中文短文本去重方法研究*

高翔[1]，李兵[2]

（1.北京大学汇丰商学院，广东省 深圳市 518055；2. 对外经济贸易大学，北京市 朝阳区 100029）

摘要：本文针对中文短文本冗余问题，提出了有效的去重算法框架。考虑到短文本海量性和简短性的特点，以及中文与英文之间的区别，引入了 Bloom Filter、Trie 树以及 SimHash 算法。算法框架的第一阶段由 Bloom Filter 或 Trie 树进行完全去重，第二阶段由 SimHash 算法进行相似去重。本文设计了该算法框架的各项参数，并通过仿真实验证实了该算法框架的可行性及合理性。

关键词：文本去重；中文短文本；Bloom Filter；Trie 树；SimHash 算法

Research on a method to detect reduplicative Chinese short texts

Xiang Gao[1], Bing Li[2]

(1.Peking University HSBC Business School ,Shenzhen, Guangdong Province 518055, China ; 2. University of International Business and Economics , Beijing 100029 ,China)

Abstract: The article presents an effective algorithm framework for text de-duplication, focusing on redundancy problem of Chinese short texts. In view of the brevity and huge volumes of short texts, we have introduced Bloom Filter, Trie tree and the SimHash algorithm. In the first stage of the algorithm framework, Bloom Filter or Trie tree is designed to remove duplications completely; in the second stage, we use the SimHash algorithm to detect similar duplications. This text has designed the parameters used in the algorithm framework, and we testified the feasibility and rationality of it.

Key words: text de-duplication;Chinese short texts;Bloom Filter;Trie tree;SimHash algorithm

1 引言

近年来，随着我国计算机科学技术的迅猛发展，微博客、BBS、即时通讯工具等通过中文短文本形式承载信息的各项传播技术日益普及。短文本信息的迅猛增长，在为信息决策带来丰富资料来源的同时，也产生了大量冗余、无效的重复信息。庞大的重复信息集，不仅大量占用了系统的存储空间，同时也不利于针对短文本信息进行有效的数据挖掘，对于信息决策的准确性与及时性都会造成影响。因而迫切需要有效的中文短文本去重方法应用于企业与研究实践。

对于文本去重技术，我们根据算法原理的不同将其分为两类[1]~[3]，一类采用基于字符串的比较方法（基于语法的方法）。1994 年，sif 系统[4]的提出，使得在大规模文件系统中寻找内容相似的文件成为可能。虽然并未涉及文本去重的相关技术，但是其率先提出的“信息近似指纹”思想，已经被之后的很多文本去重系统所应用（Heintze 的 KOALA 系统[5]与 Border 等人[6]提出的“Shingling”便一脉相承了 sif 的思想原理）。1995 年，Brin 与 Garcia-Molina 首次提出了文本复制检测系统——COPS[7]，为以后的检测系统搭建了基本框架。2000 年，采用后缀树搜寻字符串间最大子串的 MDR 原型由 Monostori 等人[8]建立，随后又针对存储部分进行了改进[9]。而 YAP3[10]与 MDR 类似，也是通过字符串匹配算法直接在文档中搜寻最大的匹配字符串。2002 年，Chowdhury 等人采用与 sif 相同的技术原理，开发出了 I-Match 系统[11]。I-Match 假设的前提是

*基金项目：教育部人文社会科学项目（No. 11YJA870017）

作者简介：高翔，男，硕士研究生在读；李兵，男，博士，副教授

不考虑高频词与低频词对语义的影响，忽略了语义分析导致 I-Match 算法的精度不是很高。针对其精度不高的问题，文献[12]提出了可以利用随机化的方法来解决。

另一类是基于词频统计的方法（基于语义的方法）。1995 年，Shivakumar 与 Garcia-Molina 建立了基于向量空间模型的 SCAM[13]~[14]，随后又提出了 dSCAM[15]。1997 年，针对主流的基于语句比较的检测方法，Antonio 等人另辟蹊径，设计了 CHECK 原型[16]。CHECK 将文档分解为树型结构, 再利用向量点积法来比较文档相似度，有效地解决了检测速度慢以及检测率低的问题。2003 年，Jung-Sheng Yang 与 Chih-Hung Wu[17]提出用图结构来存储测试文本，并比较相似性的方法。之后的 SimHash[18]与 Spotsig[19]算法重视了语义的层面，对特征值进行了有效的选取。

