**摘　要**

我们的分布式爬虫的系统,可以快速，通用爬取各大电商,博客网站。

[摘　要 1](#_Toc492748946)

[1 开发技术方案 2](#_Toc492748947)

[1.1网页采集策略 2](#_Toc492748948)

[1.1.1 广度优先策略 2](#_Toc492748949)

[1.1.2深度优先策略 3](#_Toc492748950)

[1.1.3最佳优先策略 3](#_Toc492748951)

[1.1.4本软件的基于列表页爬取的策略 3](#_Toc492748952)

[1.2网页清洗: 4](#_Toc492748953)

[1.3 url去重技术 4](#_Toc492748954)

[1.3.1 url压缩 4](#_Toc492748955)

[1.3.2布隆过滤器（Bloom Filter） 5](#_Toc492748956)

[1.3.3本软件采用key-value的数据库和布隆过滤器的结合对url进行存储 5](#_Toc492748957)

[1.4深层网络采集技术(主要在 crawlerSys 项目下的downloader包下) 5](#_Toc492748958)

[1.4.1动态页面這染技术 5](#_Toc492748959)

[1.4.2本软件 6](#_Toc492748960)

[1.5防止反爬 6](#_Toc492748961)

[1.5.1本文采用的防止反爬策略是 7](#_Toc492748962)

[1.6 分布式调度 7](#_Toc492748963)

[1.6.1 Zookeeper 概述 7](#_Toc492748964)

[1.6.2 RPC 远程调用协议概述 8](#_Toc492748965)

[1.6.3本软件的分布式调度实现就是采用 8](#_Toc492748966)

[zookper及hadoop自带的RPC组件混合使用,zookper负责任务的调用，redis主要利用setnx 的特性实现分布式锁, 8](#_Toc492748967)

[1.6.3.1任务调度 8](#_Toc492748968)

[1.6.3.2 master选举 8](#_Toc492748969)

[1.7网页结构化 9](#_Toc492748970)

[1.7.1简单树匹配(主要适合于列表页的结构化,在 9](#_Toc492748971)

[crawlerSys 项目下com.crawler.utils.SdeUtils) 9](#_Toc492748972)

[1.7.2 文本去重算法 11](#_Toc492748973)

[1.7.3 利用网络监控获取评论源,自动获取评论的ajax请求(crawlerManage项目下com.crawler.utils.ReviewUtils) 11](#_Toc492748974)

[1.7.4 基于列表页结构化算法（就是部分树对齐） 12](#_Toc492748975)

[1.7.5 基于相同模板结构化算法(crawlerManage项目下com.crawler.itemUtils) 12](#_Toc492748976)

[1.7.6 正文抽取，摘要，关键字提取 14](#_Toc492748977)

[自动义包装器：(主要的实现在AutomaticExtractor) 14](#_Toc492748978)

[2 系统分析与设计 15](#_Toc492748979)

[2.2系统架构 16](#_Toc492748980)

[2.3博客爬取流程 16](#_Toc492748981)

[2.4基于相同模板爬取流程 19](#_Toc492748982)

[3.6结构化数据展示 21](#_Toc492748983)

# 1 开发技术方案

## 1.1网页采集策略

万维网中的网页之间通过url链接在一起，形成了一张巨大的图，网络爬虫

的采集过程就可以看着是对这张图的一个遍历问题。爬虫行进的路径，就是我们采集数据的路径。当前主要的网页爬行策略主要有：广度优先搜索、深度优先搜索、最佳优先搜索策略。

