Journal of Computer Applications
计算机应用,2013,33(2):554-557
ISSN 1001-9081
CODEN JYIIDU
2013-02-01
http://www.joca.cn

文章编号:1001-9081(2013)02-0554-04
doi:10.3724/SP.J.1087.2013.00554

# 基于网页正文结构和特征串的相似网页去重算法

熊忠阳,牙　漫*,张玉芳
(重庆大学 计算机学院,重庆 400044)
(*通信作者电子邮箱 yamanv@163.com)

**摘　要**:为了减少重复网页对用户的干扰,提高去重效率,提出一种新的大规模网页去重算法。首先利用预定义网页标签值建立网页正文结构树,实现了层次计算指纹相似度;其次,提取网页中高频标点字符所在句子中的首尾汉字作为特征码;最后,利用 Bloom Filter 算法对获取的特征指纹进行网页相似度判别。实验表明,该算法将召回率提高到了 90% 以上,时间复杂度降低到了 $O(n)$。

**关键词**:网页去重;网页标签值;高频标点;特征码;网页指纹相似度
**中图分类号**:TP391.1;TP393.092　　**文献标志码**:A

# Detection and elimination of similar Web pages based on text structure and string of feature code

XIONG Zhongyang, YA Man*, ZHANG Yufang
(*College of Computer Science, Chongqing University, Chongqing* 400044, *China*)

**Abstract:** In order to reduce the interference of the duplicated Web pages, and improve the efficiency of detection and elimination of similar Web pages, a new kind of large-scale Web page detection algorithm was proposed. Firstly, adopting the Web label values, the algorithm created the text structure trees to realize the fingerprint similarity calculation layer by layer. Secondly, the head and tail words of a certain sentence, in which high frequency punctuations occur, were extracted out as the feature code. Lastly, the fingerprint similarity of Web page features was discriminated with Bloom filter algorithm. The experimental results show that the algorithm can improve the recall rate up to more than 90%, and reduce the time complexity to $O(n)$.

**Key words:** detection and elimination of similar Web pages; Web label value; high frequency punctuation; feature code; fingerprint similarity of Web page



## 0　引言

随着互联网的高速发展,越来越多的机构、组织通过网络发布信息。一些网站为了获取相应利益对热点话题高频率转载造成了网络信息的冗余。这些网页一部分是一字不差的完全重复的网页,一部分是进行了修改的网页。文献[1]通过“用户满意度测评理论模型”发现,百度和 Google 中国都存在比较严重的重复网页问题;作者也通过实验发现在这两个搜索引擎首页出现的检索结果中,重复率高达 25% 左右,用户满意度都较低。据研究,在一个大型信息采集系统中,30% 的网页是和另外 70% 的网页是完全重复或者近似重复的[2]。通常,重复的网页内容不能给人们带来太多的新信息,不仅增加了人们筛选、阅读的负担,降低了检索效率,也浪费了大量的空间资源。由于管理漏洞并缺乏行业规范,仅单纯靠增加人力检测转载网页显然得不偿失。目前只能从技术上解决网页重复问题。如何快速准确地发现这些内容上相似的网页已经成为提高搜索引擎服务质量的关键技术之一。

## 1　相关工作

网页查重具体可以分为以下几个步骤:

1)提取网页正文(经过去噪等的预处理提取出的文档)的特征;

2)用提取出来的网页特征与已有的网页特征进行比较;

3)将2)中比较的结果与设置的阈值进行对比,判断其重复与否。

网页特征的提取及大规模网页特征比较是网页去重的关键。目前国内外常用的算法有以下几种:

1)基于特征串的去重算法,例如 SCAM(Stanford Copy Analysis Mechanism)算法[3]、DSC(Digital Syntactic Clustering)[4-5]及其改进的 DSC-SS(DSC's Supper Shingle)[7]算法、I-Match[8]算法。这种去重算法有较高的准确率,但是空间复杂度很高不适合大规模网页去重;并且 DSC-SS 在处理短小文档时准确率很低。

2)基于聚类统计的去重方法,例如近似镜像网页检测算法[8],以国家标准的汉字编码字符集[9]作为向量的基,统计文本中每个汉字的字频作为特征向量,通过TF-IDF算法计算向量的夹角判断相似度。聚类时间复杂度 $O(n^2)$,对大规模网页去重不实用;由于每个汉字表示的内容重要程度不一样,并且此方法忽略了文本结构信息,导致一些主题相关但内容并不重复的网页误判。

