**中文网页爬虫程序使用手册**

贾婉华 郝培蓉 欧琳琳

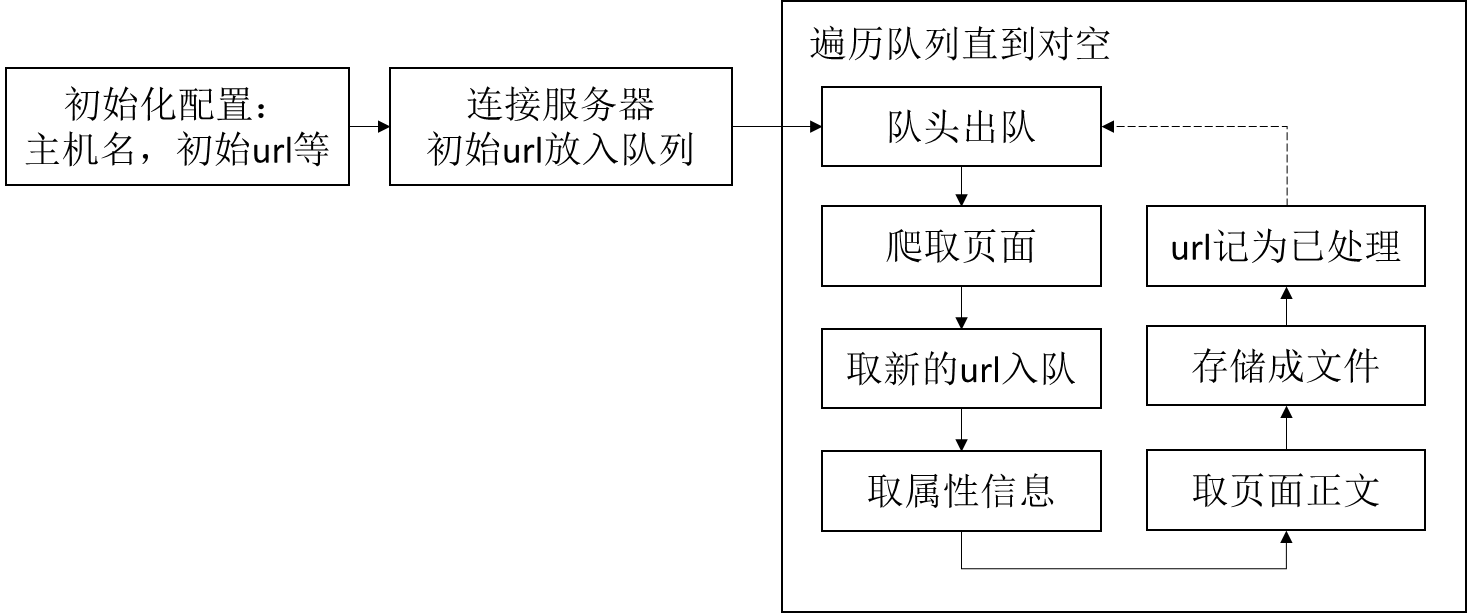
**一、主要功能**

本系统主要完成了从指定的基于http协议的网站，持续爬取网页文本，并去除html标签仅保留正文的功能。

**二、程序工作原理**

首先，用户在配置文件中设置要访问的主机名、初始url和爬取url的限定范围。其后运行本系统，即可开始爬取首个页面。从爬的页面中，提取超链接中的url，经过和已处理的url、待处理的url比对，将全新的url放入待处理队列中。最后从html文本中提取正文，并按照层次存储到文件。遍历待处理的url队列，重复前两个操作，直到队列中没有新的url。