### 1.1.1 广度优先策略

广度优先策略是最常用的搜索算法，其基本思想是：从起始网页作为定点

开始，沿着数的宽度，一层一层的遍历节点，也称为分层处理。该算法设计和

实现比较简单且能获得较高的网页覆盖率，但效率比较低。如果想尽可能的覆

盖较多的网页，广度优先搜索是较好的采集策略。

### 1.1.2深度优先策略

深度优先是相对于广度优先搜索的策略，其基本思想是：从起始网页开始，

找到网页中的一个url进行爬取，在采集到的这个页面中找一个url再继续爬取，

一直进行下去，直到最后一个页面没有url，然后回到源节点，找到第二个url,

继续上面的采集步骤。该算法的优点是实现简单，但是由于图的结构复杂，容

易出现死循环，因此一般都会配合URL去重策略使用避免死循环。并且网页的采集顺序比较随机，会导致同一频道下的数据采集前后间隔很大，但此方法在短

时间内对站点的覆盖率较高。若果希望短时间内获取不同站点数据时，采用此

采集策略效果较好。

### 1.1.3最佳优先策略

广度优先和深度优先策略不能满足所有的爬虫的需求，有时我们需要对一些

些重要的站点或频道进行优先采集，这时就需要使用最佳优先策略来完成我们

将定的任务。最佳优先策略的基本思想是：通过用户预设的权重评分算法，对

待采集的网页进巧评分计算，优先采集得分较高的网页。理论上，最佳优先策

略是非常理想的采集策略，他可以通过调整算法计算权重，根据要求进行搜索

采集。这样可有效地的保证主要信息被优先采集，但是也存在不足，权重计算需要消耗机器资源，同时如何设计一个有效地算法也是一个难点。

### 1.1.4本软件的基于列表页爬取的策略

经过测试，发现大部分电商网站都是利用get提交请求,所以我们可以提取首页的导航关键字，替换列表url的关键字的key,再更具自动翻页算法进行翻页，这样可以大大减少重复次数,也可以对数据进行深层次的挖掘

## 1.2网页清洗:

dom文档树(Document Object Model,文档对象模型)对网页的层次结构提供了具体的描述方法，把web编程等与网页相关的操作提供了一个简单方便的途径。如今基本上所有的web编程等都是把一个网页的html标签视为一个文档树结构。由于本文方法后续的所有研究工作都是基于DOM文档树的结构基础上进行的，所有本文在对一般dom文档树的生成方法上做了一些优化处理，使得html源文件解析后的dom树简洁明了操作

对dom文档树的观察可知很多标签是完全可以删除的，其结果完全不会影响html的主要数据内容。对此我们定义一个解析html源文件过程中可以删除的标签集合(hide/select/input/style,scripy等相关标签),在进行dom 操作前,先把没用的信息清除干净

## 1.3 url去重技术

由于一个网页的url可能链接在其他多个网页中，因此爬虫就会重复的获取这类url，为避免重复采集，因此爬虫需要对待采集的url进行去重处理，将新的未采集过的url加入到采集队列。在一个大型的网络爬虫的系统中，url的数量是巨大的，可被称为海量。如此巨大的url数量简单的导入计算机内存已经不可能了，即使使用数据库存储按照URL进行查也变得非常耗时。当采集数据达到一定量时，去重URL将成为影响整系统效率的关键瓶颈。因此我们就需要设计一个有效地大数量的排查机制了解决这个问题当前主要的方法有：

### 1.3.1 url压缩

原始的url字串很长，这导致了存储url的开销，因此常规的压缩方法是采用md5或者simhash对url进行签名形成一个16byte的字符串，在去重时采用查询url的编码是否存在来判断链接是否己经采集过。

### 1.3.2布隆过滤器（Bloom Filter）

由布隆1970年提出，用于检测一个元素是否属于某个集合，具有很高的空间和时间效率。其基本思想是：通过一组哈希（hash）函数将一个元素映射到数组的一些位，当这些位都为１时，表示该元素存在于集合中，否则说明没有存在于集合中。它的查询及检测时间复杂度

都是常数级，因此效率非常高。但是随着插入元素的逐渐増多，其准确性会慢慢降低，也就是说会存在误判的情况。在初始化时，预估一个足够大的位数组，将有效降低误判率，如果在可接收范围，可考虑使用布隆过滤器。

### 1.3.3本软件采用key-value的数据库和布隆过滤器的结合对url进行存储

利用了redis 的bitmap 特性，结合布隆过滤器的概念

**主要代码在 crawlerSys 项目 com.crawler.lock下**

## 1.4深层网络采集技术(主要在 crawlerSys 项目下的downloader包下)

随着web2.0的发展，互联网上使用ajax技术的网站越来越多，大量的动态页面需要表单提交、动态触发、动态這染才能呈现，给数据采集工作带来了很大的挑战。传统的基于静态网页的爬虫工具己经无法采集此类网页了。针对该类网页进行采集的技术被称为深层网络采集技术。如何改进传统爬虫，使之支持动态页面采集，也是本文研究的一个方面。

### 1.4.1动态页面這染技术

当前采集动态网页主要由两种方法：

**1**使用封装浏览器内核开发的页面這染器，通过提供相关接口对html及dom进行操作。具有代表性的如:jrex,jexplorer,webkit等。

**2**使用脚本解释引擎，绑定dom，执行脚本更新页面，然后对最终渲染的dom进行相关采集操作。具有代表的性的如：rhino,spiderMonkey,google,v8,

Web2.0中前端页面的逻辑越来越复杂，在开发工程中已经使用了大量的javacript库如jquery，dojo等），运使得基于脚本引擎的方式出现了一些兼容缺陷，因而在广泛的实际应用中多使用基于浏览器内核的方式来进行采集处理。

