Algorithm　Min Hashing

preliminaries:

a. Jaccard 相似度：

可以参考我写的文档

b. k-shingle：

可以参考我写的文档，不知道 k-shingle 也没关系，知道的话更有助于帮助理解算法使用环境背景

background:

a. 假如我们有若干份文档，想对比这若干份文档之间的相似度，我们现在对每一份文档使用 k-shingle 算法比如我的某一个文档 1 中的内容为“A document is a string of characters”，我们用 3-shingle 算法可以得到集合 S1{ “A d”, “do”, “doc”, “ocu”, “cum”, “ume”, “men”, “ent”, . . . , “ers” }，对每一份文档都用 k-shingle 可以得到每一份文档的 shingle 集合 S2,S3...。

问题来了，为什么我们要用 Min-Hashing 呢？ Min-Hashing 又有什么用呢？

你有一份文档，假如你想用 5-shingle 来提取出这个文档的特征集合，那你的集合中元素的数量为 $27^5 = 14348907$ ，其中 27 代表 26 个字母加空格符，一个文档你就要存这么多特征，海量数据中文档起码上亿了，这你要存(14348907×几亿)的数据吗？那显然是不可取的，所以我们希望可以降一降这个 14348907 这个数字，所以引入了 Min-Hashing 方法，再尽最大可能保留文档之间的 Jaccard 相似度的同时大大降低了每个文档的特征维度！

我们还在 Locality-sensitive-hashing 中会讲到如何降低那搜索文档的数量。这里不涉及。

Min Hashing：

基本原理：

注意这里讲的只是基本原理，或者 Min Hashing 实际计算中的一部分，看完这个基本原理，你只能知道这个 Min-Hashing 的灵魂，但并不能知道到底应该怎么实际操作！所以我们先介绍基本原理，再介绍实际步骤

step 1. 我现在有 S1、S2、S3 集合，S1 = {A,D}、S2 = {B}、S3 = {D}, 构造 0-1 矩阵如下：

	S1	S2	S3
A	1	0	0
B	0	1	0
C	0	0	0
D	1	0	1

step 2. 对所有集合的行进行随机置换运算

	S1	S2	S3
B	0	1	0
D	1	0	1
A	1	0	0
C	0	0	0

step 3. 得到最小哈希值：

每一个集合的最小哈希值为随机置换后该集合那一列元素的第一个非 0 得元素的行号索引，即 $\mathrm{h}_{\min}(S_1)=\mathrm{D}$，$\mathrm{h}_{\min}(S_2)=\mathrm{B}$，$\mathrm{h}_{\min}(S_3)=\mathrm{D}$。

现在就要讨论 Min-Hashing 的核心了，为什么两个集合的最小哈希值相等的概率等于这两个集合的 Jaccard 相似度，为什么定义最小哈希？

假如我们只考虑集合 S_1 和集合 S_2，则这两列所在的行会存在以下三种类型:

a. 若 $S_{1,j}$ 与 $S_{2,\mathrm{j}}$ = 1 （其中 j 代表第 j 行），则记该事件为 X

b. 若 $S_{1,j}$ 或 $S_{2,\mathrm{j}}$ = 1，则记该事件为 Y

c. 若 $S_{1,j}$ 与 $S_{2,\mathrm{j}}$ = 0，则记该事件为 Z

S_1 和 S_2 交集的元素个数为 x，并集的个数为 x+y， 所以 $J(S_1,S_2)=\dfrac{x}{x+y}$,其中 J(A,B)为 Jaccard 相似度函数。