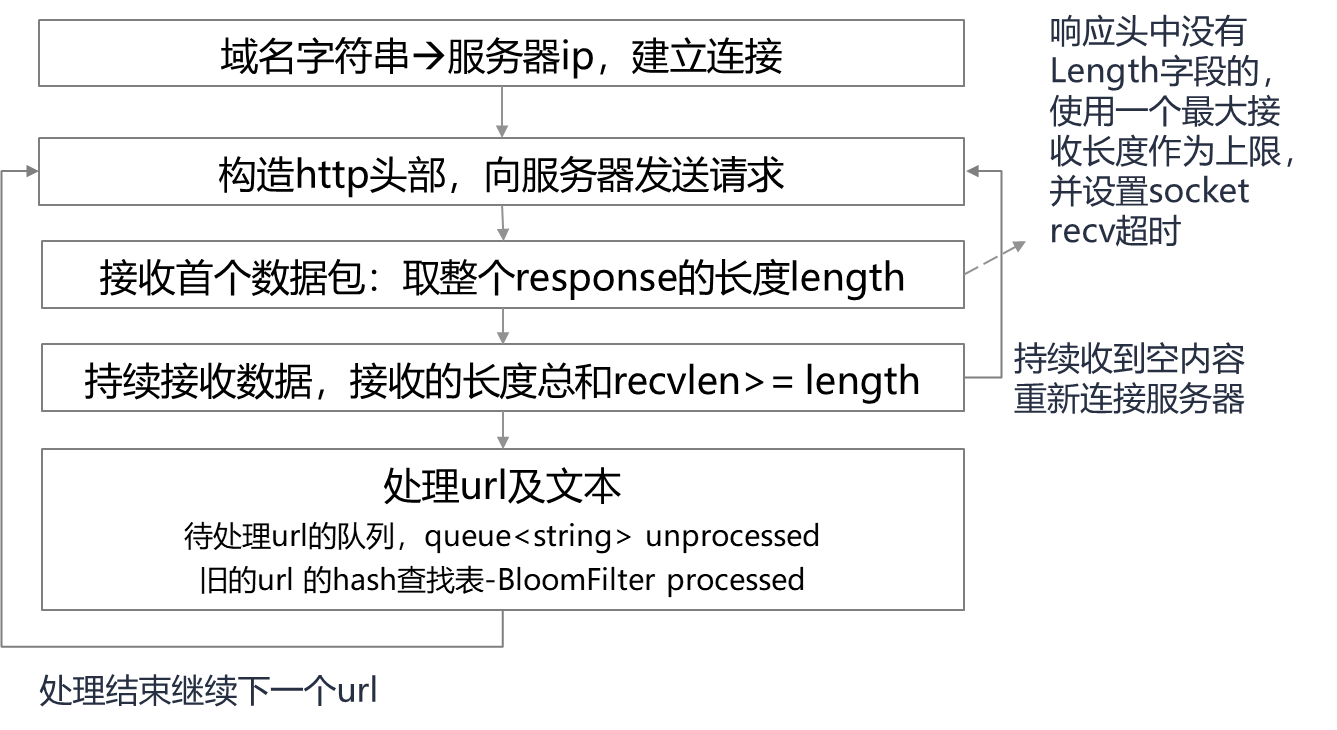
整体流程如下图所示：



**2.1.爬虫**

爬虫部分主要由socket发送http头部和接收响应，找出新的url，再继续访问获取相应来实现。

首先通过配置文件中的hostname域名字符串，转换成服务器ip地址，再和服务器建立TCP连接，连接后根据从队列中取得的url来构造http头部，向服务器发送请求。接收到第一次响应后取整个response的长度，然后持续接收服务器的数据，直到本次接收的长度总和>=从响应的http头中取出的长度。再从接收的文本中找出新url，放入待处理队列中。找新的url这一步骤用到了布隆过滤器，详细解释见2.2节。



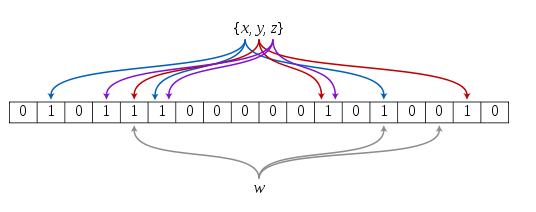
其中需要维持一个待处理url的队列，本系统使用的是两个数据结构，首先是存储url的队列queue<string> unprocessed，还有一个存储已在队列中出现过的url的hash表，即布隆结构BloomFilter processed，用于过滤出新的url。

对于响应头中没有Length字段的网页，使用一个最大接收长度作为上限使socket持续接收，并利用setsockopt函数的SO\_RCVTIMEO选项来设置socket接收超时。以防这个页面已经接收完毕了，但还未达到最大长度，而recv函数一直阻塞等待服务器的数据。设置了recv超时后，一旦接收完毕，超过一定时长没有收到任何来自服务器的信息则继续往下执行。

其中长时间连接服务器可能因为被识别为机器等原因而接收不到响应。为了解决这个问题，对接收不到响应的页面进行计数，超过一定数量则断开与服务器的连接重新连接后再请求数据。

