第3章 互联网大数据的提取技术

本章描述了从 Web 页面上提取感兴趣信息的方法，包括基于特征模板、基于页面解析树的方法，以及基于统计的方法等。同时考虑到互联网大数据来源的多样性，除了 Web 页面外，也简单介绍了 Web 日志信息和 ETL 信息提取方法，并结合阿里云公众趋势分析介绍了 Web 信息提取的应用效果。

3.1 Web 页面内容提取技术

Web 页面中包含有丰富的信息内容，对于互联网大数据分析有用的信息可能是某个新闻报道页面中的正文部分，也可能是某网络论坛中的帖子信息、人际关系信息等。在进行 Web 页面内容提取时，一般是针对特定的网站，因此，可以假设页面结构特征是已知的。在这种条件下，页面内容的提取就是根据结构和内容特征进行提取，在方法上大同小异。这里主要介绍两大类目前使用的主要方法，即基于 HTMLParser 的解析和基于 Jsoup 的页面内容提取。

3.1.1 Web 页面内容提取的基本任务

从 Web 页面中提取内容，首先要对 Web 页面的各种常见版面进行整理归纳。目前 Web 页面版式各式各样，但可以归结为以下 3 种。

(1) 新闻报道型页面。页面上尽管可能会有导航区、外部链接区、版权声明区等区域，但是作为新闻正文文字一般是占主要的位置。典型的如图 3-1 所示的参考消息网站的新闻报道，页面的最上面是一些广告、导航条，右边是一些信息推荐。对于这种类型而言，目标就是提取正文部分的内容。

(2) 列表型页面。这类页面为用户提供一种列表式的阅读，一般是作为聚集信息的访问入口。比较常用于新闻列表、网络论坛中的讨论区入口等。对于这种类型，通常会遇到翻页，即上一页、下一页等链接，允许用户在不同的列表页面上跳转。图 3-2 所示的是两种典型的列表型页面，左右两边分别来自网络论坛和新闻网站。对于这种类型而言，目标就是提

图 3-1　新闻报道的版面

取列表部分的所有内容。

图 3-2　两种列表型页面

（3）评论型页面。用户在页面对某个事物、话题发表自己的观点。这种页面整体上看可以是一种列表型的，但是设计者更加关心每个评论中的具体信息。一般每个评论会有评论人、评论内容、评论时间、评论对象及评论的一些量化信息等。图 3-3 所示的是大众点评网上针对某个菜馆的评论信息。对于这种类型而言，目标就是提取每个评论的各个具体信息。

以上是从界面的角度来看页面内容提取，设计者关心的是从程序处理角度的 Web 页面信息提取。

与浏览器界面所输出的效果不同，程序所看到的是 Web 页面对应的 HTML 编码文件。例如，对于上面的股票网络论坛的列表型页面，其对应的 HTML 编码文件内容如下（其中列出了前面两个记录）：

图 3-3 评论型页面

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
    <meta name="renderer" content="webkit">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta charset="utf-8">

    <title>浪潮信息(000977)_浪潮信息股吧_000977 股吧_股吧_东方财富网股吧</title>
    <meta name="keywords" content="浪潮信息_000977_股吧">
    <meta name="description" content="浪潮信息(000977)_东方财富网股吧">
</head>
<body class="hlbody">

...
<div class="articleh">
  <span class="l1">885737</span><span class="l2">2890</span><span class="l3">
<em class="settop">话题</em><a href="news,600258,432898335.html" title="中鑫富盈、吴峻乐操纵特力A等股票案罚没金额超过10亿元">中鑫富盈、吴峻乐操纵特力A等股票案罚没金额超</a></span><span class="l4"><a href="http://iguba.eastmoney.com/9313013693864916"
data-popper="9313013693864916" data-poptype="1" target="_blank">财经评论</a></span>
<span class="l6">07-01</span><span class="l5">07-02 16:41</span>
    </div>

    <div class="articleh">
    <span class="l1">386824</span><span class="l2">157</span><span class="l3">
<em class="settop">话题</em><a href="news,cjpl,433467336.html" title="证监会三大配套措施加强对重组上市监管">证监会三大配套措施加强对重组上市监管</a></span><span class="l4"><a href="http://iguba.eastmoney.com/9313013693864916" data-popper="9313013693864916" data-poptype="1" target="_blank">财经评论</a></span><span class="l6">07-02</span><span class="l5">07-02 16:42</span>
```

```
    </div>
    ...
</body>
</html>
```

可以看出，两个帖子记录都是由 HTML 的 Tag < div class＝"articleh">所界定，Web 内容提取就需要寻找能够定位记录的这种 Tag 标记。当然，这种特征标记也未必存在，这就要求采用一些程序上的技巧了。

3.1.2　Web 页面解析方法概述

可以看出，为了提取出在浏览器上所看到的格式化的记录信息，在程序处理中，就必须在相应的 HTML 编码文件中寻找所要提取的记录，并进行提取。

虽然页面类型很多，但无论是针对哪种类型的页面，在信息提取方面的基本思路是一致的，一般有以下 3 个步骤。

(1) 分析所处理的 HTML 源文件的特征。由于 HTML 文件中包含了大量的标记(Tag)，这些标记描述了 Web 浏览器在页面上如何显示文字、图形等内容，因此需要事先分析所要提取的信息内容所具有的标记特征。

(2) 先根据某种特征在 HTML 源文件中定位要提取的内容所在的块(Block)。

(3) 在 Block 内再利用块内特征提取具体内容。

现有方法都比较成熟，主要在于第(2)个步骤可以采用不同的定位方法。

最简单的定位方法是采用字符串匹配，以下是 Java 的一个片段，用于提取评论型页面的“楼层”信息。