3)基于词频[10]的去重算法,如白广慧[11]提出了基于文档中高频词序列的网页去重算法,此法对网页文本进行分词及词性标注等预处理,以便去掉虚词及提取关键词。此算法劣势在于依赖语料库,导致预处理时间过长,而且由于中文语言的复杂性造成关键词标注不精确。

**收稿日期**:2012-08-20;**修回日期**:2012-10-07。

**作者简介**:熊忠阳(1962-),男,重庆人,教授,博士,主要研究方向:数据挖掘、并行处理;　牙漫(1986-),女,河北保定人,硕士研究生,主要研究方向:数据挖掘、搜索引擎;　张玉芳(1965-),女,上海人,教授,主要研究方向:数据挖掘。

4)基于标点符号的去重算法,如吴平博等[12]提取每个句子中首尾的字符作为特征串,引入模糊匹配思想对特征进行判别。然而这种算法基于特征串的两两比较,虽然对完全重复网页有很精确的效果,但对近似重复网页效果较差,且时间复杂度很高,不适合大规模网页去重。

文献[13]中提出了基于正文结构和长句提取的网页去重算法,该方法利用网页文本特征提取出网页正文结构树,然后提取出每个段落的最长的 $m(m>0$ 且不大于文章句子总数)个句子,再利用 MD5 算法取其摘要得到网页的指纹,最后根据网页指纹的相似度来确定网页是否重复。由于汉语言的复杂性,导致此种方法对于段落或者句子中有删字、添字、句子断句等调整时算法的召回率不高;并且指纹进行相似性比较时空间复杂度和时间复杂度都很高。考虑到 Web 上的大部分文档大小一般只有 4 KB[14],文献[13]中提取长句的算法并不实用。

本文根据网页特征提出了基于网页正文结构和特征串的去重算法。算法首先挖掘网页文本中存在的结构,提取网页正文生成网页正文结构树,并从段落中含有高频标点符号的句子首尾各提取一个字符或者汉字,构成标签文本块;然后利用 Bloom Filter 算法计算得到其指纹,并进行网页指纹相似度的比较。

## 2　算法思想

由于自然语言的复杂性,文献[13]对完全重复网页的查重有较高的准确率和召回率,但在近似网页去重方面准确率较低。微软的研究人员耗时十一周从网络中下载了 $15\times10^7$ 个网页,发现其中有 29.2% 的重复网页,在这些重复的网页中,仅存在 22.2% 的完全重复网页(正文内容一字不差)。这个实验说明近似重复网页是影响检索结果的最大障碍。

网页查重的目的是为了方便用户,减少用户在重复网页筛选上的时间浪费。如果网页去重阶段本身浪费了过多时间,对用户来说得不偿失。因此,需要提高查重效率,降低查重的时间复杂度。本文利用自然语言中的标点符号标志,提取出含高频标点符号句子首末各一个汉字,每个段落提取出来的特征串组成了该段落的标签文本块,并利用 Bloom Filter 算法获取标签文本块的指纹来判断网页相似情况。Bloom Filter 算法能效降低算法的时间复杂度。

综上所述,本文提出了基于网页正文结构和特征串的去重算法。算法的基本思想是:首先利用网页的结构信息去掉网页中的噪声,然后利用正文结构树生成算法将网页正文生成一棵正文结构树,接着从结构树中层次地提取每层的特征串并获取其指纹,最后根据相似度计算方法判断网页相似与否。

## 3　特征串及网页正文结构树

### 3.1　网页特征串

由于标点符号将文章自然分隔成句子,本文即利用标点符号的分隔特点,提取这些独立句子的首尾字符或者汉字作为文章的特征码,首尾特征码构成特征串。通常情况下将逗号、句号、分号等高频字符作为分隔标记。

**定义 1**　标签文本块。表示成元组的形式 $TAGSTR=(STR,TAG,NUM)$,$STR$ 表示标签对中的特征串,$TAG$ 表示标签号,$NUM$ 表示特征串的个数。

在正文结构树中,将段落标签对(不含标签)之间的正文视作一个段落,整个段落的特征串构成一个标签文本块,并且记录标签文本块中特征串的个数。

### 3.2　网页正文生成树

网页中每个域或者元素都伴随着一个开始标记(如 <h1>)和一个可选的结束标记(如 </h1>),元素也有相应的属性及属性值。在本文中将这些标签被看作正文结构中段落重要性程度的标志,是建树的依据。

