# 实现标题相似度计算，文本内容相似度匹配SimHash计算标题文本内容相似度

## ****1.场景：****

在很多公司的文件管理系统中，都有类似于对比多篇文章的相似度，例如在写公众号推文时，如果标记了原创，就会对比当前文章和库里已存在文章的相似程度，如果相似度过于高，则标记为原创的文章无法实现推送，那么，该功能是如何实现的呢？可以参考如下思路。

## ****2.算法：****

此例子借助的是[海明距离](https://so.csdn.net/so/search?q=%E6%B5%B7%E6%98%8E%E8%B7%9D%E7%A6%BB&spm=1001.2101.3001.7020)的实现方式。

通过SimHash算法并结合实际的代码来实现了相似度的计算

# 一、关于SimHash

SimHash算法是Google在2007年发表的论文《Detecting Near-Duplicates for Web Crawling》中提到的一种指纹生成算法，被应用在Google搜索引擎网页去重的工作之中。SimHash值不但提供了原始值是否相等这一信息，还能通过该值计算出内容的差异程度。

简单的说，SimHash算法主要的工作就是将文本进行降维，生成一个SimHash值，也就是论文中所提及的“指纹”，通过对不同文本的SimHash值进而比较“海明距离”，从而判断两个文本的相似度。

对于文本相似度的问题，常见的解决办法有欧式距离、编辑距离、最长公共子串、余弦算法、Jaccard相似度等方法。但是这些方法并不能对海量数据高效的处理。

比如说，在搜索引擎中，会有很多相似的关键词，用户所需要获取的内容是相似的，但是搜索的关键词却是不同的，例如：  
“有哪些是养猫必须知道的冷知识？”  
“有那些冷知识是养猫必须知道的？”  
以上两个可以等价的关键词，然而通过普通的hash计算，会产生两个相差甚远的hash串。而通过SimHash计算得到的Hash串会非常的相近，从而可以判断两个文本的相似程度。

## **补充知识**

## ****一）、什么是海明距离****

海明距离是编辑距离其中之一，在信息编码中，两个合法代码对应位上编码不同的位数称为码距，又称海明距离。即两个码字的对应比特取值不同的比特数称为这两个码字的海明距离。一个有效编码集中任意两个码字的海明距离的最小值称为该编码集的海明距离。例如： 10101 和 00110 从第一位开始依次有第一位、第四、第五位不同，则海明距离为 3。  
N位的码字可以用N维空间的超立方体的一个顶点来表示，两个码字之间的海明距离就是超立方体两个顶点之间的一条边，而且是这两个顶点之间的最短距离。

## ****二）、海明距离的应用****

海明距离应用最多的是在海量短文、文本去重上，以其性能优的特点。海明距离主要就是对文本进行向量化，或者说是把文本的特征抽取出来映射成编码，然后再对编码进行异或计算出海明距离。

## ****三）、什么是编辑距离****

编辑距离又称Levenshtein距离（也叫做Edit Distance），是指两个字串之间，由一个转成另一个所需的最少编辑操作次数，许可的编辑操作包括将一个字符替换成另一个字符，插入一个字符，删除一个字符。

# 二、SimHash算法的几何意义和原理

## 一）、SimHash算法的几何意义

它首先将每一个特征映射为f维空间的一个向量，这个映射规则具体是怎样并不重要，只要对很多不同的特征来说，它们对所对应的向量是均匀随机分布的，并且对相同的特征来说对应的向量是唯一的就行。比如一个特征的4位Hash签名的二进制表示为1010，那么这个特征对应的4维向量就是(1,-1,1,-1)T，即Hash签名的某一位为1，映射到的向量的对应位就为1，否则为-1。然后，将一个文档中所包含的各个特征对应的向量加权求和，加权的系数等于该特征的权重。得到的和向量即表征了这个文档，我们可以用向量之间的夹角来衡量对应文档之间的相似度。最后，为了得到一个f位的签名，需要进一步将其压缩，如果和向量的某一维大于0，则最终签名的对应位为1，否则为0。这样的压缩相当于只留下了和向量所在的象限这个信息，而64位的签名可以表示多达264个象限，因此只保存所在象限的信息也足够表征一个文档了。

明确了SimHash算法的几何意义，使这个算法直观上看来是合理的。但是，为何最终得到的签名相近的程度，即可以衡量原始文档的相似程度呢。在SimHash的发明人Charikar的论文中并没有给出具体的SimHash算法和证明，以下列出证明思路：

SimHash是由随机超平面Hash算法演变而来的，随机超平面Hash算法非常简单，对于一个n维向量v，要得到一个f位的签名(f<<n)，算法如下：   
1、随机产生f个n维的向量r1,…rf；  
2、对每一个向量ri，如果v与ri的点积大于0，则最终签名的第i位为1，否则为0；