```
//p1 是楼层在 HTML 中的开始位置
p1 = html.indexOf("< div class = louceng >");
//s 是< div class = louceng >之后的字符串
s = html.substring(p1 + new String("< div class = louceng >").length);
//得到楼层字符串
p2 =  s.indexOf("</div >");
louceng = s.substring(1,p2);
```

这种字符串分析方法虽然实现起来很简单，但是该方法存在很多问题，主要是扩展性不好、适应能力很差、缺乏代码的复用能力。

高级的 Web 信息内容抽取方法主要有以下几种。

(1) 基于正则表达式的信息抽取技术。正则表达式是用一种用来标识具有一定信息分布规律的字符串。在网页信息抽取过程中，首先把网页作为一个字符流的文件来处理，通过配置合理的正则表达式去匹配(定位)待抽取的信息，然后抽取其中的信息。

例如，以下片段采用一个正则表达式提取页面中< date >标记的所有日期。

```
import java.util.regex.Matcher;
import java.util.regex.Pattern;
public static void findhtml(String htmlstr){
```

```
        //匹配<date>开头,</date>结尾的字符串
        Pattern pat = Pattern.compile("<date>([^</date>]*)");
        //对 Web 页面数据串 htmlstr 进行匹配
        Matcher m = pat.matcher(htmlstr);
        while (m.find()) {
          String tmp = m.group();
          if (!"".equals(tmp)) {
             tmp = tmp.substring(6);
             System.out.println(tmp);
          }
        }
        return ;
    }
```

这种方法的优点是：通过正则表达式可以高效地抽取具有固定特征的页面信息，准确性很高，而且由于现今的主流编程语言基本上都提供了操作正则表达式的封装 API，因此可以很方便快捷地构建基于这种模式的 Web 信息抽取系统。

其缺点是：不能抽取那些未知特征的网页；必须为每种类型的页面信息编写相应的正则表达式；正则表达式的编写比较复杂，需要有一定的水平，特别是要有很强的观察能力，才可以编写出高效的正则表达式，即对编写者的要求较高。

（2）基于 HTML 结构的信息抽取技术。该方法的基本思路是，根据 Web 页面的结构组织形式定位待提取的信息。基于 HTML 结构的信息抽取过程如下。

首先通过 HTML 解析器将 Web 文档解析成 DOM 树（Document Object Model，文档对象模型），这是由于 HTML 的标签是可以嵌套的。

然后结合一定的规则抽取相关信息，这些规则可以自动或半自动方式得到，从而将信息抽取转换为对 DOM 树的操作。

这类技术的典型代表信息抽取系统有 LIXTO、XWRAP、RoadRunnert 和 W4F 等，其中 LIXTO 抽取系统允许用户以可视化、交互式方式对样本页面中的信息位置进行标识，从而生成抽取规则，实现对具有相似结构的 Web 页面进行内容抽取。

（3）基于统计的信息抽取技术。基于统计的网页信息抽取技术与基于 HTML 结构的信息抽取方法类似，都是先获得 Web 页面对应的 DOM 树。但这个方法是基于统计信息。其基本思路是：该方法先利用解析器把网页按照 HTML 标记的结构生成对应的 DOM 树，然后基于某种统计信息来获得正文内容。这里的统计信息要求能够把 Web 页面中需要提取的内容和其他内容区分开来，由若干个统计量组成。

基于统计的网页信息抽取方法具有一定的适用性。但此种方式对网页正文信息的抽取依赖阈值，阈值设定得好坏，将影响信息抽取的准确性，抽取的结果还可能含有噪声。

3.1.3 基于 HTMLParser 的页面解析

HTMLParser 是一个 Java 开源系统，一种基于 DOM 树的页面内容提取方法。

文档对象模型（Document Object Model，DOM）以面向对象的方式描述文档。它是

W3C 组织(即万维网联盟,World Wide Web Consortium)推荐的处理可扩展标志语言的标准编程接口,DOM 是 W3C 制定的标准,该标准包含了通过 DOM 方式访问 HTML 和 XML 文档的方法,分别称为 HTML DOM 和 XML DOM。在 Web 页面提取的应用中就是基于前一种标准。

图 3-4 所示的是下面 HTML 编码的页面对应的 DOM 树,最底层结点就是文本信息结点。

