网络引文的爬虫系统初探

摘要

随着互联网技术的高速发展，网络信息资源越来越受到人们的重视，成为人们获取信息和进行科学研究的主要信息来源。网络引文，即网络参考文献也逐渐受到众多学术工作者的青睐，并成为一种不可忽视的引文类型。统计分析网络引文，不仅可以了解各领域对网络引文的利用现状，而且可以对其进行系统分析、数据挖掘。本文以万方数据库为例，通过爬虫拉取部分论文的网络引文，并进行解析、存储，为后续分析数据提供了强有力的数据保障。

关键字：网络引文；引文分析；数据挖掘；网络爬虫

Abstract

目录

[摘要 1](#_Toc482112314)

[Abstract 1](#_Toc482112315)

[目录 1](#_Toc482112316)

[第一章 绪论 2](#_Toc482112317)

[1.1. 论文的研究背景及意义 2](#_Toc482112318)

[1.2. 网络爬虫系统的研究现状 2](#_Toc482112319)

[1.3. 论文的相关工作 3](#_Toc482112320)

[1.4. 论文章节安排 3](#_Toc482112321)

[第二章 相关概念及技术背景 3](#_Toc482112322)

[2.1. 网络引文的概念 3](#_Toc482112323)

[2.1.1. 引文 3](#_Toc482112324)

[2.1.2. 网络引文 4](#_Toc482112325)

[2.2. 网络爬虫技术 4](#_Toc482112326)

[2.2.1. 网络爬虫的概念 4](#_Toc482112327)

[2.2.2. 网络爬虫的工作流程 4](#_Toc482112328)

[2.3. Node.js 5](#_Toc482112329)

[2.4. 正则表达式 5](#_Toc482112330)

[2.5. 本章小结 5](#_Toc482112331)

[第三章 系统的需求分析 6](#_Toc482112332)

[3.1. 业务需求描述 6](#_Toc482112333)

[3.2. 系统的功能性需求分析 6](#_Toc482112334)

[3.2.1. 网页下载与信息过滤 6](#_Toc482112335)

[3.2.2. 抽取有效数据 6](#_Toc482112336)

[3.2.3. 数据的更新与存储 6](#_Toc482112337)

[3.3. 本章小结 6](#_Toc482112338)

[第四章 网络引文爬虫系统的设计 6](#_Toc482112339)

[4.1. 系统的总体设计 6](#_Toc482112340)

[4.2. 系统的功能设计 7](#_Toc482112341)

[4.2.1. 系统的总体功能设计 7](#_Toc482112342)

[4.2.2. 网页下载与信息过滤功能设计 7](#_Toc482112343)

[4.2.3. 抽取有效数据功能设计 7](#_Toc482112344)

[4.2.4. 任务队列功能设计 7](#_Toc482112345)

[4.2.5. 数据的更新与存储功能设计 8](#_Toc482112346)

[4.3. 本章小结 8](#_Toc482112347)

[第五章 网络引文爬虫系统的实现 8](#_Toc482112348)

[5.1. 系统环境 8](#_Toc482112349)

[5.2. 系统的功能实现 8](#_Toc482112350)

[5.2.1. 网页下载与信息过滤功能实现 8](#_Toc482112351)

[5.2.2. 抽取有效数据功能实现 8](#_Toc482112352)

[5.2.3. 任务队列功能实现 8](#_Toc482112353)

[5.2.4. 数据的更新与存储功能实现 8](#_Toc482112354)

[5.3. 测试报告 8](#_Toc482112355)

[5.4. 实验结论 8](#_Toc482112356)

[5.5. 本章小结 8](#_Toc482112357)

第一章 绪论

* 1. 论文的研究背景及意义

近年来，计算机和信息技术有了突飞猛进的发展，产生了许多新概念和技术，如更高性能的计算机和操作系统、因特网（internet）、数据仓库（datawarehouse）、神经网络等等。在市场需求和技术基础都满足的情况下，数据挖掘的概念和技术应运而生。

