

# 基于 HttpClient 与 HTMLParser 的网页正文提取

陈智彬, 崔鸿雁

(北京邮电大学信息与通信工程学院, 北京 100876)

**摘要:** 随着互联网的高速发展, 针对互联网的分析处理显得日益重要。本文研究了 HttpClient、HTMLParser 等技术, 提出并实现了一种基于 HttpClient 与 HTMLParser 的网页抓取解析方法, 该方法能够快速有效对 HTML 页面进行抓取解析, 提取出所需的文本内容。

**关键词:** 正文提取; HttpClient; HTMLParser; Hadoop

**中图分类号:** TP393.092

## Web Content Extraction based on HttpClient and HTMLParser

Chen Zhibin, Cui Hongyan

(Information and Communication Engineering School, Beijing University of Posts and Telecommunications, Beijing 100876)

**Abstract:** With the rapid development of the Internet, the analysis on the Internet is becoming more and more important. This paper studies the HttpClient, HTMLParser technology, put forward and realizes a web capture and analysis method based on HttpClient and HTMLParser, this method can capture the HTML page and then extract the text content fast and effectively.

**Keywords:** text extraction; HttpClient; HTMLParser; Hadoop

## 0 引言

随着互联网时代信息与数据的快速增长, 出于科研、工程和商业等目的对网页数据进行针对性的提取显得越来越重要。提取网页信息是为后续的进一步处理工作服务的, 比如网页的分类、聚类、关联分析等等。当前, 网络所承载的业务类型也越来越多并不断呈现出新的特点, 与此同时, 网络上用户行为也呈现出多样化的特点。为了更好的分析用户的网上行为特点, 需要对用户浏览次数频繁的网页进行分析。由于要对海量的网页信息进行处理分析, 不可能采用浏览器人工阅览的方法。因此, 通过我们需要编写自己的分析处理程序, 提取出有用的信息, 为进一步分析做准备。

由于自身固有的缺点, 原始 HTML 页面的数据格式并不适合进行处理<sup>[1-2]</sup>。本研究将结合 HttpClient 与 HttpParser 对 HTML 页面进行抓取解析; 同时针对大数据量的处理情况, 采用基于 Hadoop 的分布式多线程架构。

## 1 开发工具与环境简介

### 1.1 HttpClient

网页抓取, 就是把 URL 地址中指定的网络资源从网络流中读取出来, 保存到本地。类似于使用程序模拟 IE 浏览器的功能, 把 URL 作为 HTTP 请求的内容发送到服务器端, 然后读取服务器端的相应资源<sup>[3]</sup>。虽然 java.net 包中已经提供了访问 HTTP 协议的基本功能, 但是对于大部分应用程序来说还不够丰富和灵活。

**作者简介:** 陈智彬, (1987-), 男, 硕士研究生, 主要研究方向: 下一代网络、云计算。

**通信联系人:** 崔鸿雁, (1977-), 女, 副教授, 未来网络理论与应用. E-mail: cuihy@bupt.edu.cn

40        HttpClient<sup>[4]</sup> 是 Apache Jakarta Common 下的子项目，用来提供高效的、最新的、功能丰富的支持 HTTP 协议的客户端编程工具包，并且它支持 HTTP 协议最新的版本和建议。HttpClient 能够处理 HTTP 连接中的各种问题，使用起来十分方便，只需导入 HttpClient.jar 包，就可以模拟 IE 浏览器来获取网页内容。

## 1.2 HTMLParser

45        HTMLParser<sup>[5]</sup> 是用来解析 HTML 文件的 Java 库，主要用于转换和提取两方面。它能超高速实时解析 HTML 文件，而且不会出错，是目前最好的 HTML 解析和分析的工具。HTMLParser 可以实现文本抽取、链接抽取、资源抽取、链接检查、站点检查、链接重写、网页内容拷贝、内容检验、HTML 信息清洗以及将 HTML 页面转成 XML 格式。

50        HTMLParser 采用了经典的 Composite 模式，通过 RemarkNode、TextNode、TagNode、AbstractNode 和 Tag 来描述 HTML 页面中的各元素<sup>[3]</sup>。HTMLParser 为我们提供了非常方便的 HTML 解析方式，不同的应用可以采用 Visitor 方式来遍历 HTML 节点提取数据，也可以采用 Filter 方式来过滤节点，提取出我们所关注的节点，再对节点进行处理<sup>[6]</sup>。

## 1.3 Hadoop

55        Hadoop<sup>[7]</sup>是 Apache Lucene 创始人 Doug Cutting 创建的，起源于另外一个开源的网络搜索引擎项目 Apache Nutch，是 MapReduce<sup>[8]</sup> 分布式程序设计模型的开源实现架构，用于在集群上对海量数据进行并行处理。Hadoop 已成为很多大型互联网公司计算架构的核心部分，比如 Google、Yahoo、FaceBook、LinkedIn 以及 Twitter 等。它具有可扩展、经济、可靠、高效等特点，非常适合用于海量数据的分析。