**3**以下是几种流行浏览器搜索引擎对比

|  |  |  |  |
| --- | --- | --- | --- |
| Driver 类型 | 优点 | 缺点 | 应用 |
| 真实浏览器 | 真实模拟用户行为 | 效率,稳定性低 | 兼容性测试 |
| htmlunit | 速度快 | Js引擎不是主流浏览器支持的 | 包含少量js的页面 |
| phantomjs | 速度中等，模拟行为接近真实 | 不能模拟特定浏览器的行为 | 非GUI的功能性测试 |

### 1.4.2本软件

因为博客的js脚本并不复杂,所有爬取博客时,所以本软件使用的浏览器引擎是htmlunit,爬取电商时使用的是phantomjs和firefox

## 1.5防止反爬

网络爬虫通过http对网站进行访问，这对网站来说就相当于一个在线用户的访问，如果爬虫在同一时间内对网站的多个网页进行访问，这就会消耗网站服务器大量的资源，这就会导致网站降低网站对真实用户的服务能力，严重的情况下甚至会导致网站瘫痪。因此，网络爬虫的采集效率不是越快越好，在采集数据时也应该考虑到网站本身的服务效能，进行礼貌的采集。般从以下两个方面来进行处理：

1. 严格控制爬虫访问的间隔时间和单次请求数量。
2. 模拟浏览器请求头 如(accept ,referrer,cookie,user-Agent)等
3. 使用代理ip

## 1.5.1本文采用的防止反爬策略是

1 首先独立一台服务器，先从西刺代理，全网代理,快代理等代理网站,爬取前5页

最新发布的代理Ip,通过验证算法检测代理的可用性,可用就持久化到数据库。

2 然后每隔半小时验证数据库的代理是否可用,把已经失效的IP删除掉，当数据库的代理小于的阀值时,再重新爬起代理网站



## 1.6 分布式调度

## 1.6.1 Zookeeper 概述

ZooKeeper是一个分布式的，开放源码的分布式应用程序协调服务，是Google的Chubby一个开源的实现，是Hadoop和Hbase的重要组件。它是一个为分布式应用提供一致性服务的软件，提供的功能包括：配置维护、域名服务、分布式同步、组服务等

## 1.6.2 RPC 远程调用协议概述

RPC（Remote Procedure Call Protocol）——远程过程调用协议，它是一种通过网络从远程计算机程序上请求服务，而不需要了解底层网络技术的协议。RPC协议假定某些传输协议的存在，如TCP或UDP，为通信程序之间携带信息数据。在OSI网络通信模型中，RPC跨越了传输层和应用层。RPC使得开发包括网络分布式多程序在内的应用程序更加容易。

RPC采用客户机/服务器模式。请求程序就是一个客户机，而服务提供程序就是一个服务器。首先，客户机调用进程发送一个有进程参数的调用信息到服务进程，然后等待应答信息。在服务器端，进程保持睡眠状态直到调用信息到达为止。当一个调用信息到达，服务器获得进程参数，计算结果，发送答复信息，然后等待下一个调用信息，最后，客户端调用进程接收答复信息，获得进程结果，然后调用执行继续进行