由此可见，目前文本去重技术已有了长远的发展。但就目前而言，中文文本去重技术，主要是借鉴已有的英文文本去重方法，应用于中文网页的查重[20]。针对目前日益增多却尚未有效利用的中文短文本信息，就目前查阅资料的情况来看，国内在此领域的去重技术的研究还并不完善。由于短文本具有海量性，简短性两大特征[21]，因而目前适用于网页去重，文件去重的聚类去重算法或者分段去重算法，因为时间复杂度的原因[22]~[23]不能直接应用于中文短文本的去重。

鉴于此，本文提出了将 Bloom Filter 或者 Tire 树算法，与 SimHash 算法有效的结合的算法框架，从而可以高效而精准的完成对中文短文本的去重处理工作。笔者在第三节对研究模型作出了整体介绍。本文第四节，对模型中涉及的三种主要算法进行了简要介绍与设计。本文第五节，主要是通过仿真算例对设计模型进行实验研究，以验证模型的有效性与可行性。文章第六部分则总结了实验结论，并指出了文章的创新与不足之处。

2 研究模型

在认识到短文本海量性，简短性两大特征的基础之上，本文同时考虑到了中英文之间的差别，以特征码提取为算法的基础思想，选取 Bloom Filter[24]与改进后的 Trie 树[25]进行中文短文本去重的算法设计工作，并在时间复杂度，准确率以及内存分配方面进行了研究。除此之外，本文同时注意到，基于上述两种算法的研究仅仅可以去除短文本信息中完全重复的部分，而无法对十分相似的文本进行甄别并有效的去除，故在数据集中仍然存在少部分的相似文本，在一定程度上会对信息的统计产生影响，从而影响决策的有效性。比如在微博客的统计中，他人对于一条原创微博客在更改一两字之后进行转发，统计时仍需将其单独算作一条原创微博客。在此种情况下，为了对经过完全重复去重之后的短文本信息再进行相似重复去重，本文又引入了 SimHash 算法。

由此可见，本文整体的研究模型为：（1）提取中文短文本数据集，进行数据预处理；（2）数据集首先经过 Bloom Filter 或者 Trie 树进行完全重复去重；（2）继而通过 SimHash 算法进行相似重复去重；（4）得到去重后结果集。

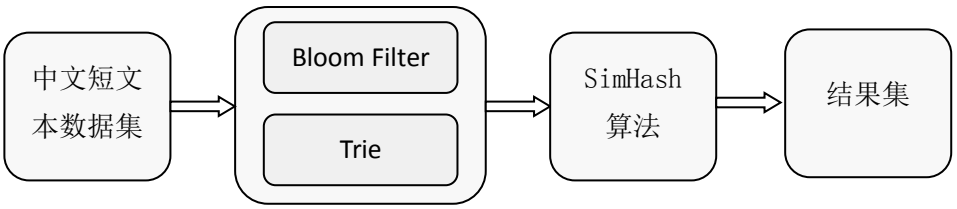


图 1 算法框架图

3 基本算法设计

由于在现实情况下，文本内容是人的自然语言，大量文本数据会呈现出半结构化以及非结构化的特征。因而，在文本数据中不仅仅存在大量完全重复的文本，同时也有一些相似重复的文本。而在相似重复文本比较中，相似性的比较往往需要较大的时间与空间开销，所以，一般情况下不易直接对大批量的文本数据进行重复的查询与处理工作。针对这种特点，为了有效的去除重复，在短文本处理的第一阶段，笔者将 Bloom Filter 与 Trie 树的算法思想引入，主要基于以下两点。由于出错率的牺牲，Bloom Filter 跳出了时间复杂度与空间复杂度不可兼得的情况[26]，因而在保证速度的同时，空间占用率极低；而对于 Trie 树而言，其本身较之于 Hash，有着更快的速度。