```
    <!
    DOCTYPE html PUBLIC " - //W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1 - transitional.dtd">
    <html xmlns = "http://www.w3.org/1999/xhtml">
    <head><meta http - equiv = "Content - Type" content = "text/html; charset = gb2312">
<title>标题 - www.XXXYYY.com</title></head>
    <body>
    <div id = "top_frame" class = "name">
    <div id = "logoindex">
        样例
        <a href = "http://www.baidu.com">样例 - www.baidu.com</a>
    </div>
</div>
</body>
</html>
```

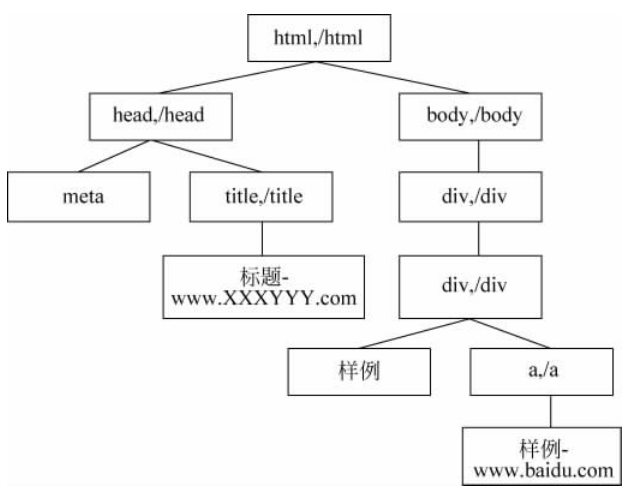

图 3-4　DOM 树

在 HTML DOM 树中,每个结点都拥有包含着关于结点某些信息的属性。这些属性是 nodeName(结点名称)、nodeValue(结点值)和 nodeType(结点类型)。特别地,在基于 DOM 树的结点信息提取中,通常需要对结点类型进行判断。W3C 标准中定义了如表 3-1 中所列出的 12 种结点类型。

表 3-1 HTML DOM 树中的结点类型

结点类型		描述	子结点
1	Element	代表元素	Element、Text、Comment、ProcessingInstruction、CDATASection、EntityReference
2	Attr	代表属性	Text、EntityReference
3	Text	代表元素或属性中的文本内容	None
4	CDATASection	代表文档中的 CDATA 部分(不会由解析器解析的文本)	None
5	EntityReference	代表实体引用	Element、ProcessingInstruction、Comment、Text、CDATASection、EntityReference
6	Entity	代表实体	Element、ProcessingInstruction、Comment、Text、CDATASection、EntityReference
7	ProcessingInstruction	代表处理指令	None
8	Comment	代表注释	None
9	Document	代表整个文档(DOM 树的根结点)	Element、ProcessingInstruction、Comment、DocumentType
10	DocumentType	向为文档定义的实体提供接口	None
11	DocumentFragment	代表轻量级的 Document 对象,能够容纳文档的某个部分	Element、ProcessingInstruction、Comment、Text、CDATASection、EntityReference
12	Notation	代表 DTD 中声明的符号	

在 HTML DOM 标准中,定义了所有 HTML 元素的对象和属性,以及访问它们的方法(接口)。换言之,HTML DOM 是关于如何获取、修改、添加或删除 HTML 元素的标准。因此,可通过若干种方法来查找希望操作的元素,一些常用于 Web 信息提取的 HTML DOM 属性和方法列举如下。

(1) 常用的 HTML DOM 属性。

① innerHTML 表示结点(元素)的文本值。

② parentNode 表示结点(元素)的父结点。

③ childNodes 表示结点(元素)的子结点。

④ attributes 表示结点(元素)的属性结点。

(2) 常用的 HTML DOM 方法。

① getElementById(id)表示获取带有指定 id 的结点(元素)。

② getElementsByTagName 表示获得指定 Tag 的结点(元素)。

③ appendChild(node)表示插入新的子结点(元素)。

④ removeChild(node)表示删除子结点(元素)。

例如,可以通过使用一个元素结点的 parentNode、firstChild 及 lastChild 属性,getElementById()和 getElementsByTagName()这两种方法,可查找整个 HTML 文档中的任何 HTML 元素或结点,从而完成整个 Web 页面的遍历。

HTMLParser是一个开源项目，集成了对HTML DOM树的支持。这是一个在SourceForge.net上比较活跃的项目之一，该软件包可以进行DOM树的生成、遍历等一系列操作。同时，HTMLParser还具有更多的功能，是一个对现有的HTML文本进行分析的快速实时的解析器。其中的Parser类可以从互联网上获取HTML文档或处理本地文件，这是获取网络资源和处理文件的入口。

基于HTMLParser解析页面的原理与思路如下。

(1) 提供一种途径对HTML(树)进行遍历。一棵DOM树只要知道这棵树的根结点就可以通过结点的父子关系遍历出整棵树，因为node本身保存有它的父结点和子结点信息。

(2) 提供一种接口，允许程序员在遍历的过程中对结点的属性和结点关系进行检查，从而判断是否为提取内容所需要的结点。

通常情况下，只需要获取某些特定的内容，也就是DOM树中某些结点的值，而不是整个页面的数据。因此，就需要在提取的过程中对HTML页面的标签(结点)进行一些筛选，如可以采用Filter、Visitor。

以Visitor方式访问HTML文档的解析过程如下。

(1) 用一个URL或页面对应文本String创建一个Parser。

(2) 定义一个Visitor。

(3) 使用Parser.visitAllNodeWith(Visitor)来遍历树中的结点。

(4) 获取Visitor遍历后得到的数据(或在遍历的过程中进行数据处理)。

使用Parser类来解析，从程序实现的角度看，包含如下两个过程。

(1) 定义解释器。

```
try {
    NodeVisitor visitor = new ContentVisitor (title, contentList);

    if (html ! = null) {
        Parser parser1 = Parser.createParser(html,"GBK");
        parser1.visitAllNodesWith(visitor);
    }
} catch (ParserException e) {
    e.printStackTrace();
}
```

(2) 定义结点访问器。在树的遍历中进行结点判断，这种判断方法是基于一定的规则，通过结点的属性、标签等信息来指定规则。例如，针对某个网络论坛的帖子回帖信息提取部分代码片段如下。

```
private class ContentVisitor extends NodeVisitor{

  private Title title;
  private List < Content > contentList;

  private Content content = null:

  public ContentVisitor(Title title,List < Content > contentList){
```

```
        this.title = title;
        this.contentList = contentList;
    }
    public void visitTag(Tag tag){
        Tag tag1,tag2,tag3,tag4,tag5;
        int p1,p2;
        //以下获得回帖记录
        if(tag)! = null&&((tag.getTagName().toLowerCase().equals(("div")) && "zwlist".equals
(tag.getAttribute("id"))) )) (
        try(
            string text = "";
            string author = "";
            string date = "";
            int num;