## 1.6.3本软件的分布式调度实现就是采用

## zookper及hadoop自带的RPC组件混合使用,zookper负责任务的调用，redis主要利用setnx 的特性实现分布式锁,

保证 每个任务的隔离性

### 1.6.3.1任务调度

爬虫服务器在启动时先注册并监听各种任务临时节点，客户端通过RPC远程访问master，master再调度其他从节点，来进行任务的调度

### 1.6.3.2 master选举

在分布式环境中，相同的业务应用分布在不同的机器上，有些业务逻辑（例如一些耗时的计算，网络 I/O 处理），往往只需要让整个集群中的某一台机器中行执行，其余机器可以共享这个结果，这样可以大大减少重复劳劢，提高性能，于是这个 master 选丼便是这种场景下的碰到的主要问题。利用 ZooKeeper 的强一致性，能够保证在分布式高并发情况下节点创建的全局唯一性，即：同时有多个客户端请求创建 /currentMaster 节点，最终一定只有一个客户端请求能够创建成功。利用这个特性，就能很轻易的在分布式环境中迚行集群选取了。另外，这种场景演化一下，就是动态 Master 选举。这就要用到 EPHEMERAL\_SEQUENTIAL 类型节点的特性了。上文中提到，所有客户端创建请求，最终只有一个能够创建成功。在这里稍微变化下，就是允许所有请求都能够创建成功，但是得有个创建顺序，于是所有的请求最终在 ZK 上创建结果的一种可能情况是这样：/currentMaster/{sessionId}-1 , /currentMaster/{sessionId}-2 , /currentMaster/{sessionId}-3 ….. 每次选取序列号最小的那个机器作为Master，如果这个机器挂了，由于他创建的节点会马上小时，那么之后最小的那个机器就是 Master 了



## 1.7网页结构化

### 1.7.1简单树匹配(主要适合于列表页的结构化,在

### crawlerSys 项目下com.crawler.utils.SdeUtils)

**我们软件在进行子树的相似度比较时用到了简单树匹配，所以在这里先介绍一下**

简单树匹配（Simple Tree Matching， STM）的目标是找到两棵树间的最大匹配。两棵树间的最大匹配定义如下：设A和B是两棵树，而i和j分别是A和B中的两个节点。两棵树间的一个匹配定义为一个映射M，使得对每一个节点对（i，j）属于M（这里i和j都不是根节点），都有（parent（i），parent（j））属于M。其中，拥有最多节点对的匹配被称为A和B之间的最大匹配。需要注意，最大匹配可能不止一个。用W（A， B）表示树A和树B之间的简单树匹配中节点对的个数，它的形式化定义如下：其中，RA和RB分别是树A和树B的根节点，Ak是树A的第一层子树的第k个节点，Bn是树B的第一层子树的第n个节点，m（<A1,...,Ak>,<B1,...,Bn>）是指树A的第一层子树和树B的第一层子树的最大匹配中节点对的个数。W（A， B）的形式化定义是说，当树A和树B的根节点不同时，W（A， B）等于0；当树A和树B的根节点相同时，W（A， B）等于树A的第一层子树和树B的第一层子树的最大匹配中节点对的个数加1。我们还需要对m（<A1,...,Ak>,<B1,...,Bn>）作进一步的定义。先看m（<A1,...,Ak>,<B1,...,Bn>）的形式化定义：m（<>, <>）= 0 // <>代表一个空的子树列表

m（s，<>）= m（<>, s）= 0 // s匹配任意非空子树列表

m（<A1,...,Ak>,<B1,...,Bn>）=max（

m（<A1,...,Ak-1>,<B1,...,Bn-1>） + W（Ak，Bn），

m（<A1,...,Ak>,<B1,...,Bn-1>），

m（<A1,...,Ak-1>,<B1,...,Bn>）

）

这个形式化定义的具体含义是：

（1）当两个子树列表任意一个为空时，匹配数为0；

（2）两个子树列表的最大匹配可以看作第一个子树列表的前k个子树组成的子树列表和第二个子树列表的前n个子树组成的子树列表的最大匹配；这时，我们有三个选择:

a.匹配Ak和Bn，这时m（<A1,...,Ak>,<B1,...,Bn>）变为m（<A1,...,Ak-1>,<B1,...,Bn-1>） + W（Ak，Bn）；

b.不匹配Bn，这时m（<A1,...,Ak>,<B1,...,Bn>）变为m（<A1,...,Ak>,<B1,...,Bn-1>）；

c.不匹配Ak，这时m（<A1,...,Ak>,<B1,...,Bn>）变为m（<A1,...,Ak-1>,<B1,...,Bn>）。

然后，从这三个中选出最大的那个，就得到最大的m（<A1,...,Ak>,<B1,...,Bn>）了。我们可以看出，求m（<A1,...,Ak>,<B1,...,Bn>）的过程和求字符串编辑距离的过程是很相似的，实际上，我们同样可以通过动态规划的方法求m（<A1,...,Ak>,<B1,...,Bn>），并且同样可以通过得到二维数组（这里是m）对齐两个子树列表，由于在从字符串编辑距离到字符串对齐一文中已有详细阐释，这里就不再讲动态规划的具体实施过程和对齐子树列表了。不过，求m（<A1,...,Ak>,<B1,...,Bn>）也有它的特有之处，那就是里面会递归计算W（Ak，Bn）

Java代码:

**public** **int** simpleTreeMatching(Element A, Element B) {

String aTag = A.tagName();

String bTag = B.tagName();