数据挖掘（Data Mining）旨在从大量的、不完全的、模糊的随机数据中提取出人们事先不知道的潜在的有价值的信息或知识。http://www.xuexila.com/baikezhishi/1244762.html

网络的迅速发展使各行各业都开始采用计算机及相应的信息技术进行管理和运营。网络引文由于其传播方式的自由性、动态性，也逐渐成为论文参考文献的主要来源之一。为了能更方便的对论文中引用到的网络引文进行数据分析和数据挖掘，我们需要更方便的收集论文的网络引文，然而由于数据量较大、手工操作较为复杂，导致网络引文分析所需的数据供给不足，这时就需要一个网络引文的爬虫系统来拉取数据，提供数据分析和数据挖掘所需的海量数据。

* 1. 网络爬虫系统的研究现状

网络爬虫系统主要是用来在浩瀚网络中收集网页或特定类型的数据信息，然后从中解析出客户所需要的信息，并持久化到数据库系统。

目前在国内外存在很多互联网爬虫系统，很多企业也会利用爬虫系统定时爬取所需数据。

基于web的爬虫系统是最普遍的，从某个URL作为根路径，然后对整个互联网节点进行信息采集，适用于广泛的主题搜索应用。采集数据量大，并且缺少针对性。这类爬虫代表系统有Mercator和Google。

另一种基于Agent的互联网爬虫系统，是利用Agent技术在信息采集的同时，对用户倾向进行观察和记录，然后不断调整数据采集策略。Agent爬虫系统能够智能、灵活地处理某些个性化信息采集。这类爬虫的代表系统有麻省理工学院研究开发的Letizia、Amalthaca、美国爱荷华大学的ARACHNID。

还有一种是基于增量的采集系统，主要用来采集那些数据更新频繁的网页。工作原理是：在完成初次数据采集后，后续的采集只针对那些改变了的数据，对于未改变的数据将不再处理。增量采集方式可以提高数据爬取的效率，最大程度地提高网络和磁盘等资源的利用率。代表系统有：天网增量搜集系统、Univ、WebFountain、Chile Crawler。

网络舆情爬虫系统的设计与实现 李海燕

1. 1. 论文的相关工作

本文以万方数据库为例，讲述网络引文爬取过程。首先介绍了网络爬虫的技术基础，然后研究了爬虫引文爬虫的一些关键问题如根URL的选取、URL消重等，最后实现了一个针对万方数据库的爬虫系统ReferenceSpider。

本文的主要研究内容如下：

1. 分析系统需求
2. 设计各功能模块
3. 描述爬虫系统的工作流程
4. 解析和保存数据
5. 测试数据

主题爬虫系统的研究 陈玉芳

网络爬虫系统的研究与实现

* 1. 论文章节安排

第一章为绪论部分，详细介绍了网络爬虫背景、现状和意义，最后给出论文的相关工作安排。

第二章针对网络爬虫开发所涉及到的技术知识进行了详细的阐述。

第三章分析了系统需求，主要是爬虫系统体系结构及各功能模块。

第四章对本系统开发的环境进行了描述。

第五章对ReferenceSpider进行具体实现。

第六章对论文进行总结，并给出下一步的工作。

第二章 相关概念及技术背景

* 1. 网络引文的概念
     1. 引文

引文是现代学术论文的重要组成部分，它表征了本文献与被引用文献之间的引用关系。在学术研究过程中，任何一篇论文的写作，都离不开已有文献或资源的借鉴，这里的借鉴可以理解为我们通常所说的引用。引用的内容称为引文。

引文，又称参考文献，出现的形式有两种，一是作为附属内容置于文章或者论文的后面，二是作为文章内容的注释存在。引文是作者在撰写文章时由于参考并引用了他人的有借鉴意义的观点而形成的著录方式。

引文具有以下功能：

1. 体现作者的写作态度以及反映论文具有真实的科学依据；
2. 表明作者对于前人成果的尊重以及展示自己的创新和发展；
3. 为读者提供所引用参考文献以及作者简介的出处；
4. 表明学科的发展以及与其他学科之间的联系；
5. 为情报研究和文献计量研究提供依据。

