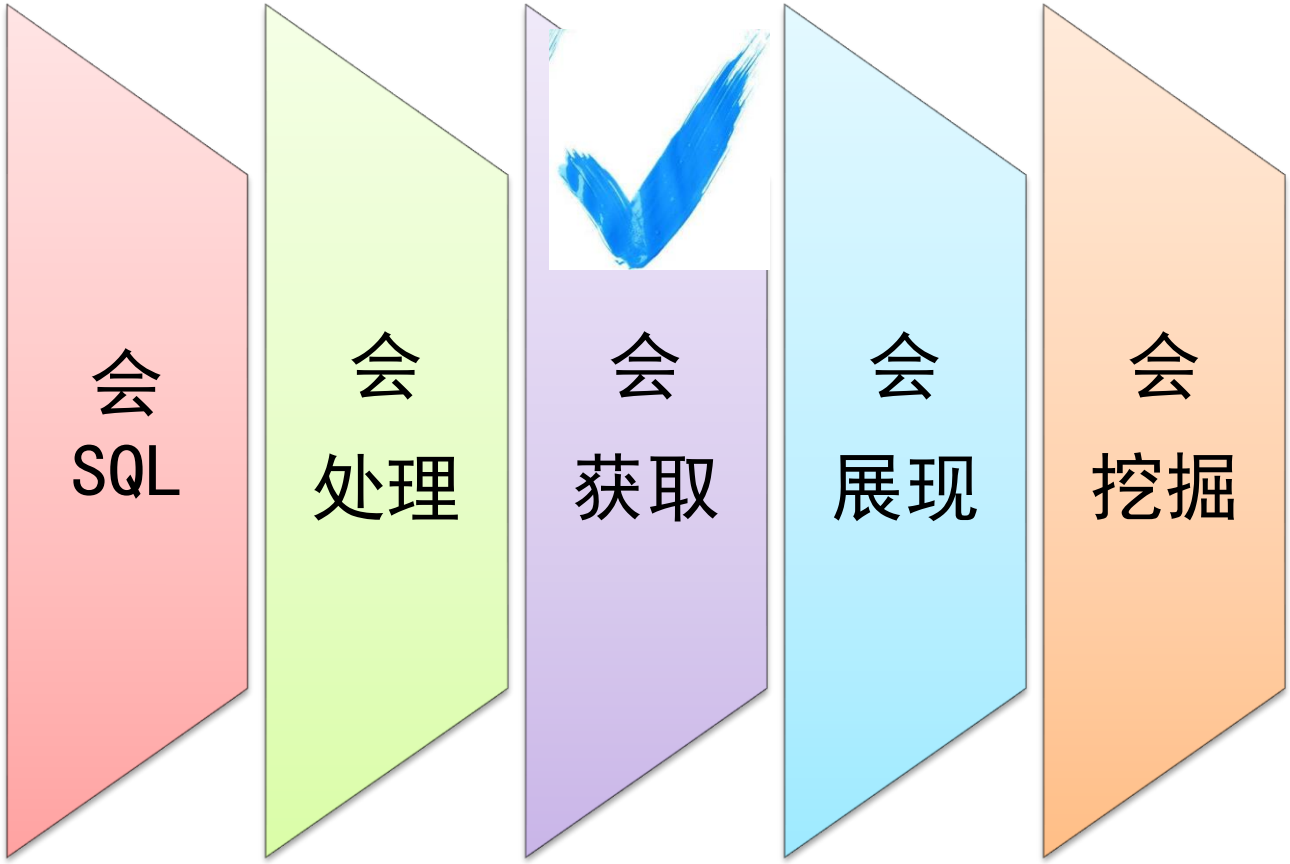
Data Engineer VS Data Scientist

Make practical things Explain why

数据科学的业务流程

出口导向

- 企业对数据科学岗位的基础技能需求



数据从何而来?

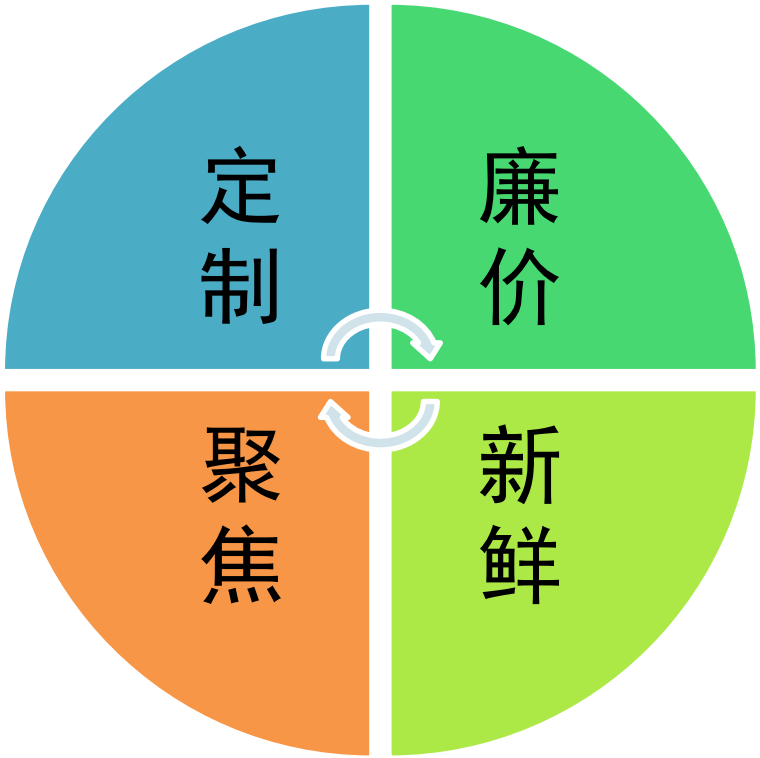
1. 单位自有历史数据
   - 文档、数据库、表格
2. 定量/定性的市场调研
   - 设计调查问卷
   - 网络、街头、电话
3. 专业机构的长期积累
   - 百度指数、国家统计局、麦肯锡、埃森哲

数据从何而来?

- 专业调研机构
  - 互联网企业
    - 百度指数、阿里指数、TBI腾讯浏览指数、新浪微博指数
  - 数据交易平台
    - 数据堂、国云数据市场、贵阳大数据交易平台
  - 政府/机构
    - 中华人民共和国国家统计局数据
    - 世界银行公开数据
    - 联合国数据
    - 纳斯达克
  - 数据管理咨询公司
    - 麦肯锡、埃森哲、艾瑞咨询

数据从何而来?

- 4. 利用网络爬虫
  - 自动动手、丰衣足食

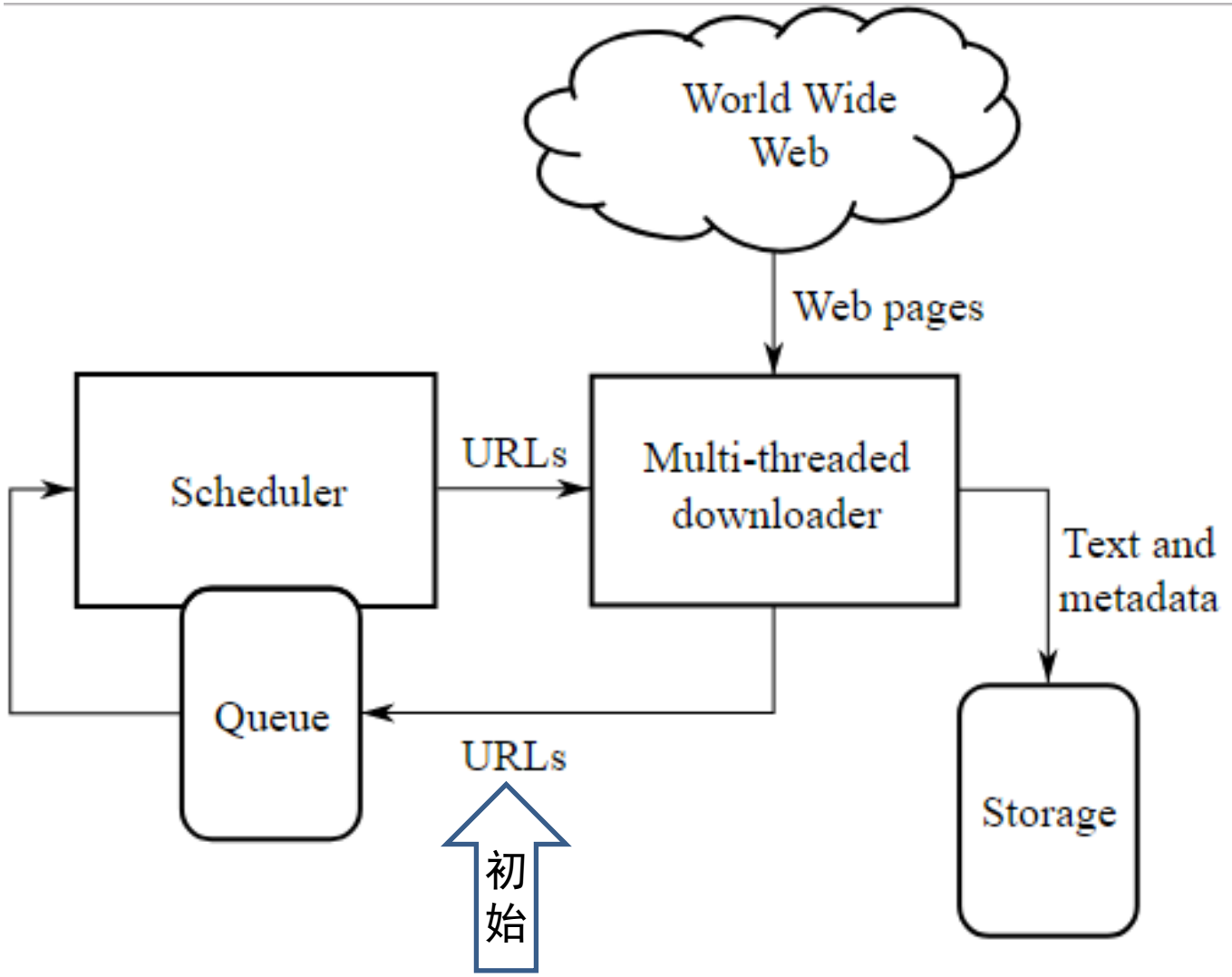


网络爬虫的定义

- 一种自动抓取网页并提取网页内容的程序。

- Many names：
  - Crawler
  - Spider
  - Robot (or bot)
  - Web agent
  - Wanderer, worm, …
  - And famous instances: googlebot, scooter, slurp, msnbot, …