**if** (!aTag.equals(bTag)) {

**return** 0;

}

// 只有标签节点，不包含文本节点

Elements aChild = A.children();

Elements bChild = B.children();

**int** aChildNum = aChild.size();

**int** bChildNum = bChild.size();

**if** (aChildNum == 0 || bChildNum == 0) {

**return** 1;

}

**int**[][] m = **new** **int**[aChildNum + 1][bChildNum + 1];

**int**[][] w = **new** **int**[aChildNum + 1][bChildNum + 1];

// 当A没有子树时，只有０个匹配

**for**(**int** i = 0; i < aChildNum + 1; i++) {

m[i][0] = 0;

}

//　当B没有子树时，只有０个匹配

**for**(**int** j = 0; j < bChildNum + 1; j++) {

m[0][j] = 0;

}

**for**(**int** i = 1; i < aChildNum + 1; i++) {

**for**(**int** j = 1; j < bChildNum + 1; j++) {

w[i][j] = simpleTreeMatching(aChild.get(i - 1), bChild.get(j - 1));

m[i][j] = Math.max(Math.max(m[i][j-1], m[i - 1][j]),

m[i-1][j-1] + w[i][j]);

}

}

**return** m[aChildNum][bChildNum] + 1;

}

### 1.7.2 文本去重算法

任何一段信息文字，都可以对应一个不太长的随机数，作为区别它和其它信息的指纹我们这里的做法是文章抽取特征词，压缩为md5指纹。利用这些指纹进行布隆过滤。

### 1.7.3 利用网络监控获取评论源,自动获取评论的ajax请求(crawlerManage项目下com.crawler.utils.ReviewUtils)

对于获取网站评论、价格等,由于都是采取Ajax技术异步传输数据，采用正常的爬取方案获取数据比较困难，我们采取phantomjs的网络监控功能，截取Ajax异步传输请求,然后伪装request请求,对数据进行爬取

商品的评论信息一般都是通过ajax调用,而且商品的评论数太多,某些可达到数十万条,如果我们利用浏览器引擎一点点的解释,显然是不合理,所有我们要通过网络监控来获取评论ajax的请求地址,

1.9.1我们来看一下,以下分别是京东，天猫的评论ajax的请求地址,我们可以发现只要我们修改商品id，和页数,要可以模拟出请求地址获取评论

https://club.jd.com/comment/productPageComments.action?callback=fetchJSON\_comment98vv76842&productId=2655649&score=0&sortType=5&page=1&pageSize=10&isShadowSku=0

https://aldcdn.tmall.com/recommend.htm?\_ksTS=1492653358687\_651&callback=

jsonpAld03194&appId=03194&itemId=39272696835&categoryId=50010535&sellerId=2103369914&shopId=111000091&brandId=111364825

**1.9.2** 经过观察我们发现，评论请求地址一般会包含 page,comment ,review 其中一项,和体积 最大

在本软件我们使用phantomjs自带的网络监控使用这一个功能,下面展示过滤的代码：

page.onResourceReceived = function(resource) {

var ct = resource.contentType;

if(ct.indexOf("text/html")!=-1){

if(ct.indexOf("page")!=-1||!ct.indexOf("comment")!=-1||!ct.indexOf("review")!=-1){

console.log(resource.url+"::"+resource.bodySize);

}

}

};

### 1.7.4 基于列表页结构化算法（就是部分树对齐）

我们都知道,列表页排列商品是用极度相似结构的dom树组成,所以我们只要把html dom树中相似的子树放在集合中,然后根据商品列表的特征(价格,评论数等)帅选出最合适的集合,然后进去特征抽取,从而进行结构化

**步骤：**

1 把爬取的列表页进行清洗在转化为标签树(利用jsoup组件转化为标签树)

2 然后遍历html dom子树,每一课子树都与该html dom树的所有子树进行相似度比较,把相似度超过某个阀值的的dom树放在一个集合中(集合中的dom子树不再进行遍历)

1. 根据然后根据商品列表的特征(价格,评论数等)帅选出最合适的集合
2. 然后根据特征进行抽取,抽取url,价格,评论数,店铺url等

### 1.7.5 基于相同模板结构化算法(crawlerManage项目下com.crawler.itemUtils)

相同模板意味着有大量的信息相同。

1 对于每一类相同的模板,我们会把该类第一页当作模板页

2 之后的每个页面都先清洗然后再 与该类模板页面进行对比,去除相同的部分,

3 然后根据要抽取数据的特征写正则表达式抽取

C:\Users\Administrator\AppData\Roaming\Tencent\Users\825540760\QQ\WinTemp\RichOle\Z0)}BA]D5N9M7W4JA$ST@JA.png