因此，引文已经成为学术论文中不可缺少的内容。

网络引文分析和传统引文分析的比较 马晓佳

国外网络引文研究的现状及展望 杨思洛

* + 1. 网络引文

网络引文（也称网络参考文献，Webcitation,Internet-based citation,Web-based citation）就是将网络文献作为学术论文参考文献的一种引文形式。网络文献是一种虚拟信息化资源，以计算机存储介质或网络为载体，传播交流形式自由，大多为超文本链接，具有动态性。

然而，不是所有的网络文献都可以成为网络引文。要成为网络引文，需要必要的学术认可，即要具有信息真实性、严肃性和资源的稳定性。在网络环境中，网络的开放性和自由度导致了其信息的不确定性和不规范性，从而无法保证信息真实性和严肃性。

虽然从学术研究的角度看，网络还未形成规范的学术交流体系，但网络的快速发展正猛烈冲击传统的学术交流体系，网络引文的大量使用已经无法避免。因此论文著作者在引用网络文献之前需要对其真实性、严肃性、稳定性进行评估，合理选择引用的网络文献。另外网络文献的开发者和建设者应该努力为网络资源的利用提供高效可行的评价指引机制，使其早日正式化、规范化。

网络引文信息的挖掘及其对数字图书馆资源建设的启示 王建芳

* 1. 网络爬虫技术
     1. 网络爬虫的概念

网络爬虫（Network Spider），顾名思义是在网络中爬行的爬虫。其实质为一段计算机程序，该程序对数据的收集通过网页链接地址实现，往往从某一根URL出发，通过向远程服务器发起HTTP请求，读取网页内容，并从中获得与其相关的节点或子节点的链接地址，然后利用获取到的链接地址进行深入循环爬取。

* + 1. 网络爬虫的工作流程

通用网络爬虫是从一个或几个初始URL开始，获得初始页面上的URL列表，在主线程中爬取网页的过程中，不断从当前页面筛选新的URL放入待爬行的URL列表，作为待执行的任务队列，由子线程执行，直到满足系统的停止条件。

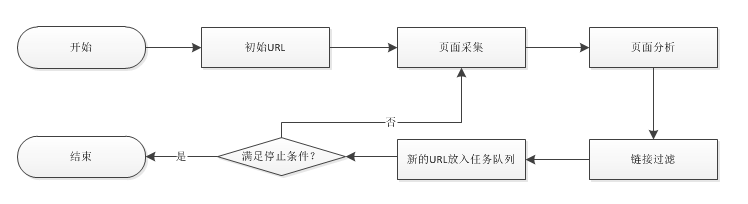


图1.1 网络爬虫工作流程

各个部分的主要功能介绍如下：

1. 初始URL：该模块提供初始的URL，以启动爬虫程序。
2. 页面采集模块：该模块是整个网络爬虫系统的入口模块，主要是通过各种web协议（通常为HTTP）来对网页数据进行采集。
3. 页面分析模块：该模块的功能是把页面采集模块采集到的页面数据进行分析，筛选出满足系统需求的URL，加入到任务队列中，如果URL为站点的相对路径，则需要加上网络协议、端口号、域名，使之成为可直接访问的完整的URL。
4. 链接过滤模块：该模块主要是过滤那些满足系统需求的正确格式的链接，并去除已经爬取的重复链接。

支持ajax的网络爬虫系统的设计和实现

* 1. Node.js

Node.js是一个基于Chrome’s V8 JavaScript engine的JavaScript运行环境（Runtime）。通过这个运行环境，可以让JavaScript像后端语言（Java，Python之类）那样，进行文件、数据库等操作。

Node.js产生的初衷，是Ryan Dahl想设计一个高性能的Web服务器。他发现高性能Web服务器有几个要点：事件驱动，非阻塞I/O。此外，Ryan Dahl设计了Node.js。

<http://www.jianshu.com/p/14bb2b4038d3>

基于Nodejs的微博系统的设计与实现 王越

Node.js具有以下特点：

Node.js以单进程、单线程的模式运行，它采用事件驱动的的机制实现并行操作。避免了系统分配线程及线程间通信的开销，可以更高效率的利用cpu的性能，降低内存的消耗。