**2.2.布隆过滤器**

布隆过滤器[1]是一个标记某个字符串集合中已有元素，但又不存储元素本身的结构和算法，其优势在于节省了存储空间以及快速检查元素是否存在。算法本身是将一个字符串通过多个Hash映射到多个空间，将对应的空间标记为占用，新的字符串进来时检测对应的空间是否以被占用，已占用则说明其已在集合中不需要再做处理，这些空间若未被占用则标记为已占用，让该字符串进入集合中。



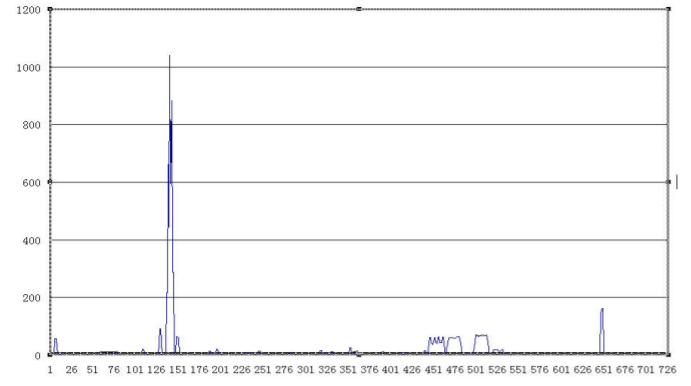
它的缺点是有一定误算率，即有可能某个字符串不在集合中，却被判定为在集合中。另外布隆算法不能删除元素，最好用于查找。

本系统在“标记已处理的url ”这一功能中使用了布隆算法。本系统在检测新的url时就将新的url添加到布隆这一hash表中，而不是处理一个url后放入已处理列表中，这样可以避免重复处理未处理的url。

**2.3.文本分布**

使用基于行块分布函数的通用办法，在线性时间内抽出正文[2]。网页正文区域肯定是文字信息分布最密集的区域之一，根据一定的标准选出文字信息分布集中的区域，即为得到正文。

行块是某一行及其附近几行的集合，其中的有效文字长度称为行块长度。以行块的序号为横坐标，行块长度为纵坐标，得出行块分布函数：

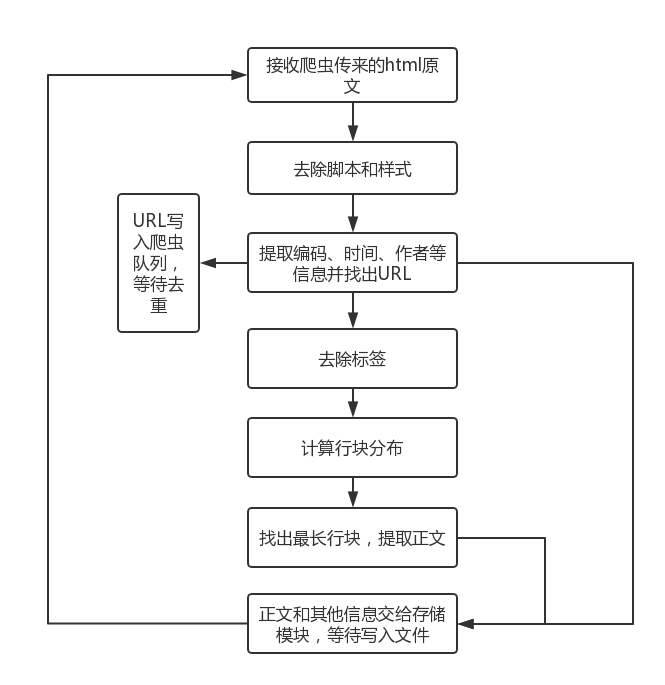


以新浪新闻为例，正确文本区域行号为136到145。由上述行块分布函数图可明显看出，正确的文本区域全都是分布函数图上含有最值且连续的一个区域，这个区域往往含有一个骤升点和一个骤降点。 于是，网页正文抽取问题转化为了求行块分布函数上的骤升骤降两个边界 点，这两个边界点所含的区域包含了当前网页的行块长度最大值并且是连续的。 用数学语言描述就是：

求正文区域所在的起始行块号 Xstart 和终止行块号 Xend（ X 为行号，Y(X)是以 X 为轴的行块长度 ），需要满足以下四个条件(X是行号，Y（X）是行块长度)：