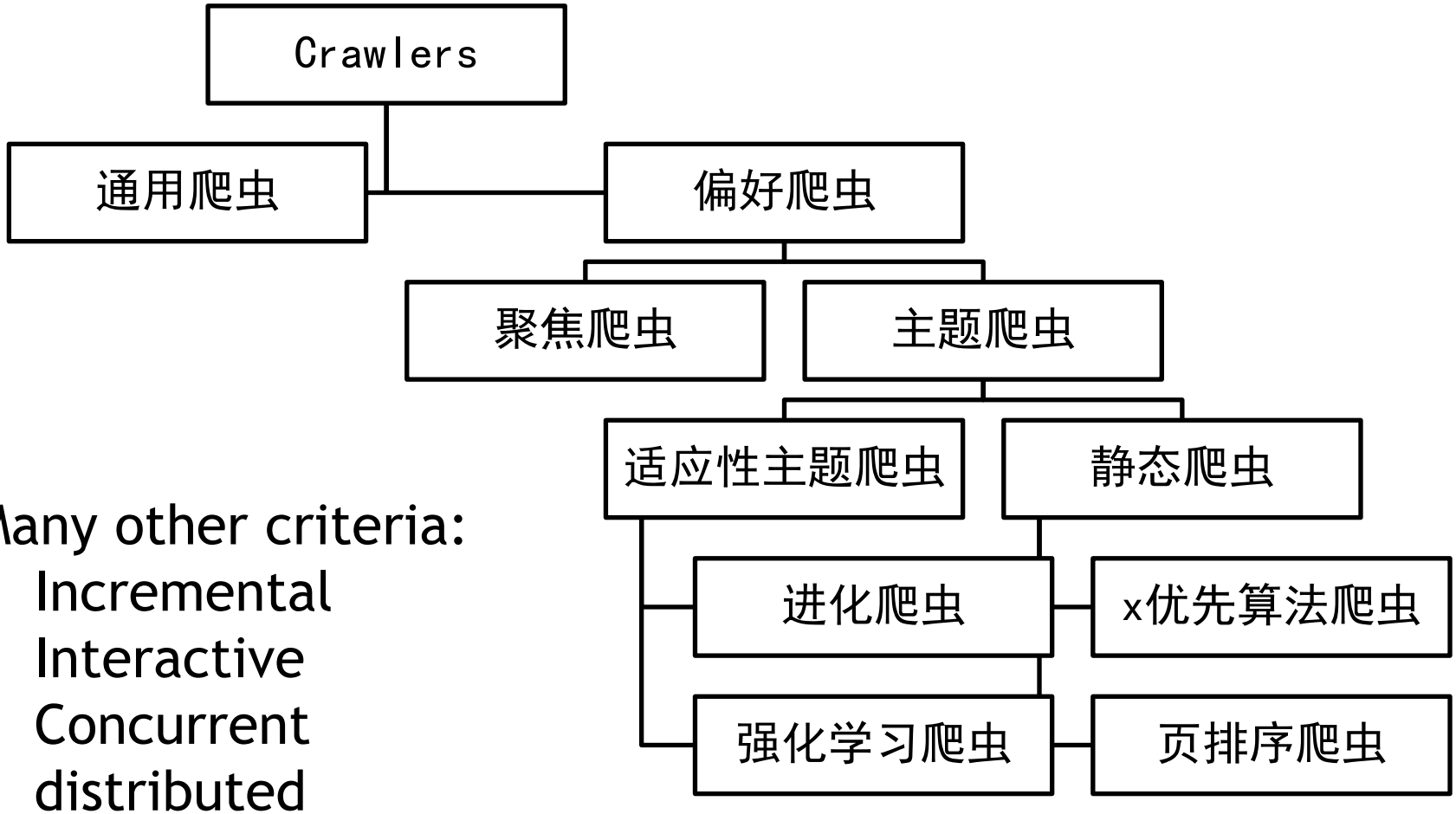
网络爬虫的工作原理


World Wide Web
Web pages
Scheduler
URLs
Multi-threaded downloader
Text and metadata
Queue
URLs
Storage
初始

著名的网络爬虫（wiki）

- Apache Camel
- archive.is
- Automation Anywhere
- Convertigo
- cURL
- Data Toolbar
- Diffbot
- Firebug
- Greasemonkey
- Heritrix
- HtmlUnit
- HTTrack
- iMacros
- Import.io
- Jaxer
- Node.js
- nokogiri
- PhantomJS
- ScraperWiki
- Scrapy
- Selenium
- SimpleTest
- UiPath
- watir
- Wget
- Wireshark
- WSO2 Mashup Server
- Yahoo! Query Language(YQL)

网络爬虫的分类（taxonomy）



Many other criteria:

- Incremental
- Interactive
- Concurrent
- distributed

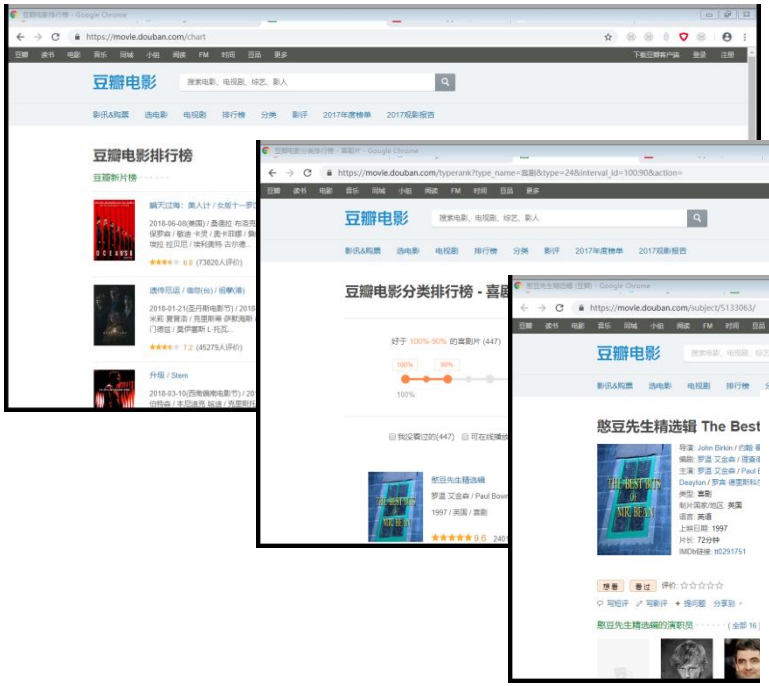
网络爬虫的功能需求

1. 检索网页，获取网页内容
   - 例如：search engine都有crawler
2. 自动化收集主题/聚焦信息
3. 自动化web应用测试或模型检测
4. 自动化安全测试和漏洞检测

网络爬虫的性能需求

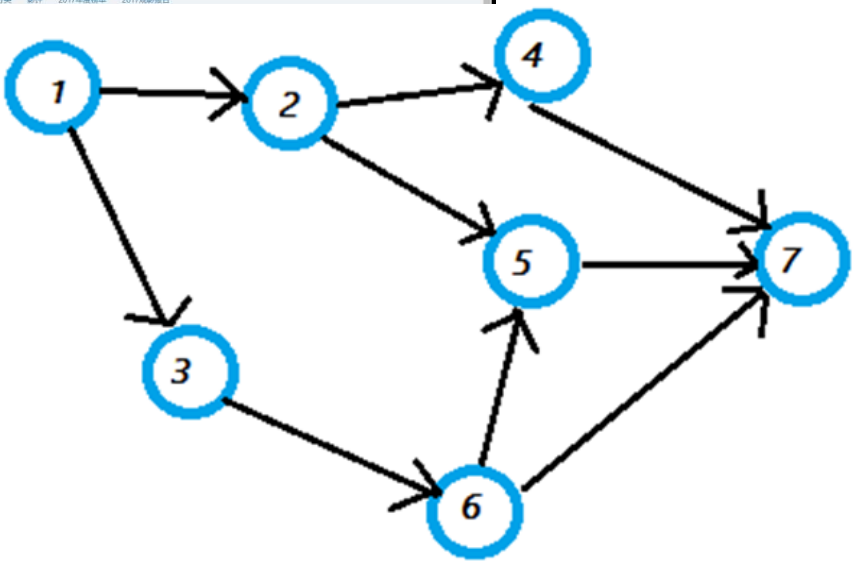
- 完整性Coverage

- 如何爬取众多URLs，提高coverage?



https://movie.douban.com/

https://movie.douban.com/chart



https://movie.douban.com/typerank

=

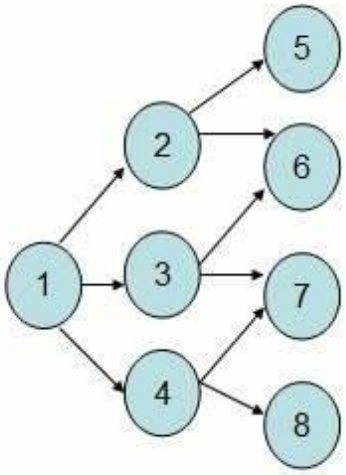
24&interval_id=100:90&action=

https://movie.douban.com/subject/

https://movie.douban.com/tag/%E5

如何爬取众多URLs，提高coverage?

- 广度优先



网页间关系的图模型

网页节点

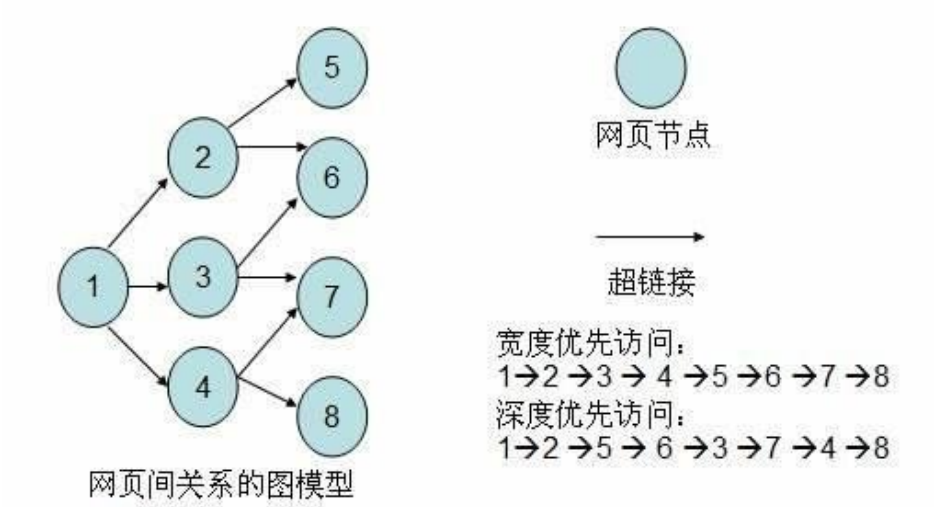
超链接

宽度优先访问：
1→2 →3 → 4 →5 →6 →7 →8
深度优先访问：
1→2 →5 → 6 →3 →7 →4 →8

1. 首先将根节点放入队列中。
2. 从队列中取出第一个节点，并检查它。
   - 如果有所需内容，则返回结果。
   - 将它所有尚未检验过的直接子节点加入队列中。
3. 若队列为空，表示整张图都检查过了，则结束。
4. 重复步骤2。

如何爬取众多URLs，提高coverage?