C:\Users\Administrator\AppData\Roaming\Tencent\Users\825540760\QQ\WinTemp\RichOle\]BO%YSK%FZ}(3V~P8PZL0HY.png

C:\Users\Administrator\AppData\Roaming\Tencent\Users\825540760\QQ\WinTemp\RichOle\E3T6`FX5$8E8[5{PJ`9)H{S.png

C:\Users\Administrator\AppData\Roaming\Tencent\Users\825540760\QQ\WinTemp\RichOle\)WTUVVMDRLDEI6{]3YC08DU.png

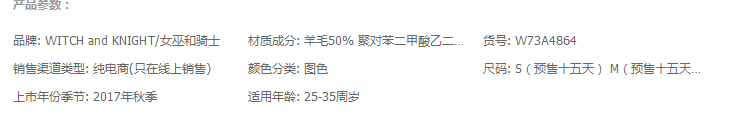
C:\Users\Administrator\AppData\Roaming\Tencent\Users\825540760\QQ\WinTemp\RichOle\J~VL5PS%~3RIGCXB6J50HQ8.png



C:\Users\Administrator\AppData\Roaming\Tencent\Users\825540760\QQ\WinTemp\RichOle\DVVZ}LF0_I~V3NQ6]XW7U5X.png

C:\Users\Administrator\AppData\Roaming\Tencent\Users\825540760\QQ\WinTemp\RichOle\}2X~~E6GPNGYW@CR1X9O4EL.png

C:\Users\Administrator\AppData\Roaming\Tencent\Users\825540760\QQ\WinTemp\RichOle\9J0GYNHHI8YFWA)ZF(EGS$X.png



C:\Users\Administrator\AppData\Roaming\Tencent\Users\825540760\QQ\WinTemp\RichOle\0T%V(KUL]SI}E4Q_L[@J2G2.png

从上面可以看出，大部分电商网站都有很相似的特征,但是如果只是写正则表达式获取，例如获取价钱，会有很多的干扰项，我们采取基于视觉块的分析方法，在浏览器渲染完之后，执行一段js代码，求出各个元素的x，y坐标,元素的宽度,高度，字体大小fz,利用这些视觉的信息，去除这些干扰，例如求价格，价格一般在中间，x坐标大概 300-700之间，y左右肯定小于1000，字体大小肯定是干扰项字体间最大

### 1.7.6 正文抽取，摘要，关键字提取

1.7.6.1 正文抽取算法

结构化正文抽取算法采用开源爬虫框架WebCollector中的正文提取算法ContentExtractor，ContentExtractor使用的算法是CEPR，适用于几乎所有的包含正文的网页，可以抽取结构化新闻，也可以只抽取网页的正文

1.7.6.2 关键字

利用nlang工具包进行分词,选择出现频率最高的若干词语作为关键字

1.7.6.3 摘要

把正文进行断句,选取包含关键字最多的若干句子,作为摘要

## 自动义包装器：(类似造数，和八爪鱼的实现,主要的实现在AutomaticExtractor)

因为自动结构化的很多时候,不是我们想要的数据，或则我们不需要用这么多的字段，所以，我基于javafx 的webview 做了一个自定义包装器的抽取程序，有点类似于，造数和八爪鱼，其实就是把 程序的xpath，class,attr,及 标签名，作为特征保存起来

详情页的结构化:

1 先判断是否有id,有则直接取

1. 判断xpath 抽取出来的是否是我们(结合属性，标签名等)想要的，如果是就抽取出来
2. 如果都不满足前两个条件，则遍历元素，寻找符合 class,attr,标签名都符合条件的程序

列表页：

因为列表页，每一项商品都是符合条件我们自动义包装器的规则，所以会辨别出所有的商品字段的集合，所以我们要吧属于每个商品的字段，聚集在一起

# 2 系统分析与设计



各功能模块的详细说明如下。

1. 结构化测试模块,负责测试指定页面结构化后的数据

## 2.2系统架构



存储模块采用 MySQL+Hbase+Redis+Elasticsearch,

MySql：事务性较强，高性能，用于存储结构化后的页面信息。

Hbase：一个高可靠性、高性能、面向列、可伸缩的分布式存储系统，适用于大量的数据的存储。在网页爬取过程中涉及到大量原始网页数据的下载，将网页数据预先下载存储可以减少在页面结构化处理不符合需求后重复下载时带宽和性能的消耗，这些数据的存储传统数据库性能有限，因而采用Hbase大大提高效率。