            NodeList list = tag.getChildren();
            for (int i = 1;i<=list.size();i++) (
                if (list.elementAt(i) instanceof TagNode == false) continue;
                tag1 = (Tag) list.elementAt(1);
                if (tag1.getTagName().toLowerCase().equals("div") * *("zwli clearfix").equals
(tag1.getAttribute("class"))) (
                    tag2 = (Tag)(tag1.getChildren().elementAt(3));  //从 1 开始.  div class
="zwlitx"
                    tag3 = (Tag)tag2.getChildren().elementAt(1);    //<div class="zwliext">
                    NodeList list2 = tag3.getChildren();
                    content = new Content();
```

3.1.4 基于 Jsoup 的页面解析

Jsoup(https://jsoup.org/)是一个基于 MIT 协议的开源软件,与 Python 中的 beautifulsoup 功能相似,是 Java 中常用的 HTML 处理库。Jsoup 支持 HTML5 规范,并且针对相同的 HTML 内容与主流浏览器解析出的 DOM 相同,支持 DOM/CSS/JQUERY 方法。

Jsoup 主要的功能有:支持从 URL、文件或字符串中抽取与解析 HTML;支持使用 DOM 树方法或 CSS 选择器方法查询与提取数据;支持 HTML 元素、属性与文本的修改;支持对用户提交的表单内容进行防注入攻击处理;支持输出整理并格式化的 HTML。

Jsoup 工作流程如图 3-5 所示。

(1) 通过 URL、HTML 文件或字符串的方式采集到包含 HTML 标签的 HTML 文本内容。

(2) 将 HTML 内容转换成 Document 对象,在这个过程中支持解析不完整、有错误的 HTML。例如,标签未封闭、某些必要结构标签缺少及其他常见的 HTML 语法错误。

(3) 获取到 DOM 之后即可进行 Document 对象操作。操作主要有提取结点集合、选中指定结点、读取选中结点与修改选中结点几大类。

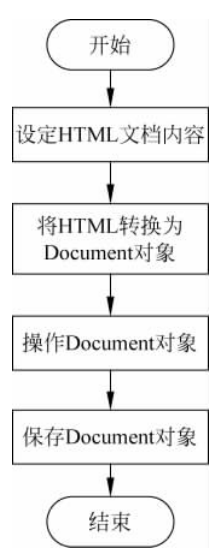

图 3-5 Jsoup 的工作流程

与 HTMLParser 的基本工作原理相似，Jsoup 也支持以 DOM 方法遍历网页内容。但是 Jsoup 还支持 CSS 标签选择器、类 Jquery 的方式获取指定样式的标签、属性、内容。

(4) 对象操作完毕后可以进行保存与输出。

Jsoup 支持对象操作的主要方法如表 3-2 所示。

表 3-2　Jsoup 支持对象操作的主要方法

大类	类别	方法名称	含　义
DOM 方法	查找元素	getElementById(String id)	按 ID 查找
		getElementsByTag(String tag)	按 Tag 查找
		getElementsByClass (String className)	按 CSS 类查找
		getElementsByAttribute (String key)	按属性查找
		siblingElements(), firstElementSibling(), lastElementSibling(); nextElementSibling(), previousElementSibling()	查找同胞元素
		parent(), children(), child(int index)	查找结点层级
	元素数据	attr(String key) /attr(String key, String value)	获取属性与设置属性
		attributes()	获取所有属性
		id()	获取 ID
		className()/classNames()	获取 CSS 类
		text()/text(String value)	获取文本内容/设置文本内容
		html()/html(String value)	获取元素内的 HTML/设置元素内的 HTML 内容
		outerHtml()	获取元素外的 HTML 内容
		data()	获取数据内容(如 script 和 style 标签)
		tag()/tagName()	获取 Tag 与名称
	操作 HTML 和文本	append(String html) prepend(String html)	在元素前后增加 HTML
		appendText(String text) prependText(String text)	在元素前后增加文本
		appendElement(String tagName) prependElement(String tagName)	在元素前后增加元素
		html(String value)	修改 HTML 内容
CSS/Jquery 选择器语法	查找元素	Element. select(String selector) Elements. select(String selector)	选择器的功能非常丰富，支持 Document/Element/Elements 的选择，并且与上下文相关，可实现指定元素的过滤与链式选择访问

在使用Jsoup采集一个页面之前首先需要分析页面结构组成，以东方财富网的页面(http://guba.eastmoney.com/news,000977,444182485.html)为例。

图 3-6 目标页面的目标元素

从图3-6的目标页面可以看出页面主要的内容分为主题帖与回复帖两个大类型，而要提取的目标元素是帖子部分。对于页面中相应的部分使用浏览器的调试功能(开发人员工具)，从图3-7的目标页面的结构分析可以发现，两者均在id为mainbody的DIV结点下，其CSS类为zwcontentmain和zwli clearfix，因此可以使用从DOM树选取结点的方式，也可以

图 3-7 目标页面的结构分析

使用 CSS 类选择的方式实现主题帖和回复帖的内容提取。

经过页面结构分析之后，可以给出提取主题内容和回帖内容的主要代码。方法 article 和 article_2 是两种不同的选择方法，执行后的结果一致，均为在屏幕打印输出当前页面的主题帖子的标题与内容。Dialog 方法在执行后则会在屏幕打印输出所有回复帖子的 DIV 的 HTML 内容，具体实践中还可以进一步对回复帖子的 HTML 内容进行分析，以提取更多信息。