不同于其他编程语言，Node.js采用非阻塞I/O，提高程序性能，更好的提高单个线程的利用率。

Node.js的设计思想是以事件驱动为核心的，Node.js提供的绝大多数API都是基于事件的、异步的风格。事件驱动、异步编程使得程序在执行时无需阻塞等待某个操作（如：HTTP请求、磁盘读写、数据库操作）的完成，使有限的系统资源能得到充分利用。

* 1. 正则表达式

对于网络爬虫系统来说，其实现对网页内容的爬取后，下一步就是利用正则表达式对网页内容进行筛选，提取所需的有效信息。

正则表达式的主要功能是实现字符串的匹配和截取，在爬虫程序中正则表达式对逻辑判断意义重大。本文实现爬虫的过程中，对URL及其他重要字段的解析与判断依靠正则表达式实现。

常见的正则表达式元字符解释如下：

|  |  |
| --- | --- |
| ^ | 匹配一个输入或一行的开头 |
| $ | 匹配一个输入或一行的结尾 |
| \* | 匹配前面元字符0次或多次 |
| + | 匹配前面元字符1次或多次 |
| ? | 匹配前面元字符0次或1次 |
| {n} | 精确匹配n次 |
| {n,} | 匹配n次以上 |
| {n,m} | 匹配n-m次 |
| [xyz] | 匹配这个集合中的任一一个字符(或元字符) |
| [^xyz] | 不匹配这个集合中的任何一个字符 |
| \d | 匹配一个数字字符 |
| \D | 匹配一个非数字字符 |
| \s | 匹配一个空白字符 |
| \S | 匹配一个非空白字符 |

* 1. 本章小结

本章对网络引文爬虫相关的概念及技术背景进行了简单的介绍，为后续的系统设计与实现提供了理论基础，具体包括：引文的概念、网络引文的概念、网络爬虫的概念、网络爬虫的工作流程以及网络爬虫用到的基础技术Node.js和正则表达式。

第三章 系统的需求分析

* 1. 业务需求描述

本系统以万方数据库为例，爬取万方数据库中引用有网络引文的文献，然后持久化到数据库。以《聚焦爬虫系统研究综述》为例，在论文详情页可以看到如下内容，点击（1）处的参考文献会出现右侧具体的参考文献的列表，如果参考文献较多，则会分页显示。在（2）处出现有[EB/OL]标识的即为引用的网络引文，系统需要分页爬取参考文献列表，并解析筛选含有“[EB/OL]”的参考文献，将其提取出来并持久化到数据库。



* 1. 系统的功能性需求分析
     1. 网页下载与信息过滤

系统首先对某搜索关键字搜索到的论文页面进行下载，然后根据DOM树分析得到的DOM选择器筛选出需要的论文列表，包括：论文的标题、论文在万方数据库中的访问URL、论文在万方数据库中存储的ID。之后再根据论文的URL爬取论文详情页的参考文献部分。

* + 1. 抽取有效数据

根据论文URL爬取论文详情页面的参考文献时，需要分页爬取所有的参考文献页面，并从这些参考文献列表中抽取后缀含有“[EB/OL]”的参考文献，即表示该论文引用了网络引文。我们还需要将这些网络引文的具体标题爬取出来并存储，以方便后续的分析工作。

* + 1. 数据的更新与存储

网络爬虫对根据爬取规则下载HTML页面，并根据DOM选择器筛选出所需数据后，需要将获取到的有效数据存储到MySQL数据库中，并根据关键字段进行内容的更新。

* 1. 本章小结

本章从系统的业务需求、功能性需求两个方面描述了系统的各方面需求。并具体阐述了网页下载与信息过滤、抽取有效数据、数据的更新与存储这些功能性需求，为下一步网络引文爬虫系统的设计提供了思路。