将网页文本中的标签根据标识作用及重要程度进行赋值,从高到低依次为小标题编号、HTML 标签和短段落标签。例如:一篇网页正文既使用了小标题编号又使用了 HTML 标签,则使用小标题编号进行赋值。在 HTML 标签这个类中预先设定优先级,从高到低依次为强制类标签、字体大小标签、字体类型标签、字体颜色标签及其他。这样就保证了网页正文结构表示形式的唯一性。相同的标签权值在网页文本结构树的同一个层次。

网页正文生成树算法是根据网页标签及其属性值对网页正文进行层次划分,即根据各个层次权值判断新段落是其兄弟节点还是孩子节点,将其表示成一棵树的形式。

**定义 2**　结构树的节点[13]。表示成二元组 $TNode=(PID,PW)$,其中 $PID$ 为自然段的编号,$PID=\{PID_0,PID_1,\cdots,PID_n\}$,$PID_0$ 为网页文本的标题编号;$PW=\{PW_0,PW_1,\cdots,PW_n\}$ 为自然段对应的权值。

可按如下规则转换为一棵正文结构树[13]:

1)若 $T$ 为空,$StrucTree$ 为空。

2)若 $T$ 非空,则令 $PID_0$ 为正文结构树的根节点,$PID_1$ 为结构树的第一个孩子节点。后面的自然段编号按照其权值大小进行插入操作。

插入操作遵行以下规则:

规则 1　如果 $PW_{i+1}=PW_i$,则 $PID_{i+1}$ 作为 $PID_i$ 右兄弟节点。

规则 2　如果 $PW_{i+1}<PW_i$,则 $PID_{i+1}$ 作为 $PID_i$ 的孩子节点。

规则 3　如果 $PW_{i+1}>PW_i$,则继续与 $PID_i$ 的父节点比较。

3)使用上述三个规则对段落编号依次进行操作,直到找到合适的位置将每个段落编号都插入到正文结构树中。

**定义 3**　网页正文结构树[13]。表示成二元组的形式 $CatlaTree=(Root,F)$。其中:$Root$ 是根节点;$F$ 是 $m(m\geqslant0)$ 棵目录结构子树的森林,$F=\{CT_1,CT_2,\cdots,CT_m\}$,其中 $CT_i=(R,i,F_i)$ 称作根 $Root$ 的第 $i$ 棵子树。当 $m\neq0$ 时,在树根与其子树森林之间存在下列关系:

$$RF=\{\langle Root,R_i\rangle\mid i=1,2,\cdots,m\}$$

## 4　Bloom Filter 算法

Bloom Filter 算法[15]能够在允许一定误差的情况下同时节省时间和空间开销,有较低的时间空间复杂度,因而适合海量网页去重。该算法的基本思想是:

1)设定数据集合 $A=\{a_1,a_2,\cdots,a_n\}$,其中 $a_i$ 为提取的特征串。

2)设定一个 $\boldsymbol{V}$ 向量,$\boldsymbol{V}=\{v_1,v_2,\cdots,v_m\}$,其中 $v_i=0$ 或者 1,即设定一个 $m$ 维的比特数组。

3)设定 Hash 函数组 $H=\{h_1,h_2,\cdots,h_k\}$,$h_i$ 能均匀散列,且 $h_i[a_j]$ 为 0 ~ $m$ 的整数。

4)将 $A$ 中的元素利用 $H$ 中 $K$ 个 Hash 函数进行散列得到的数字作为 $\boldsymbol{V}$ 的下标,并令其中元素 $V[h_1(a_i)]=1$,$V[h_j(a_i)]=1,\cdots,V[h_k(a_i)]=1$。

5）在判断新的元素 $y$ 是否在 $A$ 中时，利用 Hash 函数组计算 $\{h_1(y),h_2(y),\cdots,h_k(y)\}$，再观察 $V[h_j(y)]$ 是否为 1 即可。若全部为 1，则说明 $A$ 中包含 $y$ 元素，否则不包含。由于 Hash 函数的特点，Bloom Filter 算法会存在误差。例如：已知 $h_1(x)=1,h_2(x)=1,h_1(y)=3,h_2(y)=3,h_1(z)=1,h_2(z)=3$，显然，$z\neq x$，且 $z\neq y$。但是，若利用此算法判断，原数组中 $V[1]=1=V[h_1(z)],V[3]=1=h_2(z)]$，导致误判。不过，选择合适的 Hash 函数及数组 $V$ 的维数能最大限度减小错误率。