```
package com.fudan.Jsoup;
import java.io.IOException;
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;

public class JsoupTest {
     static String url = "http://guba.eastmoney.com/news,000977,444182485.html";
     /**
      * 获取指定URL的帖子内容
      */
    public static void Dialog() {
        Document doc;
        try {
            //通过指定的URL,使用Get方法获取HTML内容,并转换为Document对象
            doc = Jsoup.connect(url).get();
            //通过getElementsByAttributeValue方法选择指定的回帖的Div内容
            Elements ListDiv = doc.getElementsByAttributeValue("class","zwli clearfix");
            for (Element element :ListDiv) {
                //输出DIV
                System.out.println(element.html());
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    /**
     * 获取页面的文章标题和链接
     */
      public static void article() {
        Document doc;
        try {
            doc = Jsoup.connect(url).get();
            //通过getElementsByAttributeValue方法选择指定的主题帖(zwcontent)的Div内容
            Elements ListDiv = doc.getElementsByAttributeValue("class"," zwcontent");
            for (Element element :ListDiv) {
                //通过指定的ID获取到标题(zwconttbt)与正文(zwconbody)
                Elements content_title = element.getElementsByID("zwconttbt");
                Elements content_body = element.getElementsByID("zwconbody");
                //解析标题与正文文本,并输出
                String contentTitleText = content_title.text().trim();
```

```
                System.out.println(contentTitleText);
                String contentText = content_body.text().trim();
                System.out.println(contentText);
            }
        } catch (IOException e) {
        //TODO Auto - generated catch block
        e.printStackTrace();
    }
}
/**
 * 获取页面的文章标题和链接
 */
public static void article_2() {
        Document doc;
        try {
            doc = Jsoup.connect(url).get();
            //使用 Select 方法取得主题帖(zwcontent)的 Div 内容
            Elements ListDiv = doc.select("div.zwcontent");
            for (Element element :ListDiv) {
                Elements content_title = element.getElementsByID("zwconttbt");
                Elements content_body = element.getElementsByID("zwconbody");
                String contentTitleText = content_title.text().trim();
                System.out.println(contentTitleText);
                String contentText = content_body.text().trim();
                System.out.println(contentText);
            }
        }
        } catch (IOException e) {
            // TODO Auto - generated catch block
            e.printStackTrace();
        }
    }
}
```

从前文可以看出，Jsoup 最重要的优点有两处，一处是能够处理不完整、不规范甚至结构异常的 HTML，另外一处则是支持功能强大的选择器。因而，与 HTMLParser 的单纯 DOM 树查找相比 Jsoup 的兼容性更好，并且能够更快速地选择到目标内容。而 Jsoup 与 Python 中的 BeautifulSoup 相比功能各有千秋，BeautifulSoup 额外支持正则表达式搜索与 XML 解析。实际工作中可以根据项目技术选型进行与生产环境的情况具体决策，当然选择一种开源架构还要看它是否得到了及时的维护和更新。

3.2 基于统计的 Web 信息抽取方法

对于 Web 爬虫而言，能爬取到各种各样的页面，大部分可能是事先并不知道的 Web 站点，因此在后续的提取过程中，并无法事先为这些站点上的页面设定正文的特征标志(Tag)，由此就不能直接利用前述所提到的方法。另外一种场景是，尽管爬虫爬取固定的某

些网站页面，但是网站也会经常升级改版，由此导致写好的程序针对新版失效了。

在处理这些场景下的页面内容提取，提取程序就需要有一定的智能性，能够自动识别某个 Web 页面上的正文位置，其前提是在没有人工参与的情况下。一种典型的方法是基于统计的页面信息提取，其基本步骤如下。

(1) 构建 HTML 文档对应的 DOM 树。

(2) 基于某种特征来构建基于 DOM 树的信息提取规则。

(3) 按照规则，从 HTML 中提取信息。

在这个处理流程中，最重要的是信息提取规则。规则的制定或生成方法有以下两种。

(1) 启发式方法。一般通过人工对 HTML 页面进行观察和总结，以 DOM 树所确定的基本组成单位为规则中的特征，人工估计其对应的特征值，从而形成启发式规则。常用的特征包括如下几点。

① 每个结点(Node)所包含的文本信息内容多少(TextSize)。

② 每个结点所包含的标签个数(tagCount)。

③ 每个结点内，有链接与无链接文本条内字符总个数的比值(LinkTextCountRatio)。

④ 链接锚文本的平均字符个数(LinkAvgCount)。

除此以外，还可以找到更多特征。根据这些特征，制定启发式规则来对页面上的内容进行识别和提取。例如，“IF LinkTextCountRatio(Ni)＞2 AND LinkAvgCount(Ni)＜＝6 THEN Ni”为导航区，这个规则对 DOM 树中的结点 Ni 从 LinkTextCountRatio 和 LinkAvgCount 两个特征做了限定，如果符合这个条件，则认为该结点对应于一个导航区。

类似的，考虑到页面中的正文区内通常会有较多的字符个数，即 TextSize 较大，并且所包含的超链接比较少，因此，可以根据这些特征制定判断规则。

显然，这种方法的有效性取决于规则的合理性，主要是特征值的具体取值。为了避免启发式规则中的人为因素，另一种方法就是采用机器学习的方法来计算最佳的特征值。

(2) 机器学习方法。这种方法通过人工选择大量的 HTML 页面，并对页面中的正文区域进行标注，再由程序计算正文结点中各种特征对应的特征值，以及其他类型结点对应的特征值。从而将正文结点的判断转换成为一个分类问题，即根据某些特征及特征值，判断结点是否为正文。这样的问题显然适合用机器学习方法来解决。

这种方法的优点是可以通过样本和机器学习的方法获得最佳的特征值，从而能避免在判断上的主观性，提升判断的准确性。其缺点是需要有一定量的人工标注样本。

要注意的是，前面提到的基于 HTML 结构的信息抽取方法中也涉及抽取规则，但是这些是比较简单的规则，如信息块的位置等，因此其适应性比这里所讨论的基于统计的方法要差一些。

3.3　其他互联网大数据的提取

互联网大数据除了来自 Web 页面外，还有其他一些来源，包括 Web 日志、网络探针、各种关系型和非关系型数据库等。

IIS、Tomcat、Apache 等知名的 Web 服务器软件，都允许对用户访问服务器的行为进行记录，并生成相应的日志文件。不同的 Web 服务器日志文件格式并不完全相同，但都记录

了用户访问的日期、时间、客户端 IP、请求方法、使用的协议、URI、状态等。这些日志中的内容提供了很多 Web 页面上无法获得的用户访问行为信息，如用户什么时候登录了服务器、使用 FTP 下载了什么文件、上传了什么内容、访问行为是否成功等。因此，这些日志文件也是互联网大数据中的一种与用户访问行为相关的数据源。

这种日志数据本身被组织成为一个文本文件，可以当作是关系型数据库来访问。但是由于该文件是按照用户访问网站的时间先后来写入日志的，因此不同用户的访问会混杂在一起。在进行提取时就需要进行用户的识别，通常可用的信息有客户端 IP、用户名及客户端 Agent 信息。此外，还应当对同一个用户在不同时间段的访问记录进行分割。图 3-8 所示的是部分来自 apache-tomcat-7.0.54 的日志文件内容，该文件在安装目录下的 logs 子目录中。