- 深度优先



网页间关系的图模型

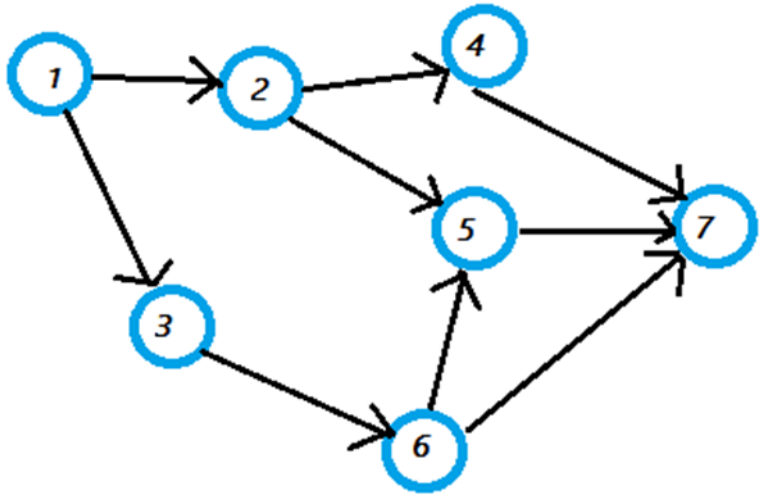
1. 首先将根节点放入队列中。
2. 从队列中取出第一个节点，并检查它。
   - 若有所需内容，则返回结果。
   - 将它某一个尚未检验过的直接子节点加入队列中。
3. 重复步骤2。
4. 如果不存在未检查过的直接子节点。
   - 将上一级节点加入队列中，之后重复步骤2。
5. 重复步骤4。
6. 若队列为空，表示整张图都检查过了，结束。

如何爬取众多URLs，提高coverage?

- 广度优先
  - 层次遍历，完全遍历
  - 在队列中需保留全部URLs，占用空间多
  - 无回溯操作，速度快。
  - 有最佳解
- 深度优先
  - 深入分支，完全遍历
  - 不需要存所有URLs，空间占用少
  - 有回溯，运行速度慢
  - 无最佳解

网络爬虫的性能需求

- 覆盖度Coverage
- 新鲜度Freshness
- 礼貌性Politeness
  - 防止造成DoS Attack
- 绕过陷阱Avoid traps
  - 利用Black-lists
- 扩展性Scalablility
  - Webpage指数级增长；web技术不断演变
- 时效性Timeliness
- 提交效率Submission efficiency
  - 成功提交表单的比例
- 状态敏感度State sensitivity
  - 反映正确感知DOM状态等页面元素的改变的能力



网络爬虫的发展历程

1. 静态 WEB 爬取
2. 动态 WEB 爬取
   - ASP/PHP/JSP 表单 数据库
3. RIA WEB 爬取

   Flash\SilverLight\curl\javaFX\ActiveX\HTML5

网络爬虫的发展历程

类型	输入	图组件
Static web crawler	URLs seed set	Nodes: different URLs; Directed Edges: hyperlinks
Dynamic web crawler	URLs seed set User context data Domain taxonomy	Nodes: pages Directed Edges: forms
RIA web crawler	URL of index.html	Nodes: DOM states Directed Edges: javascript actions(引起DOM状态改变且爬虫敏感的)

网络爬虫的发展历程

- 静态web爬取过程

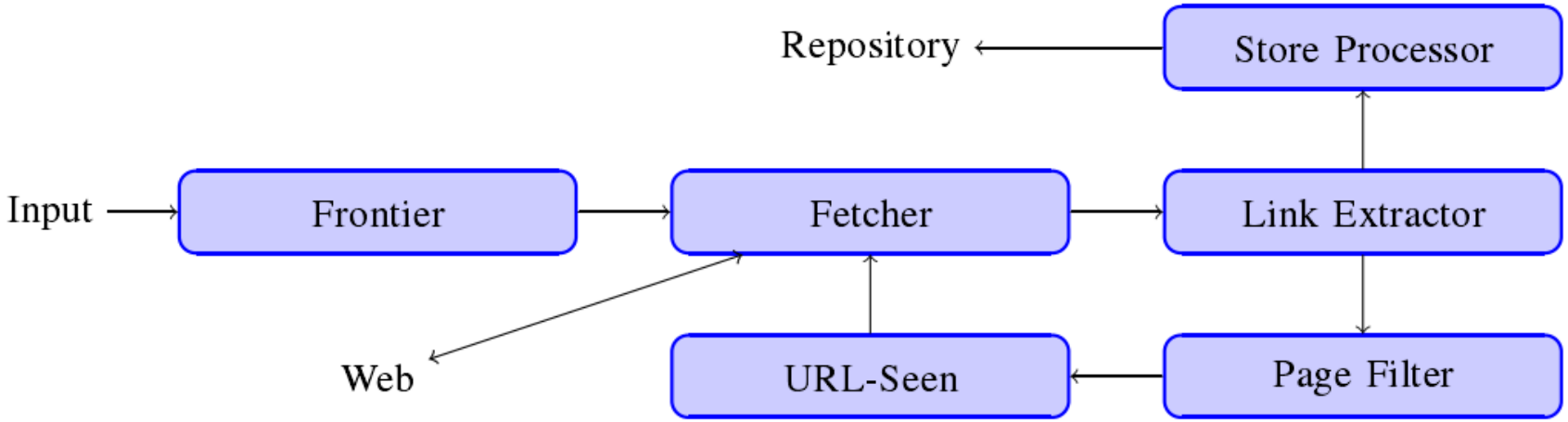


Figure 1: Architecture of a Traditional Web Crawler.

网络爬虫的发展历程

- 静态web爬虫的演进

相关研究	特色组件	方法	目标
WebCrawler(1993) MOM spider	Fetcher, Frontier, Page filter	Parallel downloading of 15 links. robots.txt, blacklist	Scalability Politeness
Google(1998)	Store Processor, Frontier	Reduce disk access time by compress, Pagerank.	Scalability Coverage Freshness
Mercator(1999)	URL-Seen	Batch disk checks, cache	Scalability
IBM Web Fountain(2001)	Storage processor, Frontier, Fetch	Local copy of the fetched pages, Adaptive download rate, Homogeous cluster as hardware.	Scalability Coverage Freshness
Polybot(2002)	URL-Seen	Red-Black tree to keep the seen URLs	Scalability

网络爬虫的发展历程

- 静态web爬虫的演进

相关研究	特色组件	方法	目标
UbiCrawler (2002)	URL-Seen	P2P architecture	Scalability
pSearch (2002)	Storage processor	Distributed Hashing Tables (DHT)	Scalability
Exposto et.al. (2002)	Frontier	DHT	Scalability
IRLbotpages (2002)	URL-Seen	Access time reduction by disk segment.	Scalability

网络爬虫的发展历程

- 动态web爬取过程

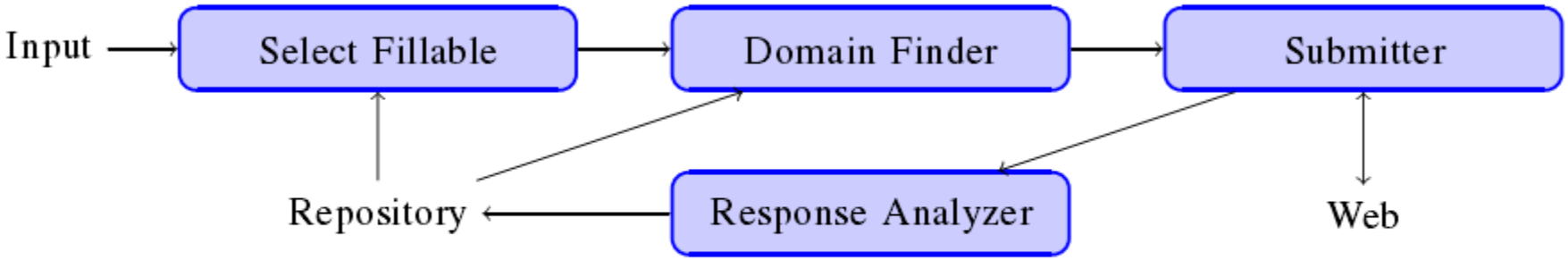


Figure 2: Architecture of a Deep Web Crawler.

