### [相似数据检测算法](http://blog.csdn.net/liuben/article/details/6897314)

分类： [计算机理论](http://blog.csdn.net/liuben/article/category/616625) [数据结构与算法](http://blog.csdn.net/liuben/article/category/373757)2011-10-22 22:21 4893人阅读 [评论](http://blog.csdn.net/liuben/article/details/6897314#comments)(20) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/liuben/article/details/6897314#report)

相似数据检测算法对给定的一对数据序列计算两者之间的相似度([0,1], 1表示完全相同)或距离([0, ), 0表示完全相同)，从而度量数据之间的相似程度。相似数据检测在信息科学领域具有非常重要的应用价值，比如搜索引擎检索结果的聚类与排序、数据聚类与分类、Spam检测、论文剽窃检测、重复数据删除、Delta数据编码等应用。正是由于它的重要性，近年来成为了研究的重点，不断有新检测方法涌现并得到评估。其中，Broder提出的shingling算法和Charikar的simhash算法被认为是目前为止最好的算法。  
  
对于相似数据检测，最为简单地可以采用类似Unix diff的方法。Unix diff对文档进行逐行对比来检测相似文件，它采用经典的LCS(Longest Common Subsequence，最长公共子串)算法，运用动态规划方法来计算相似性。LCS的含义是同时包含在字符串里的一个最长字符序列，LCS的长度作为这两个字符串相似性的度量。Diff算法以整行作为"字符"来计算最长公共子串，性能上比字符级的LCS算法快很多。这种方法效率很低，而且只适用文本文件的相似比较，不能直接适用于二进制文件。为此，研究者们提出为每个文档提取一组特征，这样将文件相似性问题转换为集合相似性问题，如基于shingle的计算方法。这种方式的核心思想是为每个文件提取组特征值，以特征值集合来计算相似性，从而降低空间和计算复杂性来提高性能。  
  
经过对shingle算法和simhash算法以及笔者基于bloom filter实现算法的分析，相似数据检测算法大致流程如下：  
(1) 将数据段分解成一组shingle(即子序列或数据块)，可以采用定长、变长、单词或段落(文本文件)等分块算法；  
(2) 为了降低空间和时间计算复杂性，可以对shingle集合进行抽样，比如Min-Wise，Modm，Mins方法；  
(3) 基于选定的shingle集合为数据文件抽取特征，通常是为每个shingle计算hash值组成的序列作为特征值；  
(4) 为了降低空间和时间计算复杂性，可以对文件特征进行降维处理，比如simhash和bloom filter；  
(5) 基于文件特征计算两个数据对象之间的相似性，计算方法有Cosine、Overlap、Dice、Jaccard或Hamming距离。  
  
**Shingle算法**  
Shingle算法的核心思想是将文件相似性问题转换为集合的相似性问题，集合的相似性度量方法主要有resemblance 和containment两种，其定义如下。

**|shingle(f1, w) ∩ shingle(f2, w)|  
　Rw(f1, f2) = ----------------------------------------------  
                           |shingle(f1, w) ∪ shingle(f2, w)|  
  
                           |shingle(f1, w) ∩ shingle(f2, w)|  
　Cw(f1, f2) = ----------------------------------------------  
                                          |shingle(f1, w)|**

数量较大时，如果对所有shingle进行相似性处理则系统开销较大，包括内存和CPU资源。这时就可以考虑对shingle集合进行抽样，以降低空间和时间计算复杂性，但同时由于样本覆盖率有限，相似性精确度会有所降低。shingle取样主要有三种方法，即Min-Wise，Modm，和Mins。Min-Wise技术是通过将shingle的长度w和整数值进行映射产生随机哈希的公共集，在此相同的模式下进行随机最小独立置换的采样，从而得到采样集合；Modm 技术是通过在与Min-Wise同样的公共映射集中选择所有模m为0 的哈希值对应的shingle组成取样集合；Mins技术同样也是先将shingle和整数集进行映射，然后从中选择最小s个元素组成取样集合。此外，还可以使用shingle的hash值代表shingle进行相似性计算，能够节省一定计算开销。  
  
**Simhash算法**  
Shingle算法的空间和时间计算复杂性都比较高，对于大数据集的Simlarity Join问题将难以适用。Charikar的simhash算法的核心思想是用一个b位的hash值来表示文件的特征值，然后使用simhash之间的Hamming距离来衡量相似性。Hamming距离的定义为，两个二进制序列中对应位不同的个数。simhash的计算方法如下：  
(1) 将一个b维的向量V初始化为0，b位的二进制数s初始化为0;  
(2) 对每一个shingle，用hash函数(如MD5, SHA1)计算一个b位的签名h。对i=1到b，如果h的第i位为1，则V的第i个元素加上该特征权重；否则，V的第i个元素减去该特征权重；  
(3) 如果V的第i个元素大于0，则s的第i位为1，否则为0；  
(4) 输出s作为simhash。  
与传统hash函数相比，simhash具有一个这样的显著特征，即越相似的文件具有越相似的simhash值，也就是说Hamming距离越小。显而易见，Simhash仅使用b位的hash值来表示文件 的特征，节省了大量的存储开销；Hamming距离计算简单高效，Simhash使用Hamming距离来衡量相似性，计算复杂性得到大大降低。简而言之，simhash算法通过对文件特征的降维，有效解决了Shingle算法的高空间和时间计算复杂性问题。然而，simhash算法的精确度也会有所损耗，并且与simhash的位数b有关，b越大精确度越高。  
  