## 5 基于正文结构和特征串网页去重算法

为了能更好地提取网页特征，首先要进行网页正文提取，依照网页的结构信息去掉网页中的噪声。例如过滤掉网页中的导航条、广告条、超链接以及网页下方的网站说明等信息。提取得到的网页文本质量越高，那么对其进行去重检测越精确。预处理后保留网页的结构、标点等模板信息。

本文中首先利用网页正文生成树算法得到一棵网页正文结构树，然后利用 Bloom Filter 算法计算每一层次特征串的指纹，最后进行相似性的检测（如图 1 所示）。基于 Bloom Filter 的处理方式不仅节省空间，而且该方法使大规模的数据映射可以读入内存，同时减少和外存的交互，从而提高了检索的效率。

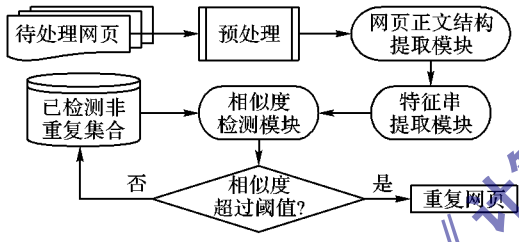


图 1 网页去重整体流程

PageRank 算法根据网站的外部链接和内部链接的数量和质量来衡量网站的价值，是 Google 衡量网站等级、重要性的基本方法。

$$PR(i)=C1\sum_{j\in B(i)}\frac{PR(j)}{N(j)}+C2\times E(i)$$

其中：$PR(i)$ 指节点 $i$ 的 PageRank 评分，$B(i)$ 为指向 $i$ 的节点集合，$N(j)$ 表示 $j$ 的外连接个数，$E(i)$ 表示用户随机打开 $i$ 的概率，$C1$、$C2$ 为给定常数。

本文利用含有 PageRank 算法的网页爬虫获取相关网页，当然，在重复网页中，排名在前的网页作为源发网页。

本文算法伪代码如下：

```
计算已存网页的特征向量 V,标签文本块 LT
for 各个层次的标签文本块 LT_i do
  for 标签文本块中特征串 TC_j do
    利用 BloomFilter 算法判断 TC_j 是否包含
    在已检测网页的 LT 中
    if TC_j 包含在 LT 中 then
      记录重复的特征串数目
    endif
  endfor
  计算同层次特征串的相似度 similarity
  if similarity > β then                    //β 为阈值
    此层次内容相似,记录相似层个数
  endif
endfor
计算相同层次所占层次总数比例 similarity'
if similarity' > β then                     //β' 为阈值
  此为重复网页
else 保留此非重复网页
endif
```

如果此网页不重复，则继续与其他已存源发网页进行比较，直至所有源发网页都比较完；若非重复，则保存，添加进入源发网页库。

令同层次检测到的重复特征串个数为 $c_{strm}$，同层次源发网页特征串数目为 $o_{strm}$，待检测网页特征串数目为 $w_{strm}$，则：$similarity=c_{strm}\times 2/(o_{strm}+w_{strm})$。

令相同层次数目为 $s_{level}$，源发网页总层次数目为 $o_{level}$，待检测网页总层次数目为 $w_{level}$，则 $similarity'=s_{level}\times 2/(o_{level}+w_{level})$。

$\beta$、$\beta'$ 均为阈值，其取值为经验值。

## 6 实验结果和分析

实验利用 Java 语言开发；开发环境为 Windows 7 + IDE（Eelipse3.2）+ JDKI.5 + perlZ.s.s；数据库为 SQL Server 2005。

实验中，根据大量的实验获得经验值，令 $\beta=0.92$，$\beta'=0.90$ 可以取得最佳效果。

### 6.1 召回率、准确率的比较结果

令 $A$ 表示实际重复文档集合，$B$ 表示利用算法处理后得到的重复文档集合，则：

准确率 $=\frac{|A\cap B|}{|B|}\times 100\%$，即重复网页的比率；

召回率 $=\frac{|A\cap B|}{|A|}\times 100\%$，即重复网页被检测出的比率。

实验一 实验网页来自于各大门户网站上收集的 5 000 篇新闻网页。

**表 1 新闻网页去重结果对比**

| 去重方法 | 准确率/% | 召回率/% |
|---|---|---|
| 基于正文结构和长句提取的网页去重算法 | 94.2 | 88.3 |
| 本文算法 | 95.3 | 90.4 |
| 基于特征串的大规模去重算法 | 98.1 | 80.4 |