这个算法相当于随机产生了f个n维超平面，每个超平面将向量v所在的空间一分为二，v在这个超平面上方则得到一个1，否则得到一个0，然后将得到的 f个0或1组合起来成为一个f维的签名。如果两个向量u,v的夹角为θ，则一个随机超平面将它们分开的概率为θ/π，因此u, v的签名的对应位不同的概率等于θ/π。所以，我们可以用两个向量的签名的不同的对应位的数量，即海明距离，来衡量这两个向量的差异程度。

SimHash算法与随机超平面Hash如何联系起来？在SimHash算法中，并没有直接产生用于分割空间的随机向量，而是间接产生->第 k个特征的Hash签名的第i位拿出来，如果为0，则改为-1，如果为1则不变，作为第i个随机向量的第k维。由于hash签名是f位的，因此这样能产生 f个随机向量，对应f个随机超平面。下面举个例子：   
假设用5个特征w1,…,w5来表示所有文档，现要得到任意文档的一个3维签名。  
假设这5个特征对应的3维向量分别为：   
h(w1) = (1, -1, 1)T   
h(w2) = (-1, 1, 1)T   
h(w3) = (1, -1, -1)T   
h(w4) = (-1, -1, 1)T   
h(w5) = (1, 1, -1)T

按SimHash算法，要得到一个文档向量  
d=(w1=1, w2=2, w3=0, w4=3, w5=0) T的签名，先要计算向量  
m = 1\*h(w1) + 2\*h(w2) + 0\*h(w3) + 3\*h(w4) + 0\*h(w5) = (-4, -2, 6) T，  
然后根据SimHash算法的步骤3，得到最终的签名s=001。

上面的计算步骤其实相当于->先得到3个5维的向量：  
第1个5维向量由h(w1),…,h(w5)的第1维组成：r1=(1,-1,1,-1,1) T;  
第2个5维向量由h(w1),…,h(w5)的第2维组成：r2=(-1,1,-1,-1,1) T;   
第3个5维向量由h(w1),…,h(w5)的第3维组成：r3=(1,1,-1,1,-1) T;

按随机超平面算法的步骤2，分别求向量d与r1,r2,r3的点积:   
d T r1=-4 < 0，所以s1=0;   
d T r2=-2 < 0，所以s2=0;   
d T r3=6 > 0，所以s3=1;  
故最终的签名s=001，与SimHash算法产生的结果是一致的。

从上面的计算过程可以看出，SimHash算法其实与随机超平面Hash算法是相同的，SimHash算法得到的两个签名的海明距离，可以用来衡量原始向量的夹角。这其实是一种降维技术，将高维的向量用较低维度的签名来表征。衡量两个内容相似度，需要计算海明距离，这对给定签名查找相似内容的应用来说带来了一些计算上的困难。

SimHash算法的输入是一个向量，输出是一个 f 位的签名值。为了表述方便，假设输入的是一个文档的特征集合，每个特征有一定的权重，比如特征可以是文档中的词，其权重可以是这个词出现的次数。

## 二）、SimHash的计算原理

SimHash算法主要分为五个过程：分词、Hash、加权、合并、降维。如图实例[图示-01]  
1、分词  
对给定的一段文本进行分词，产生n个特征词，并赋予每个特征词一个权重。

2.Hash  
通过hash函数对每一个词向量进行映射，产生一个n位二进制串。

3.加权  
经过前面的计算已经得到了每个词向量的Hash串和该词向量对应的权重，第三步计算权重向量W=hash\*weight。

4.合并  
对于一个文本，计算出了文本分词之后每一个特征词的权重向量，在合并这个阶段，把文本所有词向量的权重向量相累加，得到一个新的权重向量。

5.降维  
对于前面合并后得到的文本的权重向量，大于0的位置1，小于等于0的位置0，就可以得到该文本的SimHash值。

SimHash算法步骤如下：   
1、将一个 f 维的向量 V 初始化为 0 ；f 位的二进制数 S 初始化为 0；  
2、对每一个特征用传统的 Hash 算法对该特征产生一个 f 位的签名 b。对 i=1 到 f ：   
如果b 的第 i 位为 1 ，则 V 的第 i 个元素加上该特征的权重；   
否则，V 的第 i 个元素减去该特征的权重；  
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 作为签名；

## 三）、文本的相似度计算

对每条文本根据SimHash 算出签名后，再计算两个签名的海明距离（两个二进制异或后1的个数）即可。根据经验值，对64位的SimHash，海明距离在3以内的可以认为相似度比较高。  
假设对64位的SimHash，查找海明距离在3以内的所有签名。可以把64位的二进制签名均分成4块，每块16位。根据鸽巢原理（也称抽屉原理，见组合数学），如果两个签名的海明距离在3以内，它们必有一块完全相同。  
把上面分成的4块中的每一个块分别作为前16位来进行查找。建立倒排索引。