Redis：Key-Value数据库，缓存服务，性能高。有丰富的数据结构，如list、hash、String（Key-Value）。采用hash数据结构处理url，用于集群方便高效url去重，url获取。

Elasticsearch：索引数据库，高性能索引搜索。

## 2.3博客爬取流程

URL处理



初步爬取分析入口URL。若是正文网页，结构化抽取正文、标题、时间等，存入数据库。对于非正文网页（首页、分类列表页）提取页面中同网站域名URL，去重后加入待爬取队列（由爬取模块处理），根据策略选择非正文URL继续循环遍提取新URL。

爬取处理模块



当URL处理模块抽取URL存入Reids去重后，爬取模块同步开始对Reids中的正文URL进行结构化正文抽取，结构化后的页面信息存入高速索引数据库Elasticsearch。Hbase用于保存爬取原网页，当结构出错时再次爬取时直接从Hbase中获取网页数据，减少带宽消耗。

结构化正文抽取算法采用开源爬虫框架WebCollector中的正文提取算法ContentExtractor，ContentExtractor使用的算法是CEPR，适用于几乎所有的包含正文的网页，可以抽取结构化新闻，也可以只抽取网页的正文。

正文抽取效果指标 :

比赛数据集CleanEval P=93.79% R=86.02% F=86.72%

常见新闻网站数据集 P=97.87% R=94.26% F=95.33%

算法无视语种，适用于各种语种的网页

电商模块

## 2.4基于相同模板爬取流程



爬取入口URL后，初步分析抓取页面后，若是详情页则开始抓取训练网站评论URL地址。若是非详情页（首页、导航页、搜索列表页等），抽取同网站域名URL、去重后存入库等待爬取模块进一步分析爬取，再根据策略选择从中选取URL继续循环提取新URL。

2.5.1训练url的过程:

我们看到对比以下url,我们可以很容易知道,不同商品列表，只是关键字不同,及页数关键字不同,我们只需要得一个列表url当作模板,及关键字 的键（keyword，q），页数的键（page）即可

步骤如下：

1 检测定位首页的搜索框及提交按钮及导航栏关键字

2 模拟真实用户操作，把导航栏关键字,填进搜索框，点击提交按钮

3 定位下一页按钮,模拟用户点击,去前三页url进行训练（把有规律变化的数值键值,视为页数的键,与搜索关键字相同的键视为关键字的键）

**京东**

<https://search.jd.com/Search?keyword=%E6%89%8B%E6%9C%BA&enc=utf-8&suggest=1.his.0.0&wq=&pvid=6ba425eea1264341bbcf0780adcbffe7>

<https://search.jd.com/Search?keyword=%E6%89%8B%E6%9C%BA&enc=utf-8&qrst=1&rt=1&stop=1&vt=2&suggest=1.his.0.0&cid2=653&cid3=655&page=3&s=57&click=0>

https://search.jd.com/Search?keyword=%E6%89%8B%E6%9C%BA&enc=utf-8&qrst=1&rt=1&stop=1&vt=2&suggest=1.his.0.0&cid2=653&cid3=655&page=5&s=111&click=0

蘑菇街:

<http://list.mogujie.com/s?q=%E4%B8%8A%E8%A1%A3&from=undefinedquerytip0&ptp=1._mf1_1239_15261.0.0.otjAfe&f=baidusem_4uv5iimn1v>

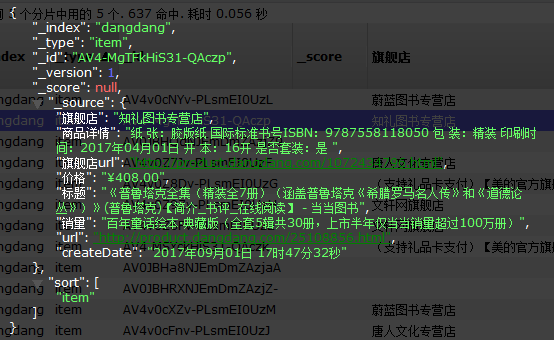
<http://list.mogujie.com/s?page=2&q=%E4%B8%8A%E8%A1%A3&sort=pop&ppath=&ptp=1.eW5XD.0.0.HuMcj&f=baidusem_4uv5iimn1v#category_all>