网络爬虫的发展历程

- 动态web爬虫的演进

相关研究	特色组件	方法	目标
HiWe（2001）	Select fillable, Domain Finder, Submiter, Response Analyst	使用词干提取等方法发现表单，忽略不完整表单，手工登录	Submission efficiency
Liddle et al.(2003)	Select fillable Domain finder	仅处理有限值集的字段，忽略文本字段的自动填充；通过hash检查重复语句	Submission efficiency
Barbosa（2004）	Select fillable, Domain Finder, Response Analyst	使用贪婪算法寻找最多内容且最小查询的页面，增加停止词来最大化完整性…	Submission efficiency

网络爬虫的发展历程

- 动态web爬虫的演进

相关研究	特色组件	方法	目标
Ntoulas et.al. (2005)	Select fillable, Domain Finder	基于单一词干关键字的查询，采用三种策略（随机、词频、适应性）学习已下载页面内容，使每个查询的唯一返回内容最大化	Submission efficiency
Lu et.al. (2008)	Select fillable, Domain Finder	查询文本数据源，对原数据源进行采样，完成表单提交，最大化爬取完整性	Submission efficiency

网络爬虫的发展历程

- RIAs web爬取过程

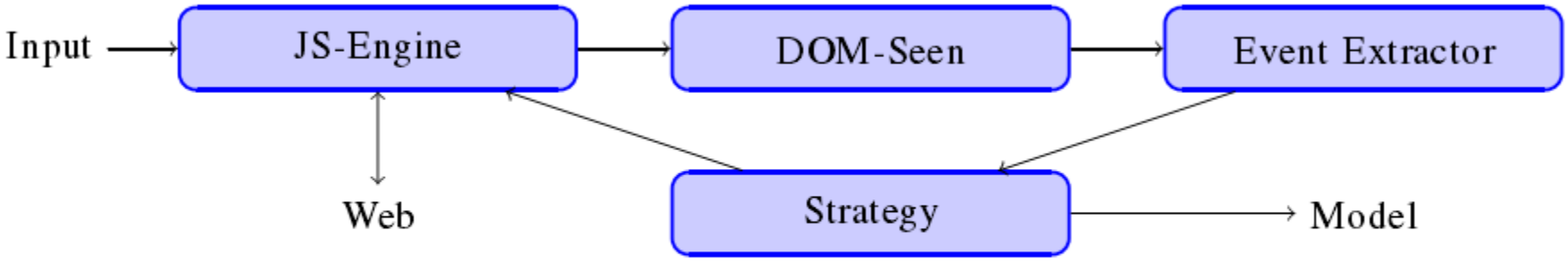


Figure 3: Architecture of a RIA Web Crawler.

网络爬虫的发展历程

- RIAs web爬虫演进

相关研究	特色组件	方法	目标
Duda et.al. (2007)	Strategy, JS Engine, DOM-Seen	BFS；缓存javascript函数调用与结果；Hash 完整的序列化DOM以判断是否出现过。	Completeness, Efficiency
Meshbah et.al. (2008)	Strategy, DOM-Seen	DFS；仅探索单一js事件，通过计算编辑距离来判断是否有状态变化。	Completeness, Efficiency, scalability
CrawlRIA (2010)	Strategy, DOM-Seen	DFS；自动生成执行跟踪；通过元素、事件类型、事件句柄的集合比较两个DOM是否一致	state coverage efficiency
Kamara et.al. (2010)	Strategy	设web应用为超立方模型，利用最小链分解和最小过渡覆盖方法检查和过滤DOM状态变化。	State coverage efficiency

网络爬虫的发展历程

- RIAs web爬虫演进

相关研究	特色组件	方法	目标
AjaxRank (2007)	Strategy DOM-Seen	URL的初始状态更重要，类似于PageRank，基于连接性，但是转换不是超链接，而是DOM的内容和结构的哈希值.	State coverage efficiency
Djincturk et al (2010)	Strategy	通过事件发现新状态的概率和遵循事件状态路径的代价,判断状态改变。	State coverage efficiency
Dist RIA Crawler (2013)	Strategy	使用JavaScript事件来划分搜索区域，运行多个爬虫进行并行爬取。	scalability
Feedex (2013)	Strategy	基于事件对DOM的可能影响，对事件进行优先级排序，考虑代码覆盖率、导航和页面结构多样性等因素，判断DOM状态改变	State coverage efficiency

网络爬虫的发展历程

- Open questions in web crawling
  - 静态、动态web爬取问题基本解决，RIAs爬取问题仍然很多…
  - Model based crawling
    - 爬取多个DOM状态，可抽象为有向图遍历，即非对称旅行商问题（ATSP）,NP-hard。
  - Scalability
    - RIAs中存在类似URL-Seen的问题，即State-Seen
  - Widget detect
    - 为了避免状态爆炸，有必要检查RIA应用中每个接口/界面的独立部分，这还可能影响不同状态的打分；
  - Definition of state in RIA
    - 一些HTML5元素（web sockets双向通信）带来了新挑战；
  - 多类型爬虫的组合

3

网络爬虫的实现思路

如何写出自己的网络爬虫?

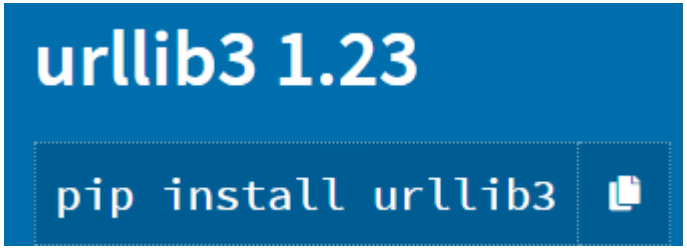
- Step 1. 了解WWW技术与web应用:
  - Hypertext Transfer Protocol
  - Robots exclusion standard
  - URL spec (RFC 3986)
  - Document Object Model
  - Daemons
  - User-Agent
  - rel=”nofollow”
- 扩展学习
  - An Extended Standard for Robot Exclusion
  - Sitemaps
  - Making AJAX Application Crawlable

如何写出自己的网络爬虫?

- Step 2. 掌握获取web内容的程序化方法:
  - PHP
    - 对多线程、异步支持不好
    - 并发能力弱，效率不高
  - JAVA
    - Python爬虫最大的对手，生态圈、支持度完善
    - 开发代码量大（重复性、不精炼、不易维护，重构成本高）
  - C/C++
    - 性能很好
    - 学习成本高，代码成型慢
  - Python
    - 语法优美、代码简洁、开发效率高、胶水语言
    - 爬虫所需模块丰富、
    - 爬虫框架成熟、高效分布式策略

Why python?

- Python世界对网络信息爬取提供的支持：



https://github.com/BruceDone/awesome-crawler……

Why python?

awesome-crawler

- Scrapy - A fast high-level screen scraping and web crawling framework.
  - django-dynamic-scraper - Creating Scrapy scrapers via the Django admin interface.
  - Scrapy-Redis - Redis-based components for Scrapy.
  - scrapy-cluster - Uses Redis and Kafka to create a distributed on demand scraping cluster.
  - distribute_crawler - Uses scrapy,redis, mongodb,graphite to create a distributed spider.
- pyspider - A powerful spider system.
- cola - A distributed crawling framework.
- Demiurge - PyQuery-based scraping micro-framework.
- Scrapely - A pure-python HTML screen-scraping library.
- feedparser - Universal feed parser.
- you-get - Dumb downloader that scrapes the web.
- Grab - Site scraping framework.
- MechanicalSoup - A Python library for automating interaction with websites.
- portia - Visual scraping for Scrapy.
- crawley - Pythonic Crawling / Scraping Framework based on Non Blocking I/O operations.
- RoboBrowser - A simple, Pythonic library for browsing the web without a standalone web browser.
- MSpider - A simple ,easy spider using gevent and js render.
- brownant - A lightweight web data extracting framework.
- PSpider - A simple spider frame in Python3.
- Gain - Web crawling framework based on asyncio for everyone.
- sukhoi - Minimalist and powerful Web Crawler.
- spidy - The simple, easy to use command line web crawler.
- newspaper - News, full-text, and article metadata extraction in Python 3
- aspider - An async web scraping micro-framework based on asyncio.

如何写出自己的网络爬虫?

- Step 3. 找好目标，由简入深，开展实践
  - 简单爬虫
  - 设立存储数据库与存储模式
  - 设定遍历策略，防止URL-Seen
  - 处理表单
  - 使用代理
  - 处理Ajax
  - 处理验证码
  - 分布式
  - 大并发
- Step 4. 优化程序，建立模型，爬向深网

我们对工具的选择

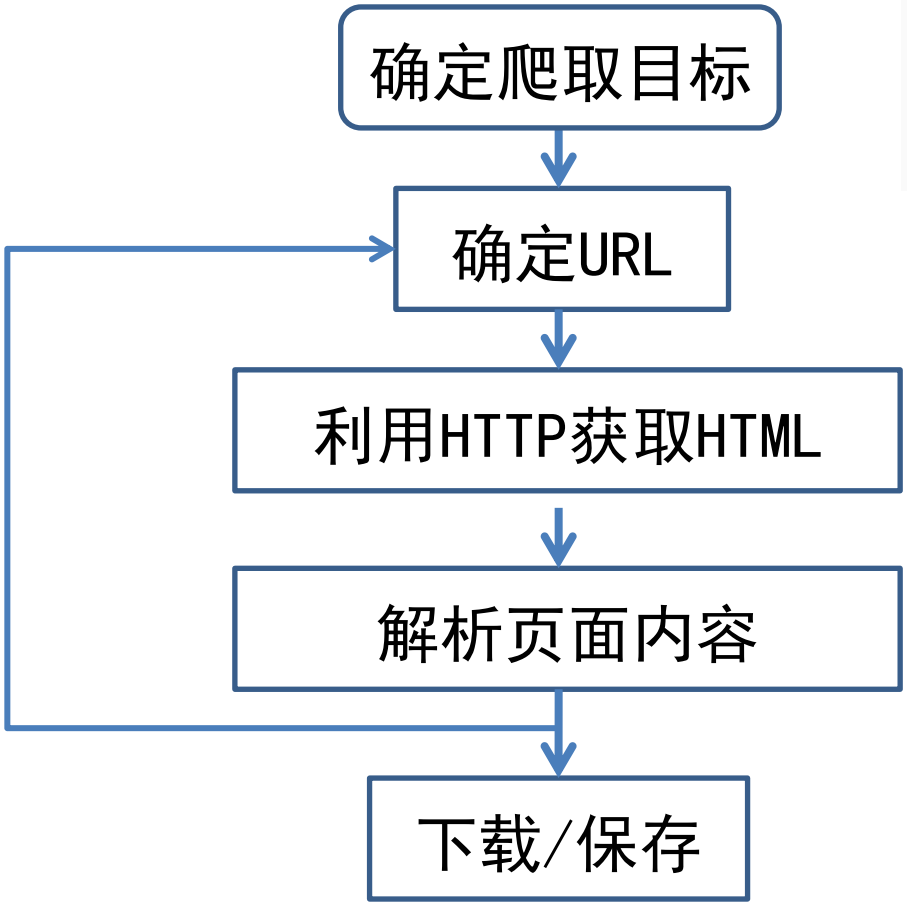
- Python3.5或以上
- Jupyter 脚本编写试用程序
- Pycharm IDE 做完整案例
- Python库或框架
  - Urllib
  - Requests
  - Beautifulsoup4
  - Scrapy
  - Scrapy-redis
  - Selenium
  - …