```
175.186.146.152 - - [09/Dec/2015:10:32:16 +0800] "GET /sqlinjection/Login.asp HTTP/1.1" 404 995
175.186.146.152 - - [09/Dec/2015:10:32:17 +0800] "GET /favicon.ico HTTP/1.1" 200 21630
175.186.146.152 - - [09/Dec/2015:10:32:29 +0800] "GET /sqlinjection/Login.jsp HTTP/1.1" 200 535
175.186.147.10 - - [09/Dec/2015:10:32:29 +0800] "GET /sqlinjection/results.jsp?username=ygs&passwor
175.186.147.10 - - [09/Dec/2015:10:32:34 +0800] "GET /sqlinjection/Login.jsp HTTP/1.1" 200 535
175.186.147.10 - - [09/Dec/2015:10:32:47 +0800] "GET /sqlinjection/results.jsp?username=u1&password
175.186.146.152 - - [09/Dec/2015:10:32:50 +0800] "GET /sqlinjection/results.jsp?username=rr&passwor
```

图 3-8　apache-tomcat-7.0.54 的日志文件

对这种数据源的处理一般需要 3 个步骤，即数据清理、用户识别、会话识别。特别要注意的是数据清理环节，一般需要涉及以下几方面。

(1) 图片、脚本和样式：从第 2 章介绍爬虫原理可以知道，页面上所包含的图片等资源需要一次独立的 Web 访问，因此它们也会在日志中生成一条记录。

(2) 弹出式广告：在用户打开网页时自动弹出，并不能反映用户主观访问意图，应当删除。

(3) 用户访问失败时的记录：返回代码会保存在记录中，错误如果是由服务器引起的，也无法反映用户意图。

(4) 类爬虫应用：有时客户端会有类似于爬虫功能的应用程序，根据一定规则自动在客户端后台抓取页面，从而产生了 Web 日志记录，显然这些也非用户主动发起的访问。

基于这些日志文件可以开展的研究包括用户行为聚类、Web 浏览服务预取、页面推荐、恶意行为检测等。

另一种数据源是数据库中的数据，这部分数据主要是存储在互联网应用的相关业务数据库中。对此，数据的提取广泛采用 ETL 技术，其最初是用于从 OLTP 抽取数据到数据仓库中。目前有很多流行的 ETL 工具可用，有利于提高工作效率。Kettle 就是一款开源的 Java 编写的 ETL 工具，提供了一个图形化的用户环境，来管理来自不同数据库的数据，它同时也提供了很多控件，可以在程序中调用集成。4.3 以上版本的 Kettle 也针对 Hadoop 中的 Hbase、NoSQL 数据进行了支持。

而从网络探针获得的数据进行内容提取要复杂得多。这种数据格式与网络协议有很大关系，图 3-9 所示的是一个来自 WireSHARK 分析器截获的网络流量数据。图中最下面部分就是原始数据，上面是经过该软件解析出来的协议字段，显示的是一个 HTTP 请求的头部信息。

显然，在程序自动处理这种类型数据时，需要从原始的网络数据流开始。首先应当把这些数据进行协议还原，这里是指针对 TCP、UDP 等标准协议的还原，提取出数据包的相关