实验二 考虑到网页文章通常都小于 4 KB，从各大网站上搜集文档不大于 4 KB 的网页 1 000 篇。

**表 2 小规模文档去重结果对比**

| 去重方法 | 准确率/% | 召回率/% |
|---|---|---|
| 基于正文结构和长句提取的网页去重算法 | 89.5 | 70.2 |
| 本文算法 | 95.6 | 91.7 |
| 基于特征串的大规模去重算法 | 99.6 | 82.3 |

基于正文结构和长句提取的网页去重算法[13]中的 MD5 算法是单项散列算法（HASH 算法）的一种，是将任意长度的信息压缩到某一定程度的函数。压缩之后生成的信息成为消息摘要，并且压缩过程不可逆。MD5 可以为任何文件（不管其大小、格式、数量）产生一个同样独一无二的“数字指纹”，并不具有语义检测功能，如果对文件做了任何改动，其 MD5 值也就是对应的“数字指纹”都会发生变化。例如：

MD5（MD5 算法是一种单项加密算法）=
71DD3618FBA1388ABC37510C8773C5FB

MD5（MD5 算法是种单项加密算法）=
ED489033C81DAEE44EEA7C11B976C59A

从该例可以看出，使用 MD5 值作为特征串指纹的粒度很

大,使得指纹表征能力不强,容易受到添字、减字等变化的影响,所以这种散列方法很容易漏检部分修改的近似重复网页。

由于基于正文结构和长句提取的网页去重算法中汉语言的复杂性及 MD5 算法规则的严格要求,使得即使语义完全相同仅稍微改变的句子经过算法计算,其指纹结果可能不同,导致很多近似重复网页漏检,最终致使召回率偏低。而基于特征串的大规模去重算法[12]保证了文章特征串的结构性和连续性,它把连续相同的特征串所占比例作为判别文章相似度的标准,致使仅能对完全相似的文章有较好查重效果,但在部分相似文章查重方面召回率偏低。

根据对网页转载规律的统计:首先,转载文章一般不会在句子首尾增加、删除或者更改字词,所以本文算法不易受到转载修改的影响;其次,文章中最长的句子不一定就是中心句,所以基于正文结构和长句提取的网页去重算法所提取的特征串并不具有精准的代表性,易将文章内容相近,但表达中心不同的文章错归为重复网页,导致低准确率。这在实验二的小规模文档去重中尤其明显。本文算法所提取的特征串为文章中高频标点所在句子的首尾汉字或者字符,能有效避免同特征串不同义的文章的错检,提高了文章的准确率;但是由于 Bloom Filter 算法自身的缺陷,可能导致准确率提高幅度不太明显。

### 6.2　时间复杂度分析

以“新型艾滋病”、“酒驾”等 10 个话题为例,每个话题分别抓取 100、500、…、15 000 篇主题相关网页,利用本文算法进行网页去重,并统计了相同数据量下不同话题进行网页去重的平均时间消耗情况如图 2 所示。

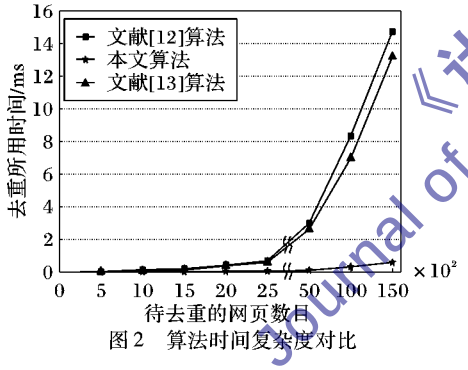


图 2　算法时间复杂度对比

从图 2 中可以看出,基于特征串的大规模去重算法和基于正文结构和长句提取的网页去重算法去重消耗的时间呈指数级增长,而本文算法的时间复杂度呈线性变化。

对基于特征串的大规模去重算法,假设文章中提取的特征串至少为 $m$ 个,则共计 $m*n$ 个特征串($n$ 为网页数目),此算法要求对提取的特征串与源发网页特征串两两比较,根据连续相同的特征串所占的比例作为网页相似性的衡量标准。显然,该算法的时间复杂度为 $O(n^2)$,不适合对大规模网页去重。