4

网络爬虫的简单实现

网络爬虫的简单实现

- 基本思路



http://httpbin.org/

网络爬虫的简单实现

- URL
- HTTP
- HTML
- 其他

获取网页的基本要素

网络爬虫实现基础

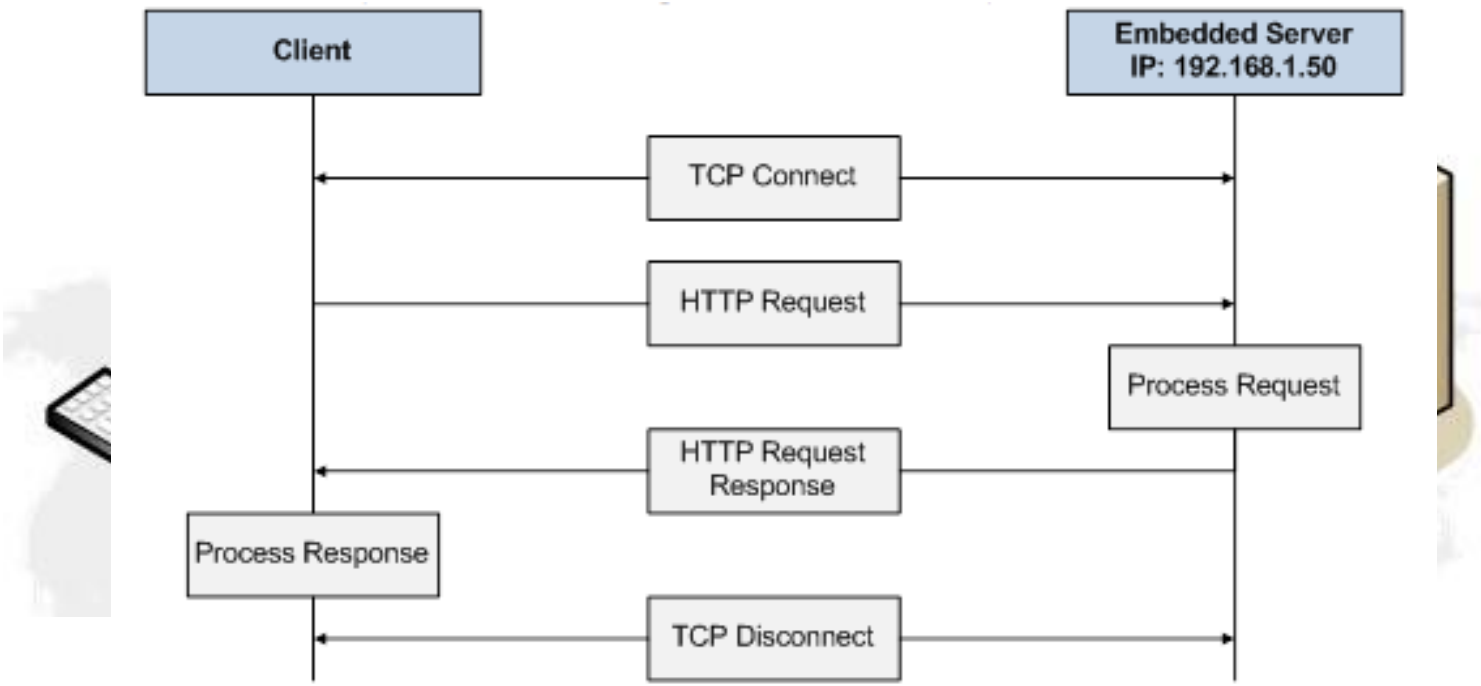
- URL（Uniform Resource Locator）
  - 是对可以从互联网上的资源的位置和访问方法的一种简洁的表示，是互联网上标准资源的地址。

协议	主机域名/IP地址	:端口	/路径

http://www.baidu.com
ftp://book:book@down.pcbookcn.com/
http://localhost:8080/baj/jjs/1.txt

网络爬虫实现基础

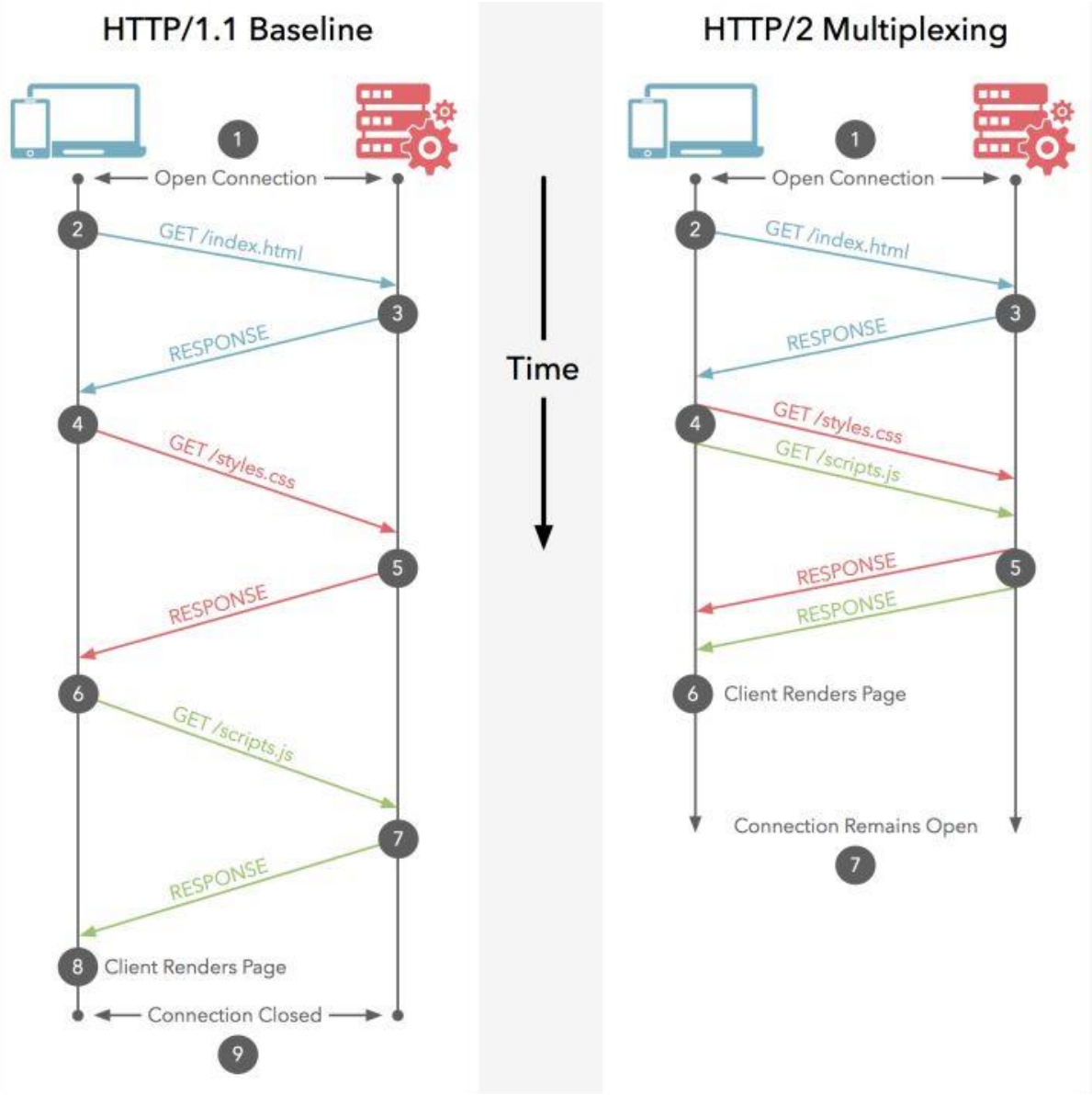
- HTTP（HyperText Transfer Protocol）
  - 一种发布和接收 HTML页面的方法。默认端口80。
  - 由 **RFC 2616**定义，是一个无状态的应用层协议。
- HTTP通信过程



网络爬虫实现基础

- HTTP 1.0/1.1/2.0的区别
  - HTTP1.0是没有host域的，HTTP1.1才支持这个参数。
  - HTTP1.1
    - 默认使用长连接，可有效减少TCP的三次握手开销。
    - 支持只发送header信息(不带任何body信息), 节约了带宽。
    - 支持传送PATCH方法（提交内容的一部分），这是支持文件断点续传的基础。
  - HTTP 2.0
    - 使用多路复用技术(Multiplexing), 允许同时通过单一的 连接发起多重的请求-响应消息。
    - Multiplexing的二进制分帧技术使多个请求都在同一个TCP连接上完成，可以承载任意数量的双向数据流。
    - 新增首部压缩（Header Compression）:采用HPACK算法
    - 新增服务端推送（Header Compression）

网络爬虫实现基础


HTTP/1.1 Baseline
1
Open Connection
2
GET /index.html
3
RESPONSE
4
GET /styles.css
5
RESPONSE
6
GET /scripts.js
7
RESPONSE
8
Client Renders Page
Connection Closed
9
Time
HTTP/2 Multiplexing
1
Open Connection
2
GET /index.html
3
RESPONSE
4
GET /styles.css
GET /scripts.js
RESPONSE
RESPONSE
5
6
Client Renders Page
Connection Remains Open
7

网络爬虫实现基础

- HTTP 请求头/响应头
  - Accept
    - 声明浏览器端可以接收的媒体类型
    - 例如：Accept: text/html,application/xhtml+xml
  - Accept-Encoding
    - 浏览器声明自己接收的编码方法，通常指定压缩方法
    - Accept-Encoding: gzip, deflate, br
  - Accept-Language
    - 浏览器声明自己接收的语言
    - Accept-Language：zh-CN,zh;q=0.9
  - Connection
    - 声明TCP 连接类型情况
    - Connection: keep-alive

网络爬虫实现基础

- HTTP 请求头/响应头
  - Host(request时 必须有)
    - 用于指定被请求资源的Internet主机和端口号，通常从HTTP URL中提取。
  - Referer
    - 显示客户端从何处链接到当前server/页面的。
  - User-Agent
    - 告知server当前用户使用的浏览器或os信息
  - Cache-Control
    - 指明当前资源的有效期，指导浏览器从缓存取内容或重新请求页面。
  - Upgrade-insecure-requests：1
    - 申明浏览器支持从 http 请求自动升级为 https 请求