**Bloom filter算法**  
与Simhash算法本质相似，Bloom filter算法的核心思想也是着眼于文件特征的降维，它使用Bloom filter数据结构来表示特征值。Bloom filter是一个空间效率很高的数据结构，它由一个位数组和一组hash映射函数组成。Bloom filter可以用于检索一个元素是否在一个集合中，它的优点是空间效率和查询时间都远远超过一般的算法，缺点是有一定的误识别率和删除困难。使用Bloom filter进行相似数据检测，可以弥补shingle中应用特征集交集计算文件相似性所导致的高计算和存储空间开销，在性能与相似性匹配精度之间取得平衡。Bloom filter构造方法如下：  
(1) 构造一个m位的bloom filter数据结构bf，并将所有位初始为0；  
(2) 选定两个hash函数作为映射函数，分别为hash1，hash2；  
(3) 对每一个shingle，分别应用hash1和hash2，并对bf相应比特位置1；  
(4) 输出bf作为文件特征值。  
这样，两个文件相似性计算就转换成两个bloom filter的相似性计算，越相似的文件在它们的bloom filter中有更多共同的1。由于Bloom filter具有有限的误识别率的特性，相似性算法精确度取决于Bloom filter的大小，越大则精确度越高，同时存储空间消耗也越大。Bloom filter同样可以使用Hamming距离衡量相似性，也可以使用Cosine、Overlap、Dice、Jaccard等方法来度量。Hamming距离前面已有定义，这里介绍一下后四种方法的计算公式。

**dot(x, y)  
Cosine\_sim(x, y) = -----------------  
                                    sqrt(|x|.|y|)  
  
  
                                       dot(x, y)  
Overlap\_sim(x, y) = -----------------  
                                     min(|x|, |y|)  
  
                                 2.dot(x, y)  
Dice\_sim(x, y) = -----------------  
                                   |x| + |y|  
  
                                            dot(x, y)  
Jaccard\_sim(x, y) = ------------------------  
                                      |x| + |y| - dot(x, y)**

其中，dot(x, y) = Σx[i].y[i]，在这里相当于两个Bloom filter数据结构中同时为1的位数；|x|表示bloom filter数据结构中为1的位数。相似性计算函数如下：

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/liuben/article/details/6897314)

1. **static** **double** bloom\_sim(BLOOM \*bloom1, BLOOM \*bloom2)
2. {
3. **int** i, r1, r2;
4. **int** c1 = 0, c2 = 0, comm = 0;
5. **double** sim;
7. **for** (i = 0; i < BLOOM\_ARRAY\_SZ; i++) {
8. r1 = bloom\_check(bloom1, 1, i);
9. r2 = bloom\_check(bloom2, 1, i);
10. **if** (r1 && r2) {
11. comm++;
12. c1++;
13. c2++;
14. } **else** {
15. **if** (r1) {
16. c1++;
17. }
19. **if** (r2) {
20. c2++;
21. }
22. }
23. }

26. /\* similarity measures \*/
27. //sim = comm/(sqrt(c1) \* sqrt(c2)); /\* Cosine \*/
28. //sim = comm/1.0/(c1 + c2 - comm); /\* Jaccard \*/
29. //sim = comm\*2.0/(c1 + c2); /\* Dice \*/
30. sim = comm\*1.0/(c1<c2?c1:c2); /\* Overlap \*/
31. **return** sim;
32. }

**三种算法对比**  
Shingle算法的空间和计算复杂性高，相似性精度也高，适合数据量不大且对精度要求高的应用。Simhash和bloom filter算法在空间消耗和计算复杂性方面都优于Shingle算法，但是精度有所损耗，取决于simhash的长度和bloom filter的大小。simhash的长度通常为64位或128位，这个基本可以满足应用的需求，可以根据实际需求增大位数。bloom filter要大于simhash长度，可以根据最大shingle数的两倍来估算，精度方面也要优于simhash。由于hash函数的碰撞问题，simhash和bloom filter算法可能出现误判现象，即不相似的文件可能会判定为相似的。总结一下，通常情况下，文件特征值存储空间消耗方面，Shingle > bloom filter > simhash；相似性计算精度方面，Shingle < bloom filter < simhash。Bloom filter算法往往是比较折中的相似数据检测方法选择，但海量数据集的相似性计算往往采用simhash算法，在计算性能方面具有很大优势，而且更加适合MapReduce计算模型。