## 2 系统结构

60        本系统采用 HttpClient 对输入的 URL 进行抓取，当抓取成功时，则将 HTML 文档保存下来。由于中文网页的编码可能是 UTF-8 或 GB2312 格式的，因此如果没有采用对应的编码格式进行解析的话会产生乱码的问题。据调查 80%以上的网页都没有设置 response Header 的 charset，所以直接通过 HttpClient 获取 charset 是不可靠的；但是有 99%的网页都设置了 meta 元素的 charset<sup>[9]</sup>，所以可以先根据一个默认编码（ISO-8859-1）获得 HTML 文档，再  
65        根据 HTMLParser 或正则表达式获得该文档的 charset。最后通过使用 HTMLParser 的 TitleTag、MetaTag、TextNode 等标签来获得 HTML 文档的文本内容。

      整个抓取解析的流程如图 1 所示。为了处理海量的 URL 信息，可以在 Hadoop 平台上实现此抓取解析应用程序。在分布式多线程的环境下，指向相同主机的 URL 将被分配到同一个抓取队列中。



图1 抓取解析流程图

### 3 系统实现

以下代码为实现抓取解析功能的调用方法:

```
75 public static ProtocolOutput getHttpOutput(String url) {  
    int status = -1;  
    HttpClient httpclient = new HttpClient();  
    //set http connection timeout  
    httpclient.getHttpConnectionManager().getParams().setConnectionTimeout(5000);  
80    GetMethod getMethod = new GetMethod(url);  
    //set get request timeout  
    getMethod.getParams().setParameter(HttpMethodParams.SO_TIMEOUT, 5000);  
    //set retry handler  
    getMethod.getParams().setParameter(HttpMethodParams.RETRY_HANDLER,  
85        new DefaultHttpClientRetryHandler());  
    try {  
        status = httpclient.executeMethod(getMethod);  
  
        if( status == HttpStatus.SC_OK) {  
90            String html = new String(getMethod.getResponseBody(),"ISO-8859-1");  
            String charSet = getCharSet(html);  
            String input = new String(html.getBytes("ISO-8859-1"), charSet);  
            String content = "";  
            try {  
95                content = readText(input, charSet);  
            } catch (ParserException e) {  
                e.printStackTrace();  
            }  
            URLContent urlContent = new URLContent(content);
```

```
100         return new ProtocolOutput(urlContent,HttpStatus.SC_OK);
           } else {
               return new ProtocolOutput(null, status);
           }
       } catch (HttpException e) {
105         LOG.error("Please check you provided http address!" + e.getMessage());
       } catch (IOException e) {
           //network abnormal
           LOG.error(e.getMessage());
       } finally {
110         getMethod.releaseConnection();
       }
       return new ProtocolOutput(null, status);
   }
```

其中返回类型 `ProtocolOutput` 是存储提取内容与响应状态的类。`getCharSet` 方法采用正

115 则表达式获得网页的编码格式，其实现代码为：

```
public static final String reg= "<meta[^>]*?charset=([a-z|A-Z|0-9]*[\\-]*[0-9]*)[\\s\\S]*";
private static String getCharSet(String content) throws PatternSyntaxException {
    Matcher m = pCharSet.matcher(content);
    if(m.find()) {
120         return m.group(1);
    }
    return "UTF-8";
}
```

网页中的标题、`keywords` 和 `Description` 三项对于网页属性的描述具有指导作用，因此

125 提取正文也应该将它们考虑在内，`readText` 实现了对去噪后的网页的正文提取：

```
private static String readText(String input, String contentEncode) throws ParserException {
    String content = delTag("script",new StringBuffer(input)).toString();
    Parser parser = Parser.createParser(content, contentEncode.trim());
    NodeList nodelist;
130    NodeFilter textFilter = new NodeClassFilter(TextNode.class);
    NodeFilter titleFilter = new NodeClassFilter(TitleTag.class);
    NodeFilter metaFilter = new NodeClassFilter(MetaTag.class);
    OrFilter lastFilter = new OrFilter();
    lastFilter.setPredicates(new NodeFilter[] { textFilter, titleFilter, metaFilter });
135    nodelist = parser.parse(lastFilter);
    Node[] nodes = nodelist.toNodeArray();
    .....
}
```

140 对于不少 HTML 页面，HTML 元素的长度超过了正文的长度，有时候会混入很多 JavaScript。`HtmlParser` 经常会将这些元素误认为是正文加以识别，导致很多正文竟然是一段 JavaScript<sup>[3]</sup>。`delTag` 方法的作用就是去除特定标签内容，这里用来去除 JavaScript。

通过对百度(UTF-8)、新浪(GB2312)、雅虎(UTF-8)这三家具具有不同页面编码的主页进行抓取解析，发现可以正确得到正文，如图 2 所示：

145

图 2 正文提取结果

## 150

本文提出了一种基于 HttpClient 与 HTMLParser 相结合的网页正文抓取解析方法，该方法可以准确快速地提取出用户所需的内容，并且很好地解决了中文乱码的问题。为了应对海量数据的处理情况，在 Hadoop 分布式平台上实现了该方法。

## 155

- 160