网页去重的算法，这里我想讨论下这些算法

1. I-Match

2. Shingliing

3. SimHashing（ locality sensitive hash）

4. Random Projection

5. SpotSig

6. combined

**I-Match算法**   
I-Match算法有一个基本的假设说：不经常出现的词和经常出现的词不会影响文档的语义，所以这些词是可以去掉的。   
算法的基本思想是：将文档中有语义的单词用hash的办法表示成一个数字，数字的相似性既能表达文档的相似性   
算法的框架是：   
1. 获取文档（或者是主体内容）   
2. 将文档分解成token流，移除格式化的标签   
3. 使用term的阈值（idf），保留有意义的tokens   
4. 插入tokens到升序排列的排序树中   
5. 计算tokens的SHA1   
6. 将元组（doc\_id,SHA hash) 插入到某一词典中，如果词典有冲突，这两个文档相似。

算法有一个缺点是稳定性差。如果文档的某个词改变了，最终的hash值就会发生显著的变化。对空文档，算法是无效的。   
有一个解决办法是，用随机化的方法，参考[Lexicon randomization for near-duplicate detection with I-Match](http://portal.acm.org/citation.cfm?id=1400281)。具体细节这里就不提了

**Shingling算法**   
Shingling算法说，I-Match以词为单位做hash显然是不准确的，因为它忽略了文档之间的顺序。另，Shingle指的是连续的若干个单词的串。   
Shingling 算法有个简单的数学背景。如果一个shingle的长度为k，那么长度为n的文档就有n-k+1个shingle，每一个shingle可以用MD5或者 其他算法表示成一个fingerprint，而两个文档的相似性Jacard相似性来表示，Jarcard公式是指两个集合的相似性=集合之交/集合之 并。为了估计两个文档的相似性，有时候n-k+1个fingerprint还是太大了，所以取m个fingerprint函数，对每一个函数fi，都可以 计算出n-k+1个fingerprint，取其中的最小的fingerprint，称为i-minvalue. 那么一个文档就有m个i-minvalue。数学上，Broder大师说：

        平均来讲，两个文档中相同的唯一single的比率和两个文档中相同的i-minvalue的比率是一样的

Shingling的算法框架是：   
1. 获取文档（或者是主体内容）   
2. 将文档分解成n-k+1个shingle，取m个fingerprint函数，对每一个fingerpint函数计算i-minvalue值   
3. 将m个i-minvalue值组合成更少m’个surpersingle   
4.计算两个文档相同的surpergingle的个数a。   
5. 如果a大于某一个值b（say：2），那么两个文档Jarcard 相似

一般的参数设置为：m=84，m’=6，b=2

**SimHash 算法**

locality sensitive hash算法博大精深。基本思想是，如果两个东西相似，我可以用一个hash函数把他们投影到相近的空间中[LSH](http://en.wikipedia.org/wiki/Locality_sensitive_hashing" \t "_blank)。用到near duplication detection上，算法框架是：   
1. 将文档转换为特征的集合，每一个特征有一个权重   
2. 利用LSH函数把特征向量转换为f位的fingerprint，如：64   
3. 查找fingerprint的海明距离

haha,看，多么简单和明朗，这里的几个问题及时寻找正确的LSH

**Random Projection算法**shingling关注了文档顺序，但是忽略了文档单词出现的频率，random projection说我要讨论文档的频率。

Random Projection也是很有意思的一种算法，它是一种随机算法。简单描述为：   
1. 将每一个token映射到b位的空间。每一个维度是由{-1,1}组成。对所有页面投影函数是一样的   
2. 每一个页面的b维度向量，是所有token的投影的简单加和   
3. 最后把b维向量中的正数表示为1，负数和0都写成0   
4. 比较两个page的b维向量一致的个数

Charikar 最牛的地方是，证明，两个b位变量一致的位数的比率就是文档向量的consine相似性。这里的数学基础还是很有意思的，如果感兴趣，可以参考M.S. Charikar. Similarity Estimation Techniques for Rounding Algorithm(May 2002)

**SpotSig算法**

**ref:SpotSigs:Robust and Efficient Near Duplicate Detection in Large Web Collection**SpotSig 是个比较有意思的算法，它说，我为什么要关注所有的单词啊，我要关注的单词是有语义的词，哪些是有语义的词呢？哦，想 the a this an 的等虚词后面的就是我要关注的东西罗。Spot就是指这些虚词的后面的词串。然后呢，每一个文档我都有很多很多Spot了，现在一个文档就是一个Spot 的集合，两个文档是相似程度就是集合的Jaccard相似度。算法虽然简单，但是我想重点是两个比较有借鉴意义的工程上的性能考虑。

     1. Optimal Partition

     Sim（A，B） = | A B交集| / | A B 并集| <= min(A,B)/max(A,B) <= |A|/|B| say: |A|<|B|

好 了，这是一个很好的枝剪条件，如果文档spot vector的个数比小于某个值（当然是，小 / 大），就可以完全不用求交，并了。Optimal Partition就是说，好啊，我把每一个文档的spot vector的长度都投影到相应的从小到大的bucket中，保证|d1|/|d2| >=r if |d1| < |d2| . 且不存在这样的反例。另一个保证是这个bucket是满足条件的最小的。有了这个partition，我们最多只用关心相邻的三个bucket了

   2. Inverted Index Pruning

   说，两个文档，如果能相似，起码有一个公共的spot。逆向索引说的就是把spot做为index，包含它的所有文档作为其value。

有了这两个工具，计算复杂度可以明显下降，因为它不会计算不能是duplication的文档。