3.1 Bloom Filter 设计

Bloom Filter 是由 Howard Bloom 在 1970 年提出，是一种基于散列函数的数据查找算法。对于某一条数据元素，可以快速检测其是否为集合中的一个成员。与普通的散列表相比，它的优势在于，可以大幅度的节省空间。但是，这种算法存在着一定的错判率，即对于查找检测的请求，其会返回“数据存在（有可能错误 false positive）”以及“数据不在集合内（必然不存在）”两种结果。由此可见，Bloom Filter 算法是通过牺牲出错率来换取时间与空间。在此算法中误判的概率为：

$$P_{err} = (1-P_0)^k = (1-(1-\frac{1}{m})^{kn})^k \approx (1-e^{-kn/m})^k$$

（n 为数据集中元素的个数，k 为 Hash 函数个数，m 为分配位向量的大小。）

对于 Bloom Filter 的设计而言，关键点在于位向量大小的设计以及 Hash 函数的选择。二者直接影响到 Bloom Filter 的去重的出错率。笔者在内存中开辟出一块大小为 m 的位向量区域，并选取了稳定性能较好的 BKDR Hash 与 AP Hash 作为散列函数。

Bloom Filter算法的具体设计如下：

(1) 初始化一块 50000bit 的位向量区域；

(2) 对于待处理的 n 项短文本数据集 $S=\{s_1,s_2,s_3...s_n\}$ 中的 $s_i(i=1,2,3,4...n)$ 取 BKDR Hash 得 k_1 与 AP Hash 得 k_2；

(3) 其中取（$size$为位向量大小）：

$$mark_1 = Math.Abs(k_1\%(size/2)-1)$$

$$mark_2 = size-Math.Abs(k_2\%(size/2))-1$$

(4) 看上述得到的 $mark_1$ 与 $mark_2$ 是否在位向量中已经置为 1，若其中一个为 1，或者全部为 1，则 $s_i(i=1,2,3,4...n)$ 为重复文本，需要予以删掉；若全部为 0，则置为 1。

3.2 Trie 树的设计

Trie 树，即字典树。由于其可以获得比 Hash 表更高的查询效率，故在字符串查找、前缀匹配等中应用很广泛。Trie 树的核心思想是空间换时间。利用字符串的公共前缀来降低查询时间的开销以达到提高效率的目的。对于 Trie 树而言，其基本的性质有（1）根节点不包含字符，

除根节点外每一个节点都只包含一个字符；（2）从根节点到某一节点，路径上经过的字符连接起来，为该节点对应的字符串；（3）每个节点的所有子节点包含的字符都不相同。为了更为直观的表示，假设存在单词组，可得其对应字典树。