信息，如时间戳、源地址、目的地址、源端口和目的端口等。由于来自所有的访问、所有应用、所有不同客户端的网络访问都混杂在同一个数据流中，因此，接下来，就应当对解析出来的格式化流量再进一步区分，按照源地址、目的地址、源端口和目的端口来标识一个访问的连接，将所有同类型的连接的数据包进行整合，才能恢复出连接上所传输的数据。当然这些数据是属于应用层的编码，因此最后需要对判断应用层的应用程序，根据相应的协议对所恢复出来的数据进行内容上的解码，获得用户可以理解的信息内容。

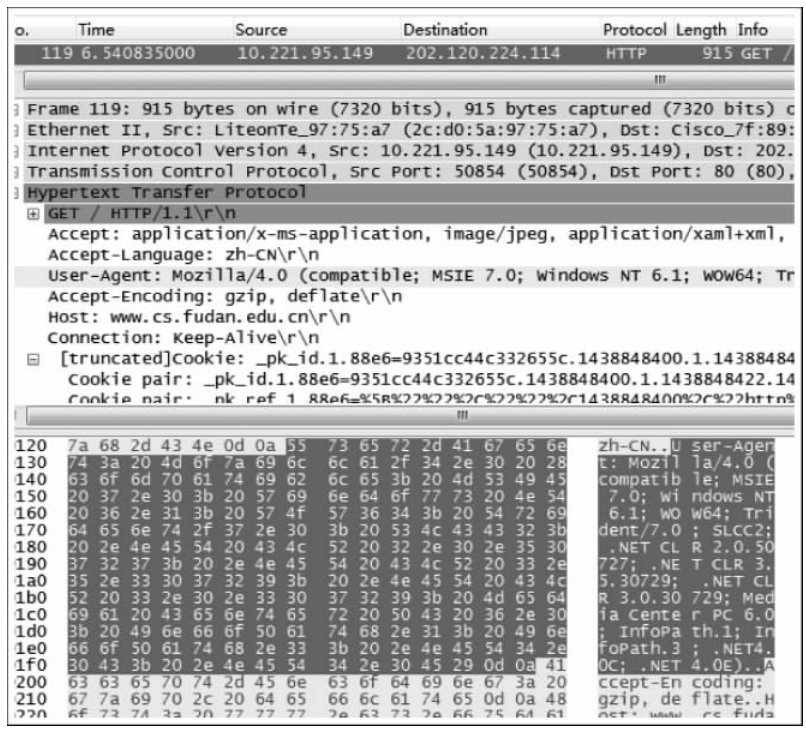

图 3-9　网络探针获得的流量数据

3.4　阿里云公众趋势分析中的信息提取应用

以 Web 信息内容的提取方法为基础的应用，在实际中有很多种，如搜索引擎系统，下面以阿里云大数据平台中的“公众趋势分析”服务（https://data.aliyun.com/product/prophet）为例构建一个以 G20 峰会为主题的互联网信息提取应用。“公众趋势分析”服务是一个类似于定义网络舆情服务的在线系统，其数据源是基于全网公开数据，每天更新 20 亿网页，覆盖各类网站、论坛、贴吧、微博、微信、自媒体等，用户还可以定义特殊站点，由该服务系统负责爬行。

在这个简单的应用中，要利用“公众趋势分析”服务提供的功能，构建一个关于 G20 的网络舆情信息监测应用。主要功能包括舆情的关键词、转发关系、涉及的媒体等。

首先定义 G20 的相关主题，这与第 2 章介绍主题爬虫提到的第一种方式相同。在该服务中，提供了一个如图 3-10 所示的主题定义界面，其中指定该主题的相关关键词，可以通过“与”、“或”

两种方法来设定关键词，此外，从图中也可以看出，还可以指定信息来源，即网站类型。

图 3-10　G20 主题定义

在设定完成后，该服务即从最近的 Web 信息及新生成的 Web 信息中匹配符合关键词要求的结果，并按照关键词、媒体（网站）类型进行聚合，显示各种聚合结果中的匹配到的记录数，如图 3-11 所示。

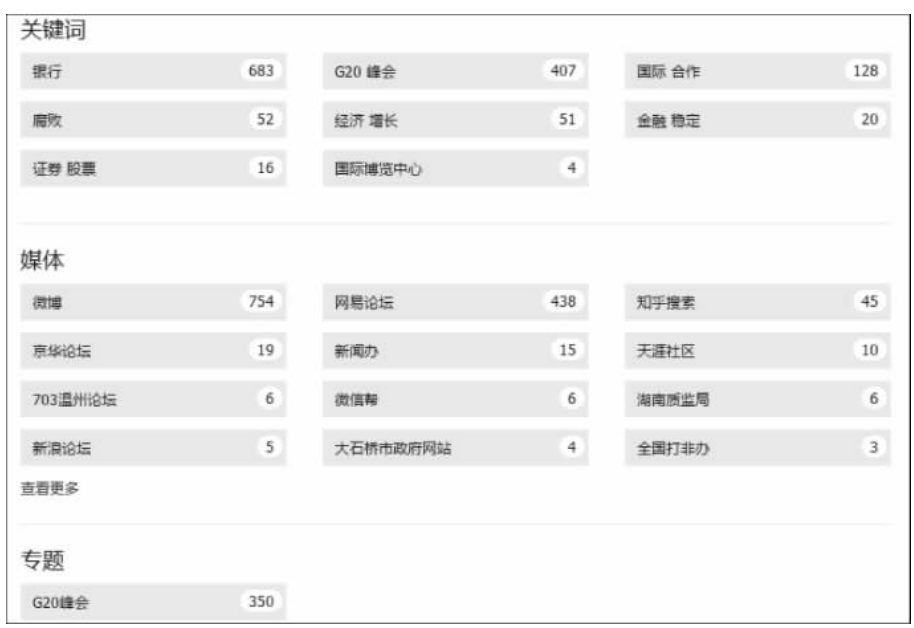

图 3-11　G20 主题聚合的统计结果

通过浏览相应的结果，可以发现匹配 G20 主题的相关关键词存在于页面的标题或主体内容中。显然在匹配之前，该服务对获取的 Web 页面信息进行了分析和提取。一些匹配的

Web 页面，如 https://zhuanlan. zhihu. com/p/22359398、http://bbs. money. 163. com/bbs/kgu/616063378. html、http://bbs. money. 163. com/bbs/kgu/616096706. html。

特别地，对于 Web 信息提取来说，也不局限于对内容的提取。针对各种不同类型网站所提供的信息不同，还可以获得更加丰富的信息。例如，对于微博而言，除了博文信息外，还有微博的转发、评论、朋友关系等信息，而此类信息的提取在技术手段上与本章所叙述的基本原理是一致的。“公众趋势分析”服务就对微博的转发行为信息进行了提取，从而可以在此基础上进一步提供诸如微博转发关系的可视化信息。图 3-12 所示的是 G20 中某个博文的转发关系图。

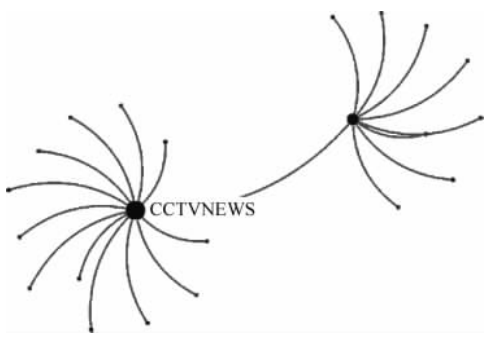

图 3-12　G20 中某个博文的转发关系

除了通过用户界面进行主题定制和阅读外，该服务系统也提供了 API 调用的方式，允许用户进行二次开发，从而可以更加灵活地集成到应用系统中。开发者可以通过 API 数据接口进行：关键词专题配置、关键词配置、接收实时抓取数据、微博传播路径分析。每种功能对应一种接口(Interface)，而 API 执行结果返回的是 JSON 格式的数据集，这些信息正是基于 Web 信息提取的结果。图 3-13 所示的是舆情数据 GET 的结果。