网络爬虫实现基础

- HTTP 请求头/响应头
  - Accpet-ranges
    - 表明自己是否接收获取某个实体的一部分
  - If-Modified-Since
    - 浏览器端缓存页面的最后修改时间，若在server端该页面的修改时间与之相同，则返回304，客户端使用本地缓存；否则返回200和新的文件内容，浏览器缓存新文件删除旧文件。
  - If-None-Match
    - If-None-Match和ETag一起工作，Etag记录在响应头中。服务器验证两者是否相等，若相等返回304，可用缓存文件；否则返回200状态和新的资源和Etag。
  - DNT(Do not track)
    - 0表示用户乐于被server跟踪.1表示用户不希望server跟踪。

网络爬虫实现基础

- HTTP方法
  - 在客户机和服务器之间进行请求-响应时，两种最常被用到的方法是：GET 和 POST。
    - **GET** - 从指定的资源请求数据。
    - **POST** - 向指定的资源提交要被处理的数据

网络爬虫实现基础

- HTTP方法
  - **GET** - 从指定的资源请求数据。
    - 通过URL发送附加参数，例如：/test/demo_form.asp?name1=value1&name2=value2
  - GET的特点
    - GET 请求可被browser缓存
    - GET 请求保留在浏览器历史记录中
    - GET 请求可被收藏为书签
    - GET 请求不应在处理敏感数据时使用
    - GET 请求有长度限制（浏览器限制，非http协议限制）
    - GET 请求只应当用于取回数据

网络爬虫实现基础

- HTTP 方法
  - **POST** - 向指定的资源提交要被处理的数据
    - 通过HTTP消息主体（请求头部）发送数据，例如：
    - POST /test/demo_form.asp HTTP/1.1
    - Host: w3schools.com
    - name1=value1&name2=value2
  - POST 的特点：
    - POST 请求不会被浏览器缓存
    - POST 请求不会保留在浏览器历史记录中
    - POST 不能被收藏为书签
    - POST 请求对数据长度没有要求，但server的处理能力事实上有限。

网络爬虫实现基础

	GET	POST
后退按钮/刷新	无害	数据会被重新提交（浏览器应该告知用户数据会被重新提交）。
书签	可收藏为书签	不可收藏为书签
缓存	能被缓存	不能缓存
编码类型	application/x-www-form-urlencoded	application/x-www-form-urlencoded 或 multipart/form-data。为二进制数据使用多重编码。
历史	参数保留在浏览器历史中。	参数不会保存在浏览器历史中。
对数据长度的限制	是的。当发送数据时，GET 方法向 URL 添加数据；URL 的长度是受限制的（URL 的最大长度是 2048 个字符）。	无限制。
对数据类型的限制	只允许 ASCII 字符。	没有限制。也允许二进制数据。
安全性	与 POST 相比，GET 的安全性较差，因为所发送的数据是 URL 的一部分。在发送密码或其他敏感信息时绝不要使用 GET ！	POST 比 GET 更安全，因为参数不会被保存在浏览器历史或 web 服务器日志中。
可见性	数据在 URL 中对所有人都是可见的。	数据不会显示在 URL 中。

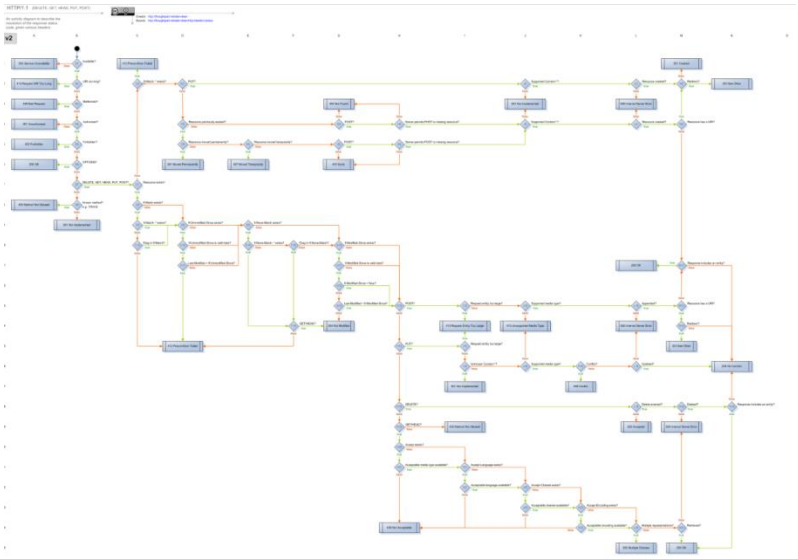
网络爬虫实现基础

其他 HTTP 请求方法：

- HEAD
  - 与 GET 相同，但只返回 HTTP 报头，不返回文档主体。
- PUT
  - 上传指定的 URI 表示。
- DELETE
  - 删除指定资源。
- OPTIONS
  - 返回服务器支持的 HTTP 方法。
- PATCH（HTTP/1.1）
  - 部分提交/修改内容。
- CONNECT（HTTP/1.1）
  - 把请求连接转换到透明的 TCP/IP 通道。
- TRACE（HTTP/1.1）
  - 回显服务器收到的请求，主要用于测试或诊断

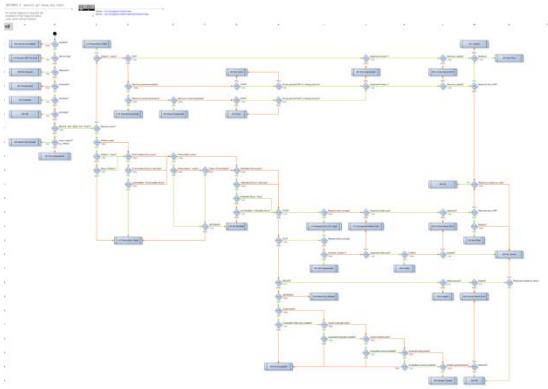
网络爬虫实现基础

- HTTP 响应码（Response code）
  - 1xx 消息
  - 2xx 成功
  - 3xx 重定向
  - 4xx 请求错误
  - 5xx 服务器错误



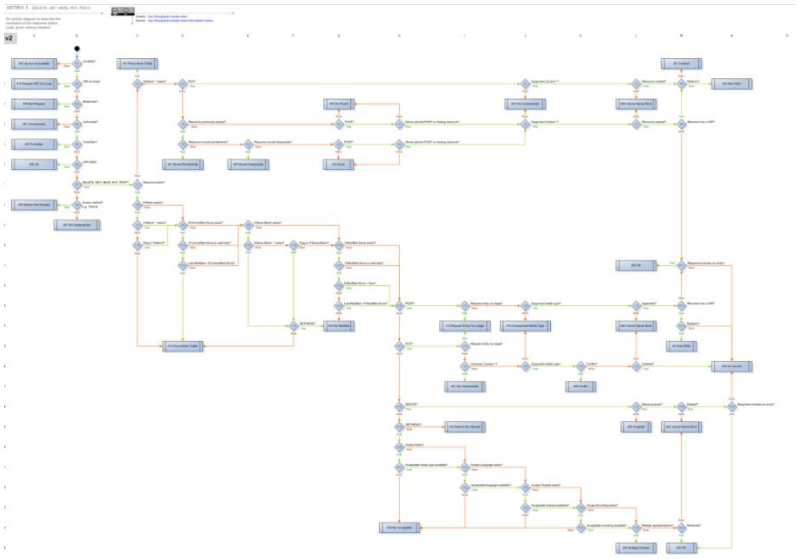
网络爬虫实现基础

- HTTP 响应码（Response code）
  - 1xx 消息
    - 100 Continue
    - 101 Switching Protocols
    - 102 Processing
  - 2xx 成功
    - 200 OK
    - 201 Created
    - 202 Accepted
    - 203 Non-Authoritative Information
    - 204 No Content
    - 205 Reset Content
    - 206 Partial Content
    - 207 Multi-Status



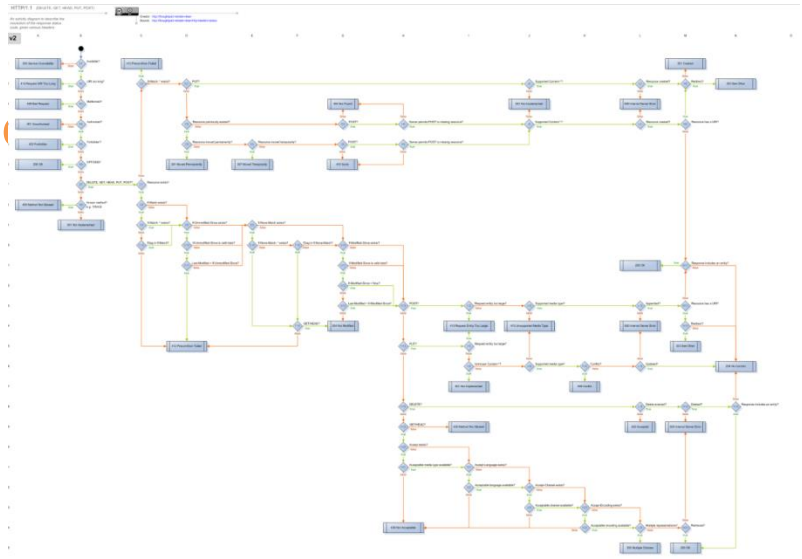
网络爬虫实现基础

- HTTP 响应码（Response code）
  - 3xx 重定向
    - 300 Multiple Choices
    - 301 Moved Permanently
    - 302 Move temporarily
    - 303 See Other
    - 304 Not Modified
    - 305 Use Proxy
    - 306 Switch Proxy
    - 307 Temporary Redirect



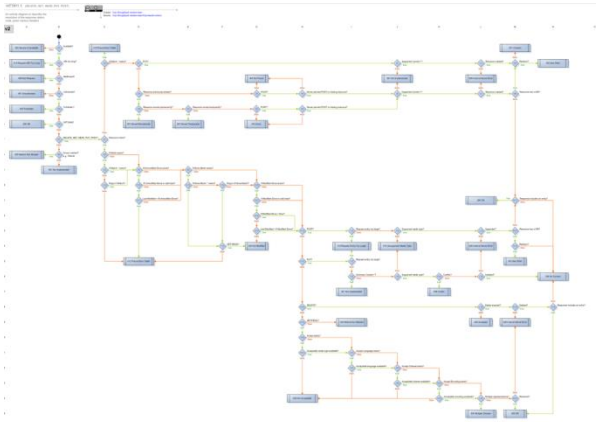
网络爬虫实现基础

- HTTP 响应码（Response code）
  - 4xx 请求错误
    - 400 Bad Request
    - 401 Unauthorized
    - 402 Payment Required
    - 403 Forbidden
    - 404 Not Found
    - 405 Method Not Allowed
    - 406 Not Acceptable
    - 407 Proxy Authentication Re
    - 408 Request Timeout
    - 409 Conflict
    - 410 Gone
    - 411 Length Required