$$\{abede, aby, at, fcat, fce, ps, ptr, zaom, zat\}$$

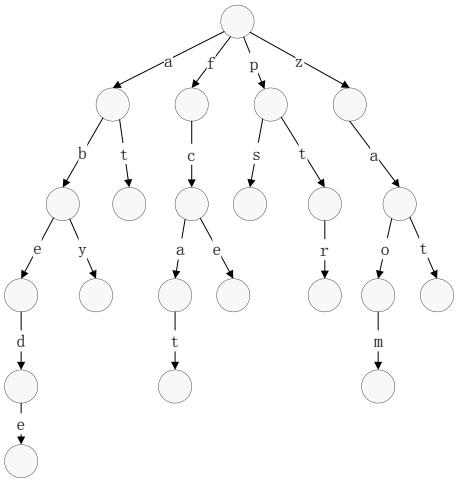


图2 Trie树（字典树）示意图

英文全部由 26 个字母构成，Trie 树的思想源泉就来源于对英文字典的查询。但是，考虑到常用中文有大约 2000 多字，据此建立 Trie 树则不切实际。基于此种考虑，笔者对 Trie 树做出了改进。对于中文短文本集合 $A=(a_1,a_2,a_3,a_4...a_n)$ 中的任意一个中文字符串 a_i，经过散列函数得到整型变量 Val，并取 $NVal=Val^2$ 之后，针对一组的 $NVal$ 构建数字 Trie 树。

改进后的Trie树具体设计如下：

(1) 初始化根节点，构建 Trie 树；

(2) 对于中文短文本集合 $A=(a_1,a_2,a_3,a_4...a_n)$ 中的任意一个中文字符串 a_i，经过散列函数（使用 Visual Studio 2010 C#内置 Hash 函数）得到整型变量 k_i；

(3) 取 $Nk_i=k_i^2$，则 Nk_i 由 m 位数字构成 $(x_1,x_2,x_3,x_4...x_m)$；

(4) 取其第一个数字 x_1，遍历 Trie 根节点下子节点，若存在 x_1，则取 x_2，遍历与 x_1 相等的子节点下的节点；若不存在，则创建一个节点为 x_1，并将剩余数字依层次在新建节点下建立节点。

3.3 SimHash 算法设计

对于 SimHash 而言，其是一种降维的技术，输入的一个向量通过程序运算之后输出一个 m 位的签名值。在文本数据处理中，其输入的向量为文本的特征向量集合（每个特征值配以一定的权重值）。而对于特征值的有效选取，直接决定了去重算法精度，在这一问题上 I-Match 算法，Shingling 算法以及 SpotSig 算法提供了基于语义以及基于语法的方法。笔者考虑到中文短文本的简短性，借鉴了 Shingling 的特征值选取方法的思想，即选择对连续的若干个单词的串（即下文所提到的 $Shingle$）进行操作。具体而言，若选取 $Shingle$ 的长度为 W，那么就会生成 $N-W+1$ 个 $Shingle$，然后通过 Hash 函数，可以得到 $N-W+1$ 个 Hash 值。据此输入 SimHash 进行运算：

(1) 将一个 m 维的向量 V 向量初始化为 0；

(2) 将取到的文档的每个特征码运用 Hash 算法哈希为 m 位 Hash 值。应用这些 m 位的 Hash 值，我们可以将向量 V 的 f 个元素增加或者减少其所对应的权重值的大小；

(3) 当所有的特征码处理完毕之后，向量 V 中有的元素为正，有的元素为负。

(4) 建立一个 f 位的二进制数 S，并将其初始化为 0。如果向量 V 的第 i 个元素大于 0，则 S 的第 i 位为 1，否则为 0；

(5) 将 S 作为文本的签名输出；

(6) 通过比较两个文本签名的 Hamming Distance[27]（两个二进制向量中不相同位的个数），可以得出相似程度[28]~[29]，并通过预先设定的相似度阈值去除相似文本。

4 实验算例研究

4.1 BloomFilter 与 Trie 树比较实验研究

4.1.1 对比速率研究实验

在大规模的文本处理中，Bloom Filter 速度快，占用空间小，适宜在精度要求不高的情况下选取；而对于 Trie 树而言，同样可以实现较好的速度。在本节中，笔者在文本数据不重复率 $(r=\frac{\text{不重复文本数据量}}{\text{总文本数据量}})$ 为 1%，5%，10%的情况下，用上述两种算法分别对文本数据进行了去重处理，并得出如下结果：