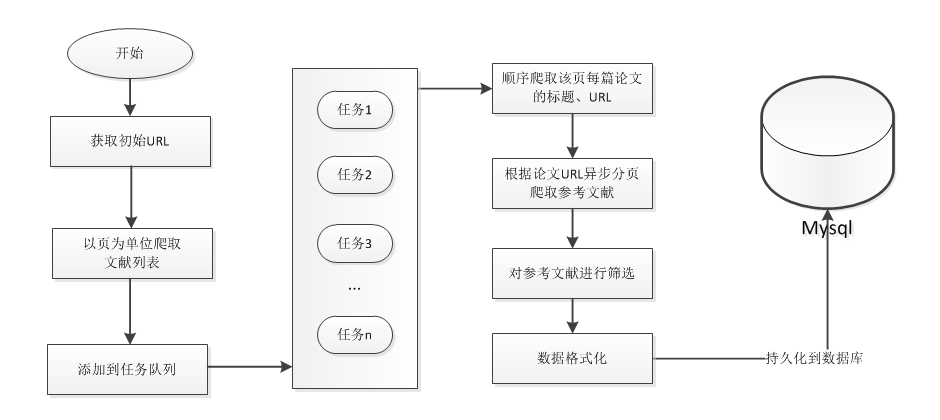
第四章 网络引文爬虫系统的设计

* 1. 系统的总体设计

根据上一章对系统需求的分析，本章设计网络引文爬虫系统，主要采集引用了网络引文的文献及具体的网络引文标题。该系统应达到以下条件：

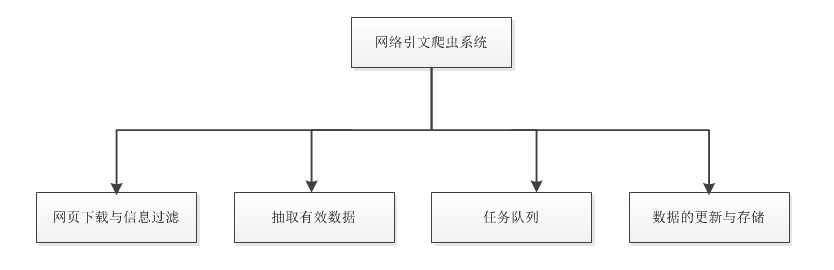
1. 易配置性。仅需通过简单的配置即可完成系统参数配置。
2. 爬取速度快。采用任务队列的方式，异步执行任务，充分利用Node.js任务驱动的特性，提高系统资源的利用率和系统吞吐量。
3. 较高的数据准确度。通过正则表达式准备匹配有效信息。

网络引文爬虫系统的系统结构如下：



网络爬虫系统结构

* 1. 系统的功能设计
     1. 系统的总体功能设计

网络引文爬虫系统主要包含了网页下载与信息过滤、抽取有效数据、任务队列、数据的更新与存储四大功能模块，功能模块图如下：

* + 1. 网页下载与信息过滤功能设计

此模块主要负责发送HTTP请求，下载HTML网页，并直观地过滤掉一些不可能存在有效信息的部分，以避免无意义的操作，浪费系统资源。

系统从初始URL开始下载网页，过滤掉网页中除了文献列表以外的其他区域，筛选出文献列表并添加到任务队列的末尾。

* + 1. 抽取有效数据功能设计

此模块负责对下载的HTML进行有效数据的进一步筛选。系统调用相应的匹配规则通过正则表达式对文献列表中各标签的内容进行匹配，匹配出最大页数以分页爬取、文章标题以及引文标题是否含有“[EB/OL]”字样等。

* + 1. 任务队列功能设计

该模块负责提供任务执行的环境，该任务队列可根据系统性能合理配置并发量，等待执行的任务进入队列末尾等待执行。这样在提高资源利用率的同时，可方便的监控队列中任务的积压情况。

* + 1. 数据的更新与存储功能设计

该模块负责将爬取到的有效数据存储到MySQL数据库中，MySQL通过唯一索引来达到数据消重的效果。

* 1. 本章小结

本章主要从系统总体设计、系统各功能模块设计两个方面详细阐述了系统的结构及主要功能，包括：网页下载与信息过滤、抽取有效数据、任务队列、数据的更新与存储，为下一步系统的具体实现提供了方案。