一个 0-1 矩阵经过随机打乱后，从上往下扫描，由于特征矩阵可能比较稀疏，所以导致两个集合之间大部分的行都发生 Z 事件。不过只有 X、Y 事件才决定相似度，所以 Z 事件也对相似度不起什么贡献。如果我们删除所有 Z 类，那么第一行发生的事件非 X 即 Y。如果第一行是 X 事件，则有 $\mathrm{h}_{\min}(S_1)=\mathrm{h}_{\min}(\mathrm{S}_2)$，而发生 {删除所有 C 事件的行，剩下的行中第一行发生 X 事件} 的事件的概率为 $\dfrac{x}{x+y}$ 因此：

$$probablity(h_{\min}(S_1)=h_{\min}(S_2))=\frac{x}{x+y}=Jaccard(S_1,S_2)$$

那这个 $probablity(h(S_1)=h(S_2))$ 的概率又应该怎么求？

我们刚才只用了一个随机置换，求了一个随机置换的最小哈希，那假如说我们用 3 个随即置换，求三个随机置换的最小哈希，会怎么样？

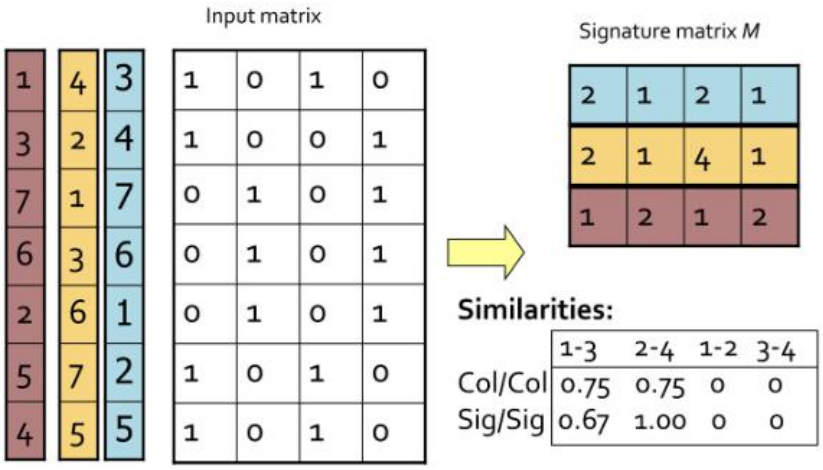

Fig 1

如上图所示，左边三个颜色的列向量分别代表三个随机置换，input matrix 在经过这三个随机置换后生成一个三行的 Signature matrix M，这个 M(r,c)的含义：input matrix 中的集合 c 经过第 r 个随机置换后的最小哈希就是 M(r,c)的元素值。

得到这么个 Signature matrix M 我们不就可以求 Jaccard 相似度了嘛？如图上右下角所示！理论上，当你的随机置换的数量**相当**大时，你利用 Signature matrix M 算出来的 Jaccard 相似度和原 input matrix 的相似度**相当**接近。这里因为我们只用了三个随机置换，所以算出来的 Jaccard 相似度和原 matrix 的相似度还是有那么一些误差的！（这话是我自己说的!）

以上内容就是为什么两个集合的最小哈希值相等的概率等于这两个集合的 Jaccard 相似度。

Min-Hashing 实际步骤：

如上图 Fig 1，我们通过一种简单易懂的方式求得了 Signature matrix M，因此知道了 Signature matrix M 的里面内容的含义，也知道了这个 Siganture matrix M 是怎么被拿来计算 Jaccard 相似度得。 但现实中，面对海量数据且特征特别多时，一个随机行置换是极耗事件的，更何况我们还要进行好多次置换。 我们难道也是用 Fig 1 的方法求 Signatrue matrix M？当然不是了。实际中用了一个巧妙地解决方法，下面我们来介绍一下：

假设我们要进行 n 次行打乱，为了模拟这个效果，我们选用了 n 个随机哈希函数 $\{g_1, g_2, g_3, \ldots, g_n\}$ 来处理特征矩阵时，分别计算打乱后的这 n 个矩阵的最小哈希值；

接下来的内容比较绕，比较烧脑，建议每句话多读几遍。

我们选用了 n 个随机哈希函数 $\{g_1, g_2, g_3, \ldots, g_n\}$，我们分别求每行的行编号对应随机哈希函数的哈希值，如下表的 g_1 g_2 所示，其中 x 代表行号，注意这里我们假设 n=2（方便举例和做表）：

x\集合+随机哈希函数g	S1	S2	S3	S4	g1=(x+1) mod 5	g2=(3*x+1) mod 5
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