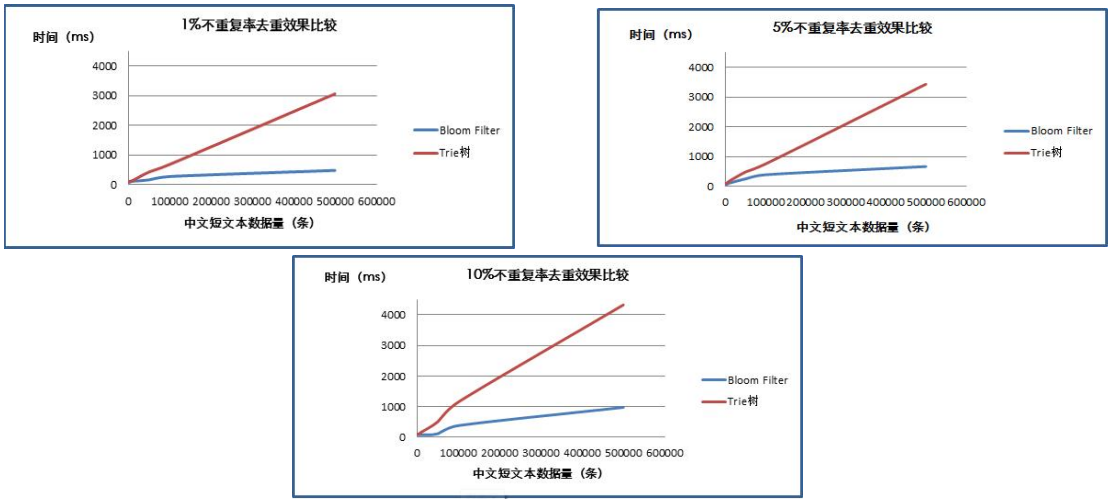


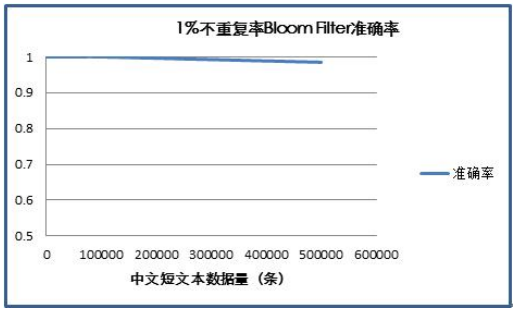
图 3 1% 5% 10%不重复率去重效果比较

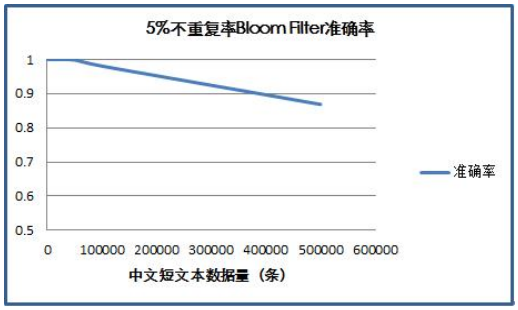
对上图分析可知，随着处理中文短文本数据量的增长，Bloom Filter 与 Trie 树的消耗时间均有所增加。其中 Trie 树较之于 Bloom Filter 而言，虽然有着精确度 100%的优势，但在处理海量文本时，其速率表现差强人意。另外，随着重复率 r 的增高，二者的速率均有所降低。

为了更加明显的比较海量数据条件下，Trie 树的低效率性，笔者比较了二者在处理 1000000，2500000 条 r 为 1%的文本数据时的表现。在五组实验中，当需要处理的中文短文本数据达到 2500000 条时，Bloom Filter 所需的平均处理时间为 3526.8ms（约 3.5s），而 Trie 树所需的平均处理时间为 14429.8ms（约 14.3s），并且由于二者的消耗时间与数据处理量成正比例关系，随着处理量的增多，二者的消耗时间差不断增大。