第五章 网络引文爬虫系统的实现

* 1. 系统环境

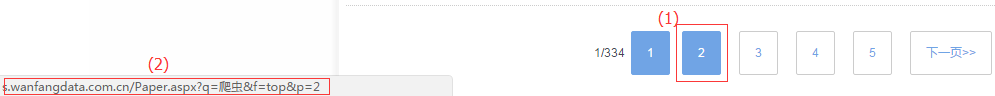
|  |  |
| --- | --- |
| 硬件环境 | CPU：Intel i7-5500U；内存：8GB DDR3；硬盘：1TB SATA3 |
| 操作系统 | Windows 7企业版 |
| 开发工具 | WebStrom 2017.1  Node.js v4.4.4  MySQL v5.0.95 |

* 1. 系统的功能实现
     1. 网页下载与信息过滤功能实现

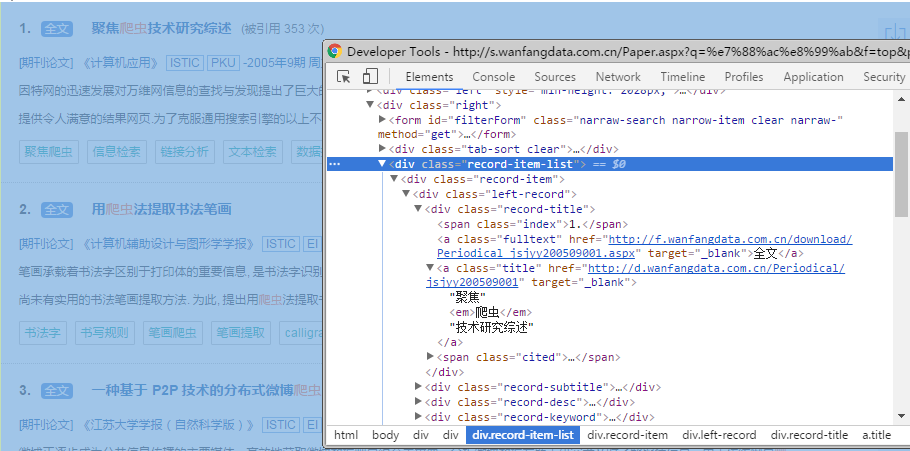
主要实现代码如下：

|  |
| --- |
| axios.get(encodeURI(domain + '/Paper.aspx?q=' + keyword + '&f=top&p=' + page))  .then(function (res) {  parseHtml(res.data, handler);  });  ***function*** parseHtml**(**html**,** handler**)** **{**  ***var*** $ **=** cheerio**.**load**(**html**);**  ***var*** pageTotal **=** parseInt**(**utils**.**getTotalPage**(**$**(**'.pager span'**).**text**()));**  ***if*** **(**isNaN**(**pageTotal**))** **{**  handler **&&** handler**(**0**,** 0**,** null**,** **{**error**:** '无数据'**});**  ***return*;**  **}**  ***var*** arr **=** $**(**'.left-record .record-title .title'**);**  ***var*** returnArr **=** **[];**  arr**.**each**(*function*** **(**i**,** item**)** **{**  ***var*** articleTitle **=** $**(*this*).**text**();**  ***var*** articleUrl **=** $**(*this*).**attr**(**'href'**);**  ***var*** articleId **=** utils**.**getArticleId**(**articleUrl**);**  returnArr**[**i**]** **=** **{**  articleTitle**:** articleTitle**,**  articleUrl**:** articleUrl**,**  articleId**:** articleId  **};**  **});**  handler **&&** handler**(**arr**.**length**,** pageTotal**,** returnArr**);**  **}** |

该处HTTP请求的地址可以通过观察页面结构得出，在将鼠标悬浮在图1中（1）处，可以观察到（2）处出现URL，由此可粗略判断该URL为包含文献列表结构的URL页面：



在图2中，通过Chrome的开发者工具，我们清楚的看到并验证了图1中（2）处的URL为包含文献列表的URL，且从图2中可清晰的看到HTML文档的DOM结构，通过“.left-record .record-title .title”选择器可对每篇文献的标题部分HTML进行定位及进一步筛选数据。