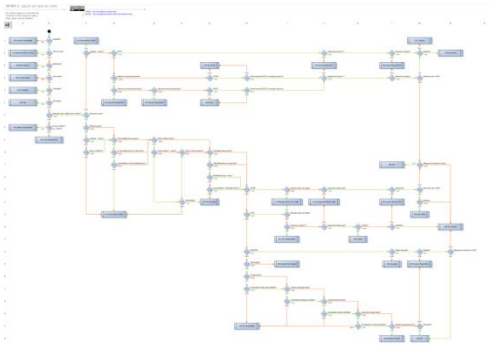
网络爬虫实现基础

- HTTP 响应码（Response code）
  - 4xx 请求错误
    - 412 Precondition Failed
    - 413 Request Entity Too Large
    - 414 Request-URI Too Long
    - 415 Unsupported Media Type
    - 416 Requested Range Not Satisfiable
    - 417 Expectation Failed
    - 421 too many connections
    - 422 Unprocessable Entity
    - 423 Locked
    - 424 Failed Dependency
    - 425 Unordered Collection
    - 426 Upgrade Required
    - 449 Retry With
    - 451Unavailable For Legal Reasons



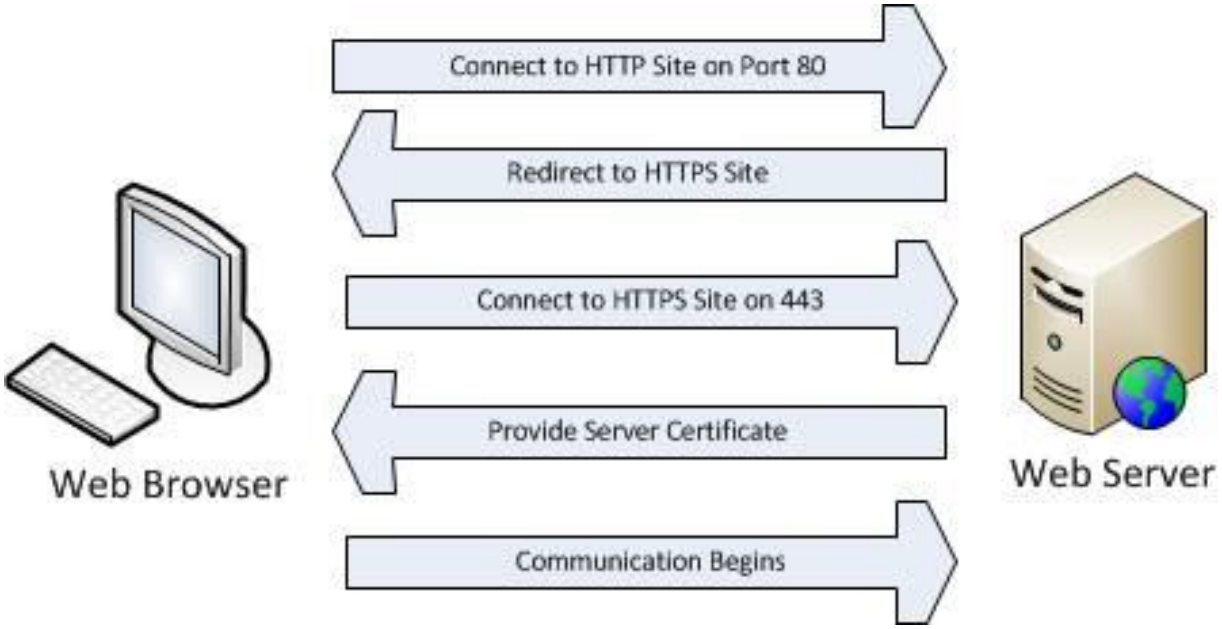
网络爬虫实现基础

- HTTP 响应码（Response code）
  - 5xx 服务器错误
    - 500 Internal Server Error
    - 501 Not Implemented
    - 502 Bad Gateway
    - 503 Service Unavailable
    - 504 Gateway Timeout
    - 505 HTTP Version Not Supported
    - 506 Variant Also Negotiates
    - 507 Insufficient Storage
    - 509 Bandwidth Limit Exceeded
    - 510 Not Extended
    - 600 Unparseable Response Headers



网络爬虫实现基础

- HTTPS
  - 承载于TLS或SSL之上的HTTP，默认端口443；
  - 通过加密、认证的方式实现数据传输安全。



网络爬虫的简单实现

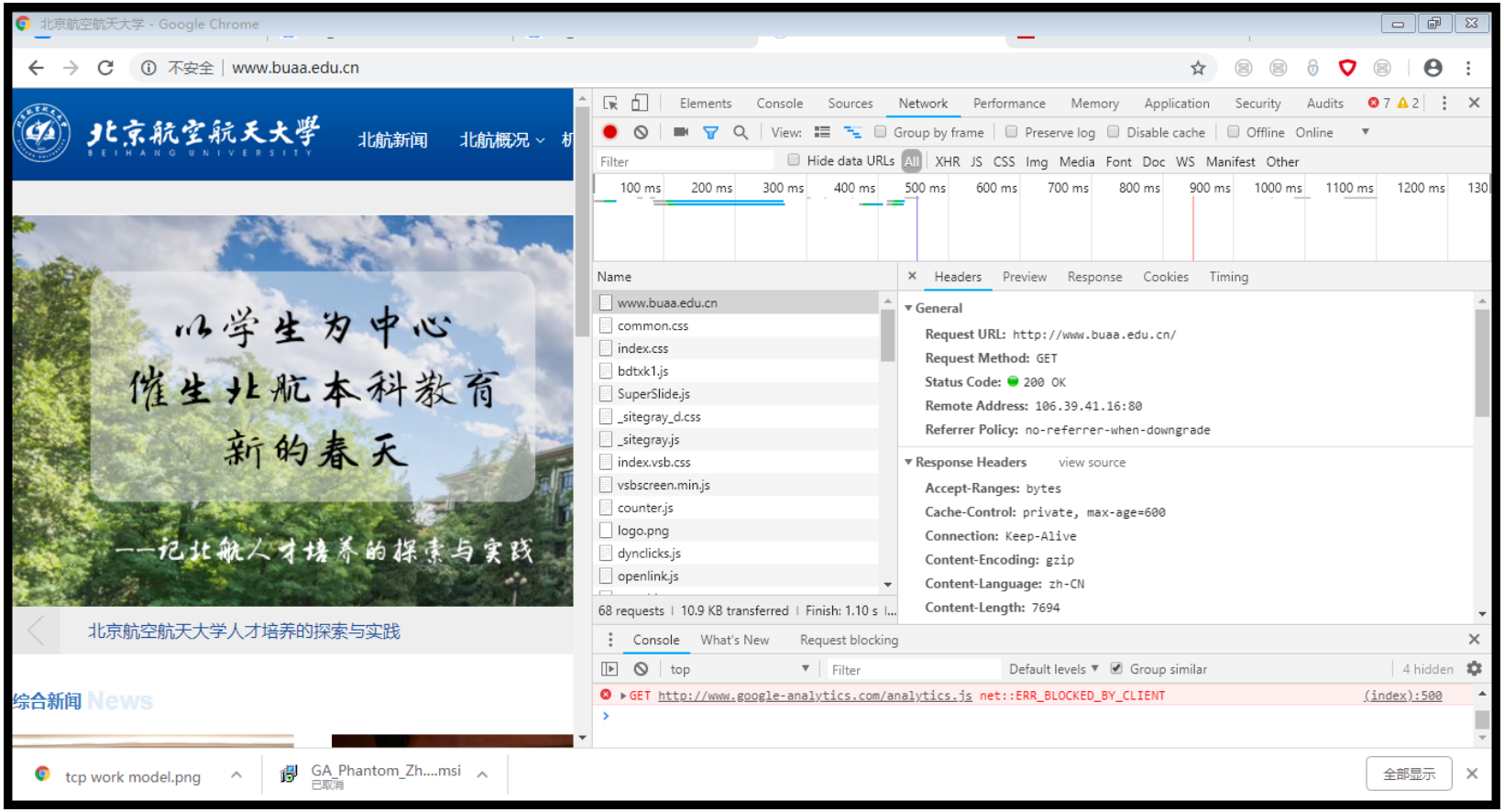
- HTML超文本标记语言
  - HTML 是用来描述网页的一种语言。
  - 由一套标记标签 和网页内容组成