4.1.2 BloomFilter 准确率研究实验

除上述研究外，笔者考虑到 Bloom Filter 算法存在误判率，在位向量所占内存为 50000bit 的情况下，对算法的准确率($k=\frac{BloomFilter处理后的所得的不重复文本数据量}{不重复文本的数据量}$)进行了统计。





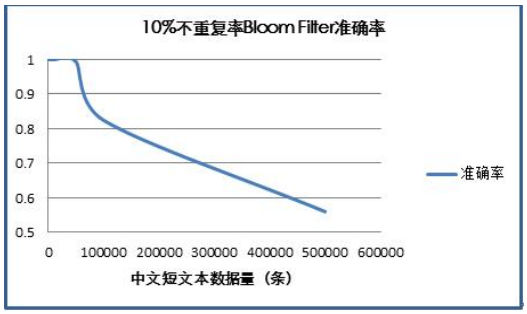


图 4 Bloom Filter 准确率比较

虽然 Bloom Filter 算法在处理中文短文本的去重问题时，时间与空间方面均有一定的优势。然而，如前文分析，随着中文短文本数据量的增加，算法的准确率 k 不断下降。并且，k 在很大程度上也受到了待去重文本不重复率 r 的影响。r 在待去重文本中所占比例越大，其准确率 k 的水平越低。在 10%不重复率的实验中，当中文短文本数据量达到 50000 条时，Bloom Filter 算法的准确率低至 56%左右，去重的效果已经受到极大的影响。

4.1.3 BloomFilter 内存分配研究实验

但是，上述关于准确率的实验是在分配内存 50000bit（约 6.1KB）的条件下进行的，不可否认后期数据量较大时导致其准确率较低是受到内存的制约，即在内存中开辟的位向量由于容量过低，使得非重复文本的 Hash 值映射到了位向量中的相同位置，进而导致误判而影响精度。为验证分析中关于准确率与内存之间的关系，笔者又进行了补充实验。

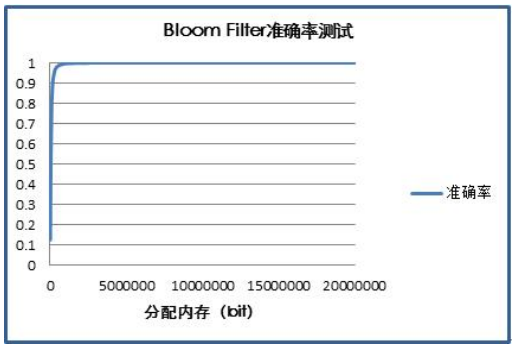


图 5 Bloom Filter 准确率测试

关于内存分配与算法准确率之间的关系，笔者在不重复率 $r=10\%$ 的条件中，对 500000 条中文短文本进行处理实验，选取实验内存区间在 10000bit (约 1.2KB)——20000000bit(约

2.4MB)。当分配内存在 160000bit 左右时，准确率升至 90%，并随着分配内存量的增加不断上升，趋向于 100%。当分配内存在 20000000bit 时，算法的准确率达到 100%。由此可见，在准确率要求并非很高的情况下，较小的内存量便可以满足去重的要求。在内存受限之时，甚至可以考虑循环采用 Bloom Filter 算法进行过滤。

综上所述，Bloom Filter 算法与 Trie 树算法对于需要完全去重的文本，在时间速率方面表现良好。而对于不需要过分注重精度的海量文本，在极低的空间开销的情况下，Bloom Filter 算法完全可以胜任。Trie 树虽然可以保证 100%的准确率，但是在海量数据的处理方面，仍要稍逊一筹。对于中文短文本处理的第一阶段，可以综合考虑实际情况，对上述算法进行选择。

4.2 SimHash 研究实验

在本实验中，笔者将向量V的维数定为 64 维，由于采用了 Shingling 特征值的选取思想，同时为了简化实验过程，实验中并未加入涉及权重值以及语料库的相关内容。在实验中，笔者分别对文本数据量为 100，500，1000，2000 的数据集进行数据相似去重处理工作（相似度阈值取 Hamming Distance<5），所得实验结果如下所示：