```
"messages":[],
    "result":{
        "records":[{
            "id":11175897,
            "monitorKeywords":"*",
            "monitorKeywordId":12359,
            "monitorTopicId":0,
            "subject":"《疯狂动物城》尼克狐的·····",
            "translateSubject":null,
            "translateDescription":null,
            "description":"*文章为作者独立观点，不代表虎嗅网立·····",
            "url":"http://www.huxiu.com/article/147005/1.html?f=index_feed_article",
            "createdAt":"2016-04-28T04:53:05.000Z",
            "pubTime":"2016-04-28T04:51:04.000Z",
            "from":"虎嗅网",
            "langType":"ch",
            "filterStatus":1,
            "wbType":2,
            "wbFansCount":0,
            "wbRepostCount":0,
            "wbCommentCount":0,
            "wbLikeCount":null,
```


图 3-13　舆情数据返回结果示例(部分)

3.5 互联网大数据提取的挑战性问题

Web 页面内容提取是获取 Web 上海量信息的基础，然而随着技术的不断发展，以及站点的防范意识的增强，Web 页面信息内容提取也面临着多种多样的挑战，需要在不断的研究中提出新的解决办法。

由于传统的 Web 页面提取可以认为是在相同的或者相似的站点，使用规则的模板进行内容提取与过滤。例如，同样的 Discuz 论坛，可以使用相似的模板规则进行论坛帖子链接采集与帖子内容提取。但随着动态页面技术的不断发展、移动终端的广泛使用，以及 HTML5 的逐步推广，目前页面信息内容存在两个趋势：一是页面动态加载内容的增多；二是多媒体内容的增加。此外还有部分站点为了防止内容信息被采集或自身的技术升级，也会导致其页面结构改变等情况，对采集者造成困扰，从而要求 Web 内容提取技术不断升级。

具体的情况如下。

(1) 多媒体内容在 HTML5 中已经是原生支持，但是对于其源地址和源文件的保护各网站实现方式不一，如何通过页面数据进一步获取到非文本的内容需要费一番力气，如从爱奇艺获取视频连接，但是需要排除其中的广告部分。

此外，还有部分网站为了保护其内容，将部分文本内容以非文本内容的方式展示，这样本文内容的解析也是一大难点。例如，Docin、百度文库等站点的内容信息均以嵌入的多媒体方式展示。

(2) 网站页面布局不断改版升级是频繁发生的，虽然基于统计的提取方法中引入了机器学习来自动学习提取规则，但是规则所涉及的特征仍是事先定义好的，因此，在适应性方面的能力也有待进一步提升。

(3) 基于 DOM 树的各种提取方法对于程序员来说还是非常方便，但是前提是需要对 DOM 树的结构和编码组成要了解得很清楚，如何以一种更加直观可视化特征来定义所要提取的内容及其位置，对于提高编程效率是十分重要的。这将是 HtmlParser、Jsoup 之后的提取架构需要考虑和突破的方向。

除了来自 Web 页面提取的挑战外，在提取通过网络探针方式获取的网络大数据时也面临较大的挑战。其主要问题在于，随着各种互联网应用的发展，未知的协议数据格式将会增加，流量越来越高，使得从流量中进行内容提取变得更加困难。

而对于 ETL 的挑战则在于，在大数据处理要求下，越来越多的大数据应用将会迁移到云环境中，从 ETL 抽取到的海量业务数据传输到云环境，将是对此类应用的很大障碍。但是这个问题的解决应当延伸到 ETL 提取阶段，而不能仅仅看作是一个数据迁移的问题。在 ETL 中根据业务中的数据关系逻辑，通过一些推理计算决定 ETL 的最小数据集，可以避免冗余的数据抽取出来。

思考题

1. 什么是 DOM 树，它的逻辑结构是怎样的？W3C 中定义的对这棵树的操作都有哪些？

2. 比较 HTMLParser 和 Jsoup 在 Web 页面提取方法上的差异。

3. 比较基于 HTML 结构的信息抽取方法和基于统计的 Web 信息抽取方法中的规则，说说它们的相同点和不同点。

4. 为了基于新浪微博构造某个博文的转发关系图，应当进行哪些信息提取？

5. 选择一个新闻 Web 页面，用 Jsoup 编写一个程序，实现对其中的新闻正文内容进行提取。