对基于正文结构和长句提取的网页去重算法,提取特征句共计 $m*n(m>0$ 且不大于文章句子总数$)$ 个,虽然明显少于本文算法和基于特征串的大规模去重算法提取的特征串的个数,但是特征串的比较还是基于待查询网页与源发网页的两两比较,时间复杂度也是 $O(n^2)$。

本文算法中查询一个元素是否在集合中,只需用 $K$ 个 Hash 函数进行计算,然后利用这 $K$ 个 Hash 输出的值作为下标,判断这些下标对应的比特数组是否为 1 即可。$K$ 个 Hash 函数计算几乎是常数的时间,当然,利用查找 Hash 值下标位置的比特数组也只需要极少时间。所以本算法的时间复杂度为 $O(n)$,查重时间不会随着网页规模的增加而迅速增大,适合海量网页的去重。

综上所述,本文算法在对检索速率要求较高,而对去重正确率、召回率方面有一定放松的情况下更具有优势。

## 7　结语

本文充分考虑了中文语言的复杂性以及小规模文档的特点,为了避免对语义相同但语句部分调整的近似重复网页的漏检,提出了基于网页正文结构和句子首尾特征串的大规模去重算法,提高了去重的召回率和准确率,且对小规模文档去重达到了良好的效果。本文利用了 Bloom Filter 算法,将正文结构树中的特征串进行散列处理,降低了算法的时间复杂度,适用于大规模网页去重。

**参考文献:**

[1] 毛晓燕. 搜索引擎用户满意度研究的实证分析——以百度和 Google 中国为例[J]. 图书馆杂志,2008, 27(3): 40 - 47.

[2] CROFT W B, METZLER D, STROHMAN T. 搜索引擎——信息检索实践[M]. 刘挺, 秦兵, 张宇, 等译. 北京: 机械工业出版社, 2010.

[3] SHIVAKUMAR N, GARCIA-MONLINA H. SCAM: a copy detection mechanism for digital documents [C]// Proceedings of the Second Annual Conference on the Theory and Practice of Digital Libraries. Austin: Texas A & M University, 1995: 201 - 210.

[4] BRODER A Z, GLASSMAN S C, MANASSE M S, *et al*. Syntactic clustering of the Web [C]// The 6th International Conference on World Wide Web. Essex: Elsevier Science Publishers, 1997: 1157 - 1166.

[5] CONRAD J G, GUO X S, SCHRIBER C P. Online duplicate document detection: signature reliability in a dynamic retrieval environment [C]// CIKM '03: Proceedings of the 12th International Conference on Information and Knowledge Management. New York: ACM, 2003: 443 - 452.

[6] CHOWDHURY A, FRIEDER O, GROSSMAN F D, *et al*. Collection statistics for fast duplicate document detection [J]. ACM Transactions on Information Systems, 2002, 20(2): 171 - 191.

[7] KOŁCZ A, CHOWDHURY A. Lexicon randomization for near-duplicate detection with I-Match [J]. The Journal of Supercomputing, 2008, 45(3): 255 - 276.

[8] 王勇建, 谢正茂, 雷鸣, 等. 近似镜像网页检测算法的研究与评价[J]. 电子学报, 2000, 28(Z1): 130 - 132.

[9] 李建超.《信息交换用汉字编码字符集·基本集》(GB2312—80)二级汉字理据性研究[D]. 济南: 山东师范大学, 2010.

[10] LI W, LIU J Y, WANG C. Web document duplicate removal algorithm based on keyword sequences [C]// IEEE NLP-KE '05: Proceedings of 2005 IEEE International Conference on Natural Language Processing and Knowledge Engineering. Piscataway: IEEE, 2005: 511 - 516.

[11] 白广慧. 网页排重技术研究及应用[D]. 北京: 中国科学院计算技术研究所, 2006.

[12] 吴平博, 陈群秀, 马亮. 基于特征串的大规模中文网页快速去重算法研究[J]. 中文信息学报, 2003, 17(2): 28 - 35.

[13] 黄仁, 冯胜, 杨吉云, 等. 基于正文结构和长句提取的网页去重算法[J]. 计算机应用研究, 2010, 27(7): 2489 - 2491.

[14] 连浩. 基于自然语言处理的网页去重关键技术研究[D]. 北京: 北京邮电大学, 2006.

[15] WANG X J, SHEN H. Improved decaying bloom filter for duplicate detection in data streams over sliding windows [C]// ICCSIT 2010: The 3rd IEEE International Conference on Computer Science and Information Technology. Piscataway: IEEE, 2010, 4: 348 - 353.