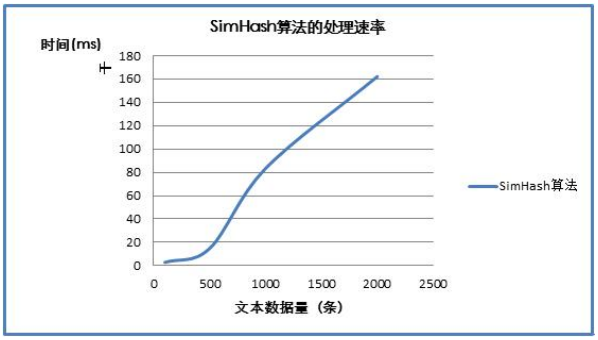


图 6 SimHash 算法的处理速率

在实验中，文本数据量在 100 条左右时，处理速率约为 2s 左右，而文本数据量在 1000 条左右时，处理速率约为 1 分钟左右，尚在可以接受的范围之内。随着文本数据量的不断增加，时间开销将逐步增大。这是由于 SimHash 算法在进行相似度的数值比较时需要进行成对比较，因而时间复杂度为$O(n^2)$。所以在数据量较低时，采用 SimHash 算法将比较有效，过高的数据量将严重影响算法的性能。在文本数据去重处理的第二阶段，如果处理数据量过大，可以考虑采用分布式的方法进行分摊处理。对于分布式的处理方法，不在本文讨论范围之内，在此并不赘述。

5 结语

本文的研究重点主要针对目前日益冗余但极富研究价值的中文短文本的去重工作。在综合文本去重领域现有的研究基础之上，本文提出了针对于中文短文本重复检测的解决算法，并对方案做了实例研究验证。实验证明，Bloom Filter 在处理中文短文本数据时能够在低内存占用的情况下，以可以接受的出错率换取较快的速度，Trie 与前者在速率上在同一数量级，但仍要逊色一些；SimHash 并不适宜直接处理中文短文本去重工作，其成对比较的低效性需要第一阶段的有力配合。然而，不容忽视的是，本文的实验研究在第二阶段并未充分引入语料库的因素，因而弱化了相似去重中语义的影响。关于中文短文本去重领域在 SimHash 算法中引入语料库以强化权重的部分，还有待于后人继续研究。

参考文献

[1] 耿崇,薛德军.中文文档复制检测方法研究[J].现代图书情报技术,2007(6).

[2] 曹玉娟,牛振东,赵堃,彭学平.基于概念和语义网络的近似网页检测算法[J].软件学报,2011,22(8).

[3] 鲍军鹏,沈钧毅,刘晓东,宋擒豹.自然语言复制检测研究综述[J].软件学报,2003,14(10).

[4] Manber U. Finding similar files in a large file system[C]. California: Proceedings of the Winter USENIX Conference, 1994:1-10.

[5] Heintze N. Scalable document fingerprinting. In: Proceedings of the 2nd USENIX Workshop on Electronic Commerce. 1996. http://www.cs.cmu.edu/afs/cs/user/nch/www/koala/main.html.

[6] Broder AZ, Glassman SC, Manasse MS. Syntactic clustering of the Web. In: Proceedings of the 6th International Web Conference.1997.http://gatekeeper.research.compaq.com/pub/
DEC/SRC/technical-notes/SRC-1997-015-html/.

[7] Brin S, Davis J, Garcia—Molina H. Copy detection mechanisms for digital documents [C]. California: Proceedings of the ACM SIGMOD Annual Conference, 1995:98-409.

[8] Monostori K, Zaslavsky A, Schmidt H. MatchDetectReveal: Finding overlapping and similar digital documents. In: Proceedings of the Information Resources Management Association International Conference (IRMA2000). 2000. http://www.csse.monash.edu.au/ projects/MDR/papers/.