实例

```
<html>
<body>

<h1>我的第一个标题</h1>

<p>我的第一个段落。</p>

</body>
</html>
```

小实验

- 利用浏览器（以chrome为例）查看URL、html和http请求与响应

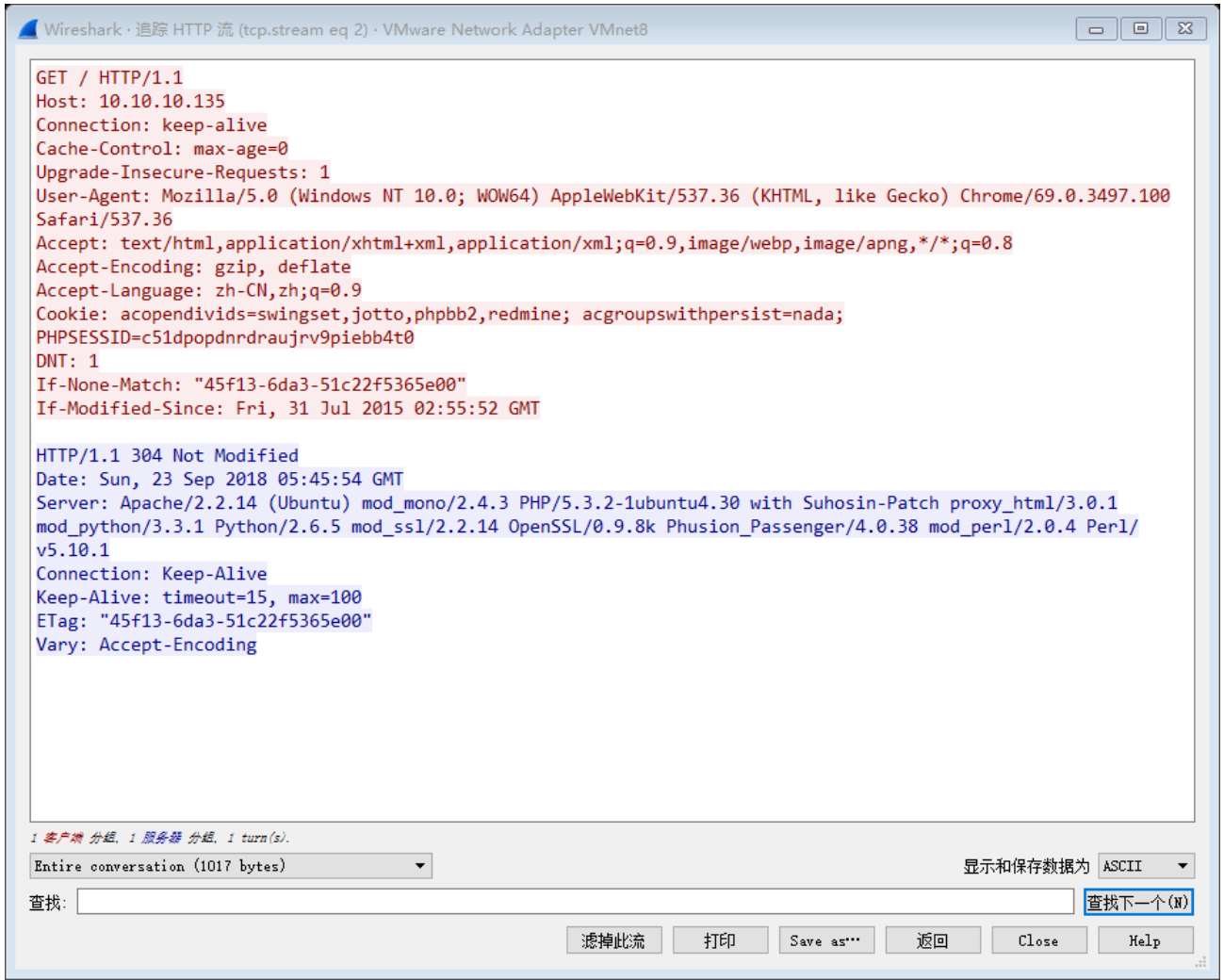


实验辅助工具

- Web debugger
  - Chrome developer tools
- Net protocol/data packeage analyser
  - Wireshark
  - Telerik fiddler
    - https://www.telerik.com/download/fiddler/fiddler4
- Proxy tools
  - SwitchyOmega

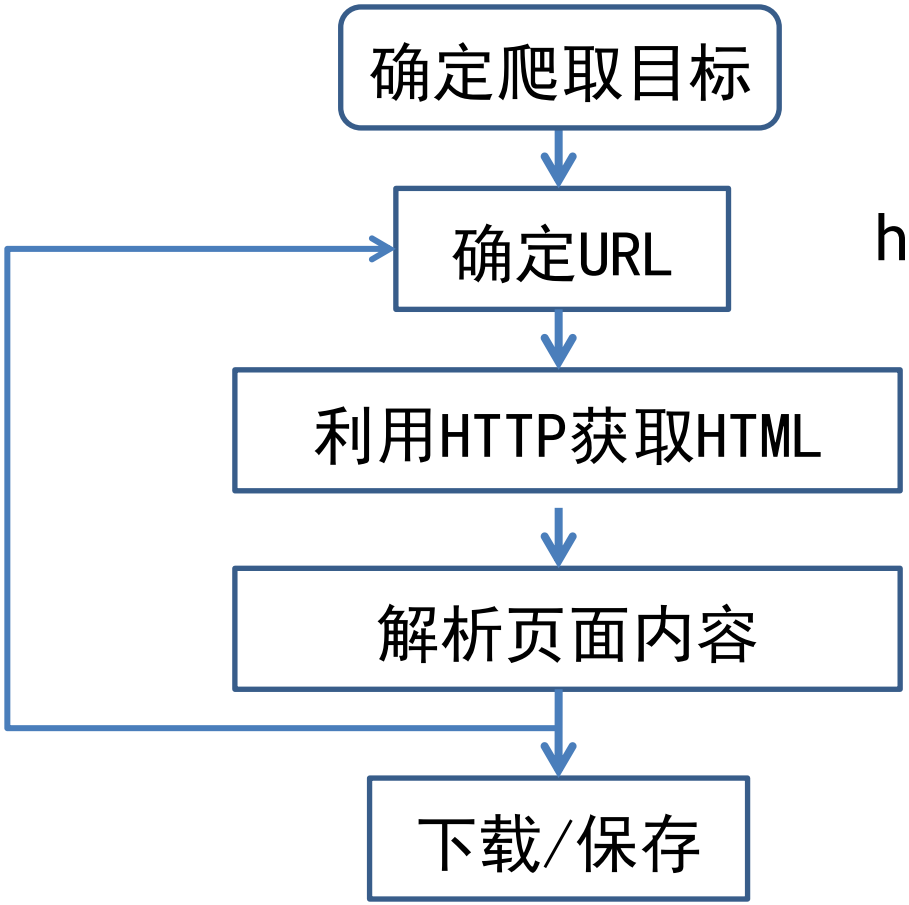
小实验

- 使用wireshark跟踪http流



网络爬虫的简单实现

- 基本思路



http://httpbin.org/

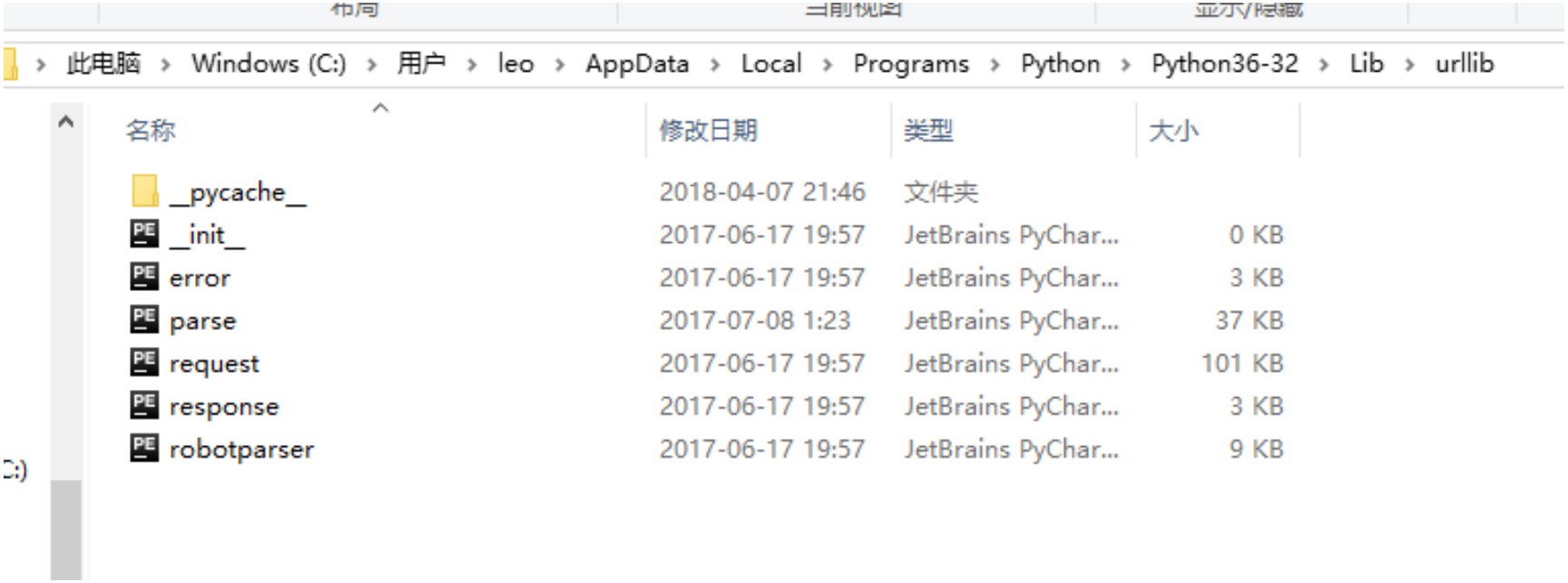
Howto auto ?

使用urllib实现简单网络爬虫

- urllib
  - A python module for fetching URLs
    - Using a variety of different protocols.
    - ftp、http...
  - It offers a Simple interface
    - urlopen( ) function
  - Also, it offers a complex interface for:
    - Authentication
    - Cookies
    - Proxies

使用urllib实现简单网络爬虫

- 先看一下urllib源代码
  - Windows 下查看 C:\Users\leo\AppData\Local\Programs\Python\Python36-32\Lib\urllib\



使用urllib实现简单网络爬虫

- 预备
  - 语言环节：python 3.5 或更高版本
  - 编辑工具：
    - jupyter（最好安装Anaconda3-5.2.0-以上版本）
      - https://www.anaconda.com/download/
    - Pycharm
      - http://www.jetbrains.com/pycharm/

使用urllib实现简单网络爬虫

- 计划一
  1. 掌握单页面内容的获取方法
     - urllib.request.urlopen();
     - urllib.request.urlretrieve();
     - 理解response对象。
  2. 掌握使爬虫更像浏览器的方法
     - 构建request对象;
     - 理解并填入必要的headers。
  3. 掌握向服务器传递参数的方法
     - GET方法传递参数;
     - POST方法传递参数。

使用urllib实现简单网络爬虫

- 计划一
  4. 掌握处理异常的方法
     - 理解urllib提供的Exception类；
     - 设置超时访问限制；
     - 建立一个较为稳定的爬虫代码框架。
  5. 阶段小结
     - 运用已学知识构建爬虫程序，爬取1-3页百度贴吧页面。

使用urllib实现简单网络爬虫

- 计划二
  1. 掌握自定义opener的方法
     - 理解handler与opener的含义；
     - 了解handler类型；
     - 掌握自定义opener的基本方式。
  2. 掌握自动提交身份认证的方法
     - 理解http basic-Auth方法；
     - 作业：自学digest-Auth。
  3. 掌握使用代理获取网页的方法
     - 了解代理工作机制；
     - 掌握免费代理与需认证代理的使用方法。

使用urllib实现简单网络爬虫

- 计划二
  4. 掌握获取ajax异步加载网页内容的方法
     - 理解ajax机制；
     - 学会使用抓包工具，分析有效链接；
     - 掌握解析json内容的方法。
  5. 掌握使用cookie获取需认证网页的方法
     - 利用已登录cookie获取需认证才能访问的网页；
     - 利用cookie处理库自动记录并登录网站。