注意我们上面特地取了这随机哈希函数 g 作为符号，是因为这里的随机哈希函数和上面讲解思想时的哈希不是一个东西。上面的 h 的作用是随机行置换并取该列（集合）第一个不为 0 的行作为哈希输出。这里 g 的作用是输入行数，输出其对应哈希值，本质应该是对行数的编号进行随机打乱输出。

我们再来记 SIG(i,c)表示签名矩阵中第 i 个哈希函数在第 c 列上的元素。SIG(i,c)开始时初始化为 Inf (无穷大)，我们给出当随机哈希函数数量 n=2 时的随机签名矩阵：

随机哈希函数g\集合	S1	S2	S3	S4
g1	Inf	Inf	Inf	Inf
g2	Inf	Inf	Inf	Inf

上面这两个表就是我们的原材料，让你有对 Min-hashing 有个直观了解，现在我们对其进行加工，我先给出伪算法，再来具体举例。

伪算法：（Yugang Yang 独家原创，看网上写的乱七八糟不如看小道写的。）

1. 我先叙述一下下面伪算法会用到的符号：

a. g_i 代表第 i 个随机哈希函数，其中 $i \in \{1,2,...,n\}$ 本例中我们选了 n=2

b. S[c][r] 代表第 c 个集合的第 r 行的值，其中 $c \in \{1,2,...,m\}$，m 是集合的数量。

$r \in \{1,2,...,R\}$,其中 R 代表这个特征矩阵中，集合内的元素数量，本例中我们选了 m=4 R=5。

c. SIG 代表签名矩阵， SIG[i][c]代表签名矩阵的第 i 个哈希函数在第 c 列上的元素，其中 $i \in \{1,2,...,n\}$ 而 $c \in \{1,2...,m\}$

2. 正式开始给出伪算法：

```
for(r = 1; r <= R; r++)//外层 for 循环遍历特征矩阵每一行
{
    for(c = 1; c <= m; c++) //内层 for 循环遍历特征矩阵在第 r 行的每一列
    {
        if(S[r][c] == 1) //在特征矩阵的第 r 行的 c 列上发现元素值为 1
        {
            for(i = 1; i < = n; i++) //在签名矩阵的第 c 列遍历每一行
            {
                SIG[i][c] = min( SIG[i][c] , g_i(r) );
            }
        }
```

```
    }
  }
```

根据这个伪算法我们可以得到签名矩阵：

3. 现在我想来探讨一下为什么这么做可以生成签名矩阵：（Yugang Yang 独家原创）

我们现在就单纯的考虑一个集合和一个哈希函数，其特征矩阵和签名矩阵如下：

特征矩阵：

x\集合+随机哈希函数g	S1	S2	S3	S4	g1=(x+1) mod 5	g2=(3*x+1) mod 5
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

签名矩阵：

随机哈希函数g\集合	S1	S2	S3	S4
g1	Inf	Inf	Inf	Inf
g2	Inf	Inf	Inf	Inf

这个过程其实本质上和从一维数组中选最小值没有什么区别 比如我们一个一维数组 array

然后就：

```
int Min
for(int i = 0; i < arrNum; i++)
  if(Min > array[i])
    Min = array[i]
```

你好好感受一下，实质上就是这个意思。

那个 if(S[r][c] == 1) 条件就跟过滤器一样，筛选出行 0 和行 3，然后行 0 和行 3 分别代表打乱后的行 1 和行 0，然后行 0 比行 1 小，且它的特征矩阵中的值为 1，所以是最小哈希！

就是这么个简单的过程！

reference:

https://www.cnblogs.com/maybe2030/p/4953039.html 图画的好

https://www.cnblogs.com/shipengzhi/articles/2826209.html

https://blog.csdn.net/liujan511536/article/details/47729721

https://www.cnblogs.com/sddai/p/6110704.html