[9] Monostori K, Zaslavsky A, Vajk I. Suffix vector: A space-efficient representation of a suffix tree. Technical Report, 2001.

[10] Wise MJ. YAP3: Improved detection of similarities in computer programs and other texts. In: Proceedings of the SIGCSE'96. 1996, 130~134. http://citeseer.nj.nec.com/wise96yap.html.

[11] Chowdury A, Frieder O, Grossman D et al. Collection statistics for fast duplicate document detection [J]. ACM Transactions on Information Systems (TOIS), April 2002, 20(2):171-191.

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

[13] Shivakumar N, Garcia-Molina H. SCAM: A copy detection mechanism for digital documents. In: Proceedings of the 2nd International Conference in Theory and Practice of Digital Libraries (DL'95). 1995. http://www-db.stanford.edu/~shiva/ publns.html.

[14] Shivakumar N, Garcia-Molina H. Building a scalable and accurate copy detection mechanism. In: Proceedings of the 1st ACM Conference on Digital Libraries (DL'96). 1996. http://www-db.stanford.edu/~shiva/publns.html.

[15] Garcia-Molina H, Gravano L, Shivakumar N. dSCAM: Finding document copies across multiple databases. In: Proceedings of the 4th International Conference on Parallel and Distributed Systems (PDIS'96). 1996. http://www-db.stanford.edu/~shiva/publns.html.

[16] Antonio Si, PHong Va Leong, PRynson W.H.Lau. CHECK: a document plagiarism detection system. Apr.1997, Proceedings of the 1997 ACM symposium on Applied Computing[C].

[17] Jung-Sheng Yang, Chih-Hung Wu. Plagiarism detection of text using knowledge-based techniques[C]. Design and application of hybrid intelligent systems, 2003:973-982.

[18] BRIN S, DAVIS J, GARCIA-MOLINA H. Copy detection mechanisms for digital documents [C/OL].Proceedings of the ACM SICMOD Annual Conference.1995:398-409 [2007 -05- 10] .http://www-db.stanford.edu/pub/brin/1995/
copy.ps.

[19] M.Theobald, J Siddharth. A.Paepcke. SpotSigs: Robust and Efficient Near Duplicate Detection in Large Web Collections[C]. Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, 2008:563-570.

[20] 李志义,梁士金. 国内网页去重技术研究：现状与总结[J].图书情报工作.2011, 55(7).

[21] 杨虎,杨树强,韩伟红等. 基于关联规则和特征码的快速去重方法[C].中国计算机大会,2007.
[22] 王曰芬,章成志,张蓓蓓,吴婷婷.数据清洗研究综述[J].现代图书情报技术,2007(6).
[23] 向培素. 聚类算法综述[J].西南民族大学学报(自然科学版),2011,5.
[24] Bloom B. Space/time Tradeoffs in Hash Coding with Allowable Errors[J]. Communication of the ACM, 1970, 13(7):422-426.
[25] http://zh.wikipedia.org/wiki/Trie#cite_note-DADS-0[OL][2012-05-07].
[26] 丁振国,吴宝贵,辛友强.基于 Bloom Filter 的大规模网页去重策略研究[J].现代图书情报技术,2008(3) .
[27] 李彬,汪天飞,刘才铭,张建东. 基于相对 Hamming 距离的 Web 聚类算法[J]. 计算机应用, 2011, 31(5).
[28] 韩冰,林鸿飞. 基于语义结构的科技论文抄袭检测[J].情报学报.2010, 29(3).
[29] 樊勇,郑家恒. 网页去重方法研究[J].计算机工程与应用.2009，45(12).

作者联系方式：
高翔　深圳市南山区西丽镇丽水路深圳大学城北大校区 C 栋 邮编:518055　15210903764
gx8600@126.com
李兵　对外经济贸易大学信息学院（北京市朝阳区惠新东街 10 号）邮编:100029　13439788086
lb0501@126.com