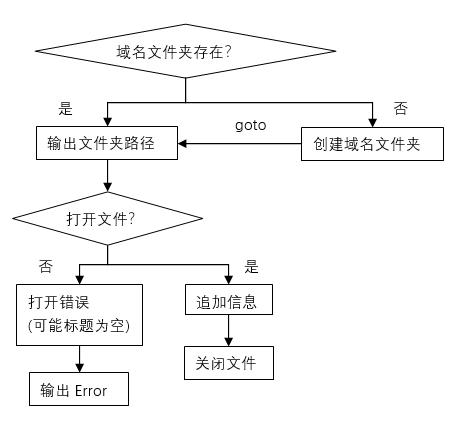
1. Y(Xstart)>Y(Xt)(Y(Xt)是第一个骤升点，必须超过某一阈值)
2. Y（Xn）不等于0（start+1<n<start+K,K为行块厚度，紧随骤升点的行块长度不能为0，避免噪声）
3. Y（Xm）=0（end<m<end+1，骤降点及其尾随行块长度为0，保证正文结束）
4. 对于某个X，当取到max（Y（X））时，Xstart<X<Xend（保证此区域是取到行块值最大的区域）

程序工作流程如下：



**2.4.存储**

将获取到的网页信息存储于本地，首先创建域名文件夹，将爬取的标题作为新文件的文件名，之后将信息输出到txt文本中。



**三、使用方式**

**3.1.输入**

输入初始URL、域名、爬取网页范围（host.conf）

**3.2.输出**

爬取到的每个有效网页的正文与其他信息的文件（pages/hostname/filename）

**四、技术规范**

**4.1代码规范**

程序用C++实现

函数命名：驼峰式命名

变量命名：下划线全小写命名

类的声明：头文件

成员函数定义：cpp文件

**4.2注释规范**

要求：

/\*=================================

\*函数名称

\*功能简介

\*参数列表和解释

\*返回值类型

\*=================================\*/

示例：

/\*=================================

\*Spider类

\*init:初始化：设置所需的url hash表，主机名和初始url

\*params:unsigned int size：hash表大小

\* const char \*hn:主机名

\* string url:初始url

\*return:void

\*=================================\*/

**4.3编译规范**

使用的操作系统和gcc版本如下：

|  |  |
| --- | --- |
| 操作系统 | GCC/G++ Version |
| Ubuntu16 | 5.3.2 |
| Ubuntu17 | 6.3.0 |
| centos7 | 4.9.2 |
| centos7 | 6.0.22 |

使用makefile编译，makefile内容如下：

all: Spider splitString.o main TextFilter.o

@echo "making spider"

g++ main.o Spider.o splitString.o BloomFilter.o Saver.o TextFilter.o -o spider -std=c++11

Spider:Spider.cpp Spider.h BloomFilter.o TextFilter.o Saver.o

g++ -c Spider.cpp -std=c++11

main:main.cpp

g++ -c main.cpp -std=c++11

#spider:spider.cpp Spider

# g++ -c spider.cpp -std=c++11

splitString.o:splitString.cpp

g++ -c splitString.cpp -std=c++11

BloomFilter.o:BloomFilter.cpp BloomFilter.h

g++ -c BloomFilter.cpp -std=c++11

TextFilter.o:TextFilter.cpp TextFilter.h

g++ -c TextFilter.cpp -std=c++11

Saver.o:Saver.cpp Saver.h

g++ -c Saver.cpp -std=c++11

clear:

rm \*.o spider

**4.4库**

|  |  |
| --- | --- |
| fstream | netdb.h |
| sys/types.h | arpa/inet.h |
| iostream | sys/socket.h |
| cstring | sys/stat.h |
| unistd.h | cerrno |
| vector | cstdio |
| string | cstring |
| cstdlib | queue |
| regex | map |

1. **实验分工**

各部分程序内容在本文第二部分已经阐述，在此仅说明实验各部分分工。

|  |  |
| --- | --- |
| 姓名 | 分工 |
| 贾婉华 | 网页爬取、布隆过滤器 |
| 郝培蓉 | 文本存储 |
| 欧琳琳 | 网页过滤与去噪 |

1. **参考文献**
2. 李玉龙.布隆过滤器(Bloom Filter)详解[EB/OL].https://www.cnblogs.com/liyulong1982/p/6013002.html,2016-10-30/2018.1.1
3. 陈鑫.基于行块分布函数的通用网页正文抽取[EB/ OL].http://cx-extractor.googlecode.com/svn/trunk@6,2013.5.22/2018.1.17