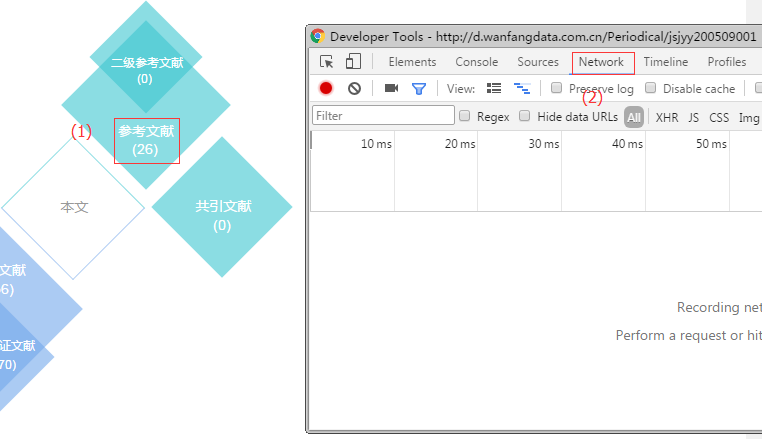
上述代码通过Node.js第三方包axios方便的实现HTTP请求的发送，下载网页内容，接下来parseHtml通过cheerio对下载的HTML进行过滤，过滤出文献的列表及每篇文献对应的标题、URL、id，提供给数据抽取模块进行下一步的数据抽取。该部分的两个步骤是爬虫系统的公共步骤，除了具体实现细节不同外，大体逻辑和结构都是相同的。