<http://list.mogujie.com/s?page=3&q=%E4%B8%8A%E8%A1%A3&sort=pop&ppath=&ptp=1.eW5XD.0.0.qGx3Z&f=baidusem_4uv5iimn1v#category_all>

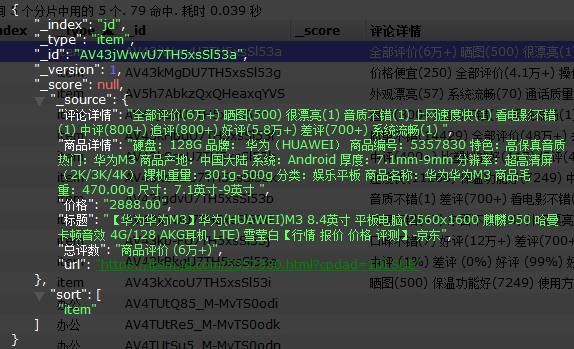
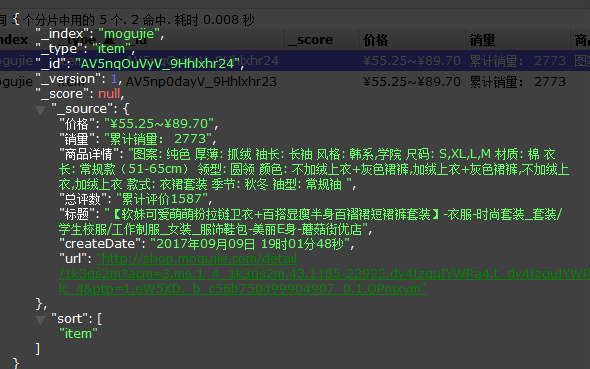
## 3.6结构化数据展示

以下是结构化电商详情页的截图

当当 国美

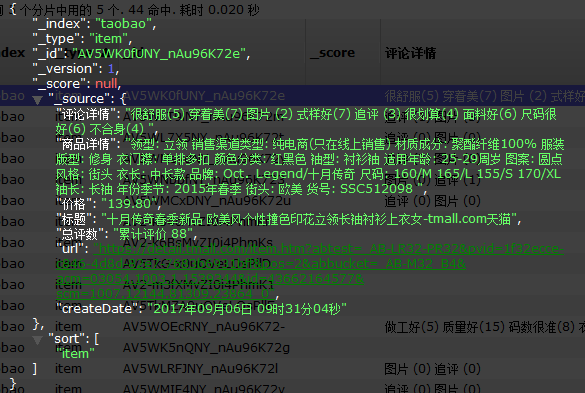
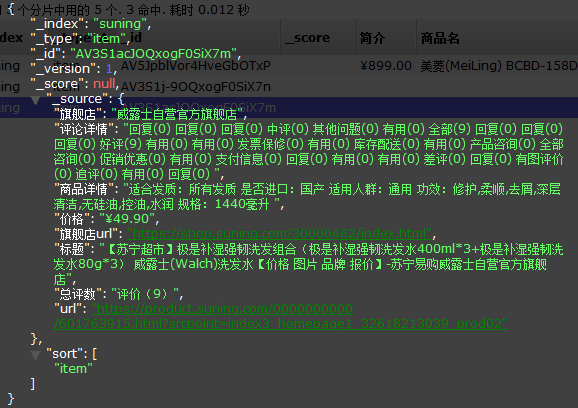
 

京东 蘑菇街

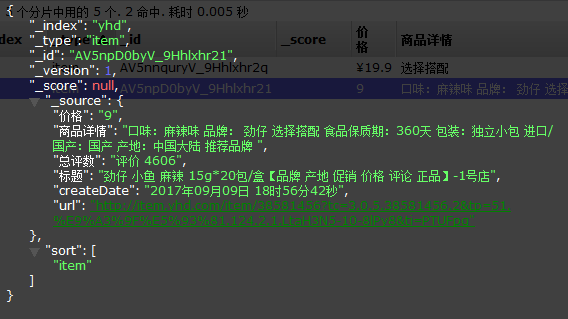
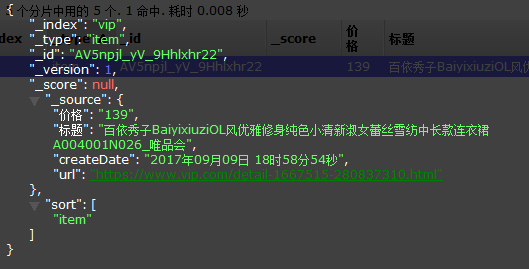
 

苏宁

淘宝



唯品会 一号店



折800