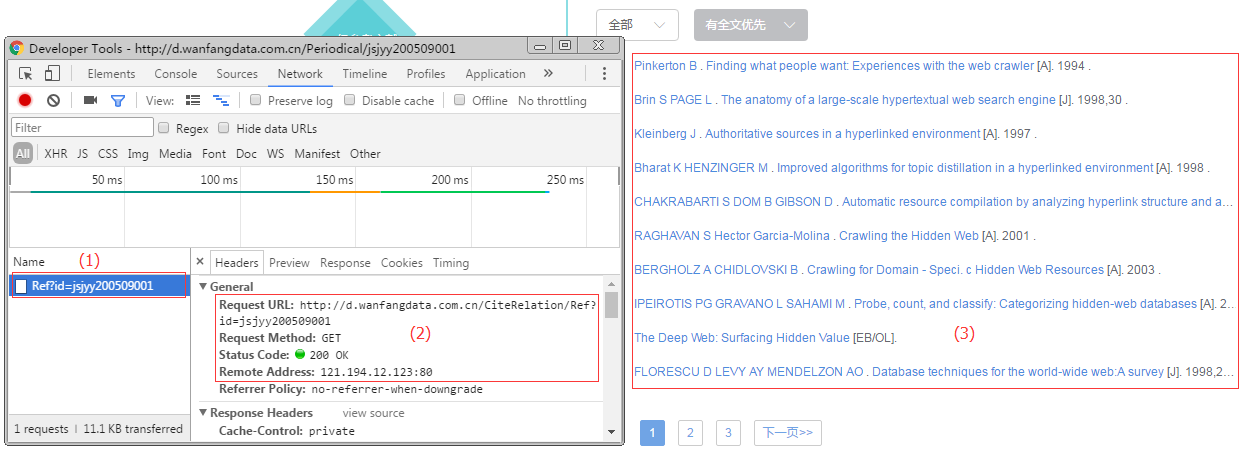
* + 1. 抽取有效数据功能实现

主要实现代码如下：

|  |
| --- |
| ***if*** **(**err**)** **{**  logger**.**error**(**'error:' **+** err**.**error**);**  **}** ***else*** **{**  arr**.**forEach**(*function*** **(**e**)** **{**  getBody**(**domain **+** '/CiteRelation/Ref?id=' **+** e**.**articleId**,** ***true*);**  **});**  **}**  ***function*** getBody**(**url**,** parsePage**)** **{**  axios**.**get**(**url**)**  **.**then**(*function*** **(**res**)** **{**  parseHtml**(**res**.**data**,** url**,** parsePage**);**  **});**  **}**  ***function*** parseHtml**(**html**,** url**,** parsePage**)** **{**  ***var*** $ **=** cheerio**.**load**(**html**);**  ***var*** totalPage **=** parseInt**(**utils**.**getTotalPage**(**$**(**'.pager .current'**).**text**()));**  ***var*** arr **=** $**(**'.paper-list .item'**);**  arr**.**each**(*function*** **(**i**,** item**)** **{**  ***var*** childrens **=** item**.**children**;**  ***if*** **(**childrens **&&** childrens**.**length **>** 0**)** **{**  childrens**.**forEach**(*function*** **(**e**)** **{**  ***if*** **(**e**.**type **===** 'text' **&&** utils**.**isEBOL**(**e**.**data**))** **{**  logger**.**info**(**getText**(**childrens**));**  **}**  **});**  **}**  **});**  ***if*** **(**parsePage**)** **{**  ***if*** **(**isNaN**(**totalPage**))** **{**  // logger.error("error:" + url);  **}** ***else*** **{**  ***for*** **(*var*** page **=** 2**;** page **<=** totalPage**;** page**++)** **{**  ***var*** index **=** url**.**indexOf**(**'&page='**);**  ***if*** **(**index **>** **-**1**)** **{**  url **=** url**.**substring**(**0**,** index**)** **+** '&page=' **+** page**;**  **}** ***else*** **{**  url **=** url **+** '&page=' **+** page**;**  **}**  getBody**(**url**,** ***false*);**  **}**  **}**  **}**  **}**  ***function*** getText**(**childrens**)** **{**  ***var*** returnStr **=** ''**;**  ***for*** **(*var*** i **=** 0**;** i **<** childrens**.**length**;** i**++)** **{**  ***if*** **(**childrens**[**i**].**type **===** 'tag' **&&** childrens**[**i**].**name **===** 'a'**)** **{**  childrens**[**i**].**children**.**forEach**(*function*** **(**e**)** **{**  ***if*** **(**e**.**type **===** 'text'**)** **{**  returnStr **+=** e**.**data**;**  **}**  **});**  **}**  **}**  ***return*** returnStr**;**  **}** |

在论文的详情页面中，我们需要爬取参考文献列表部分的HTML，但是在详情页面中我们并没有找到对应的HTML。因此，初步判断参考文献列表通过AJAX异步加载。为了验证这一假设，再次打开Chrome工具，保证图3中（2）处Network面板下内容为空，此时我们点击（1）处的参考文献，在图4中可以看到Network面板下（1）处出现了一次HTTP请求，在（2）处可以看到该HTTP请求的具体参数，如URL、请求的方法、远程IP等，在（3）处出现了参考文献列表，由此我们得到了参考文献的HTTP链接。





在上述代码中，我们依然通过网页下载与信息过滤模块的公共代码实现发送HTTP请求，解析HTML的功能。在实现细节上，通过一系列与业务相关的逻辑判断从参考文献列表中抽取有效数据，即网络引文。

* + 1. 任务队列功能实现

任务队列主要实现代码如下：

|  |
| --- |
| // 创建一个任务队列，并发：2  ***const*** q **=** async**.**queue**(*function*** **(**task**,** callback**)** **{**  // 任务执行过程  callback**(**task**);**  **},** 2**);**  // 添加到任务队列  ***for*** **(*var*** i **=** 1**;** i **<** 2**;** i**++)** **{**  q**.**push**({**page**:** i**},** ***function*** **(**task**,** err**)** **{**  logger**.**info**(**'finish page:' **+** task**.**page**);**  **});**  **}** |

在上述代码中创建了一个提供2个并发的任务队列，即同一时刻最多同时有2个任务在执行状态，后续任务被加入任务队列时，将进入队尾排队等待执行。

* + 1. 数据的更新与存储功能实现
  1. 系统测试
     1. 关键数据抓取测试
  2. 实验结论
  3. 本章小结