

part2 Locality Sensitive Hashing

局部敏感哈希是一种计算向量相似度的近似算法。

在很多场景中，我们需要计算大量高维向量的相似度。

- 在文档主题匹配中，对于文档的topic vector进行相似性计算
- 协同过滤中，相似的用户购买相似的物品
- 推荐和搜索场景

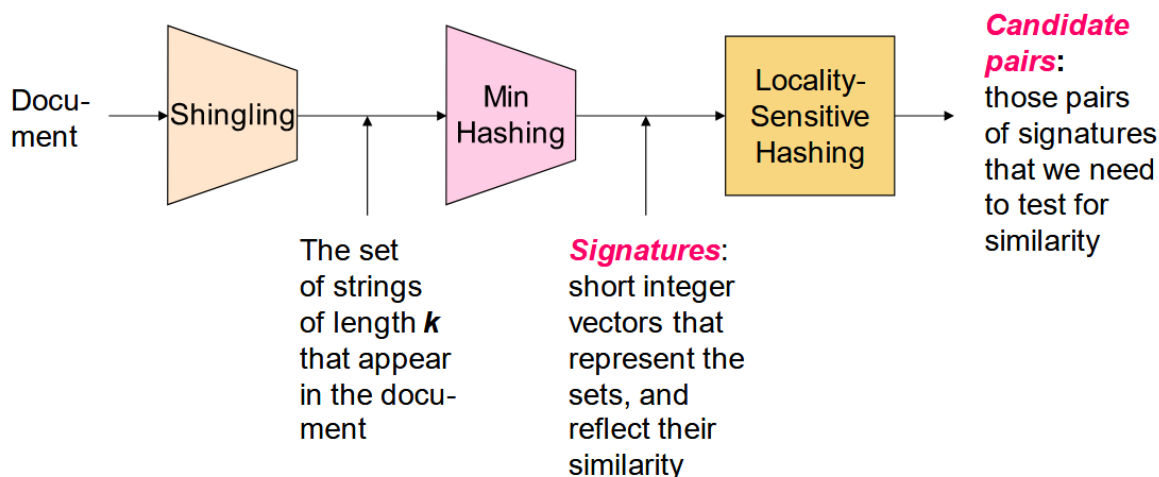
所以我们需要解决的问题为：给出高维向量 x_i ，定义距离函数 $d(x_1, x_2)$ ，如何找出使得 $d(x_i, x_j) \leq s$ 的向量组合 (x_i, x_j) 。暴力枚举的算法复杂度为 $O(N^2)$ ，如何将复杂度缩短到 $O(N)$ ？

Motivation for LSH

- Suppose we need to find near-duplicate documents among $N = 1$ million documents
 - Naïvely, we would have to compute **pairwise similarities** for **every pair of docs**
 - $N(N - 1)/2 \approx 5 \cdot 10^{11}$ comparisons
 - At 10^5 secs/day and 10^6 comparisons/sec, it would take **5 days**
 - For $N = 10$ million, it takes more than a year...
- Similarly, we have a dataset of 10B documents, quickly find the document that is most similar to query document q .

假设我们需要寻找相似的documents among $N = 1$ million。整个流程可以被分为三部分。

- Shingling : Converts a document into a set representation
- Min-Hashing: Convert large sets to short signatures, while preserving similarity
- Locality-Sensitive Hashing: Focus on pairs of signatures likely to be from similar documents



Shingling

Step 1: **Shingling**: Converts a document into a set

- A ***k*-shingle** (or ***k*-gram**) for a document is a sequence of ***k* tokens** that appears in the doc
 - Tokens can be **characters**, **words** or something else, depending on the application
 - Assume tokens = characters for examples
- To **compress long shingles**, we can **hash** them to (say) 4 bytes
- **Represent a document by the set of hash values of its *k*-shingles**

将长文档转化为 *a set of hash values*, 例如

- **Example: $k=2$** ; document $D_1 = \text{abcaab}$
Set of 2-shingles: $S(D_1) = \{\text{ab, bc, ca}\}$
Hash the shingles: $h(D_1) = \{1, 5, 7\}$
- **$k = 8, 9, \text{ or } 10$ is often used in practice**

相似的文档会有许多相同的shingling, 同时将文档切分后, 更改某个单词只会影响k-shingling中的其余k-1个单词。

进行Shingling后的document D_i 可以被表示为 $C_i = S(D_i)$, 即k-shingling的集合。

Jaccard系数是常见的衡量两个向量 (或集合) 相似度的度量:

$$\text{sim}(D_1, D_2) = |C_1 \cup C_2| / |C_1 \cap C_2|$$

Jaccard distance: $d(C_1, C_2) = 1 - |C_1 \cup C_2| / |C_1 \cap C_2|$

From Sets to Boolean Matrices

Encode sets using 0/1 (bit, Boolean) vectors

- **Rows** = elements (shingles)
- **Columns** = sets (documents)
 - 1 in row e and column s if and only if e is a member of s
 - Column similarity is the Jaccard similarity of the corresponding sets (rows with value 1)
 - **Typical matrix is sparse!**
- **Each document is a column:**
 - **Example:** $\text{sim}(C_1, C_2) = ?$
 - Size of intersection = 3; size of union = 6, Jaccard similarity (not distance) = $3/6$
 - $d(C_1, C_2) = 1 - (\text{Jaccard similarity}) = 3/6$

		Documents			
Shingles	1	1	1	0	
	1	1	0	1	
	0	1	0	1	
	0	0	0	1	
	1	0	0	1	
	1	1	1	0	
	1	0	1	0	

We don't really construct the matrix; just imagine it exists

对于进行Shingling后的文档向量，我们可以使用01向量来表示。

Min-Hashing

- **Key idea:** “hash” each column C to a small **signature** $h(C)$, such that:
 - $\text{sim}(C_1, C_2)$ is the same as the “similarity” of signatures $h(C_1)$ and $h(C_2)$
- **Goal: Find a hash function $h(\cdot)$ such that:**
 - If $\text{sim}(C_1, C_2)$ is high, then with high prob. $h(C_1) = h(C_2)$
 - If $\text{sim}(C_1, C_2)$ is low, then with high prob. $h(C_1) \neq h(C_2)$
- **Idea: Hash docs into buckets. Expect that “most” pairs of near duplicate docs hash into the same bucket!**

Min-Hashing的思想是寻找到一种将高维稀疏的shingling向量映射到低维向量的方法，同时保证映射前后的向量相似度尽可能的一致。相似的doc向量在映射之后也应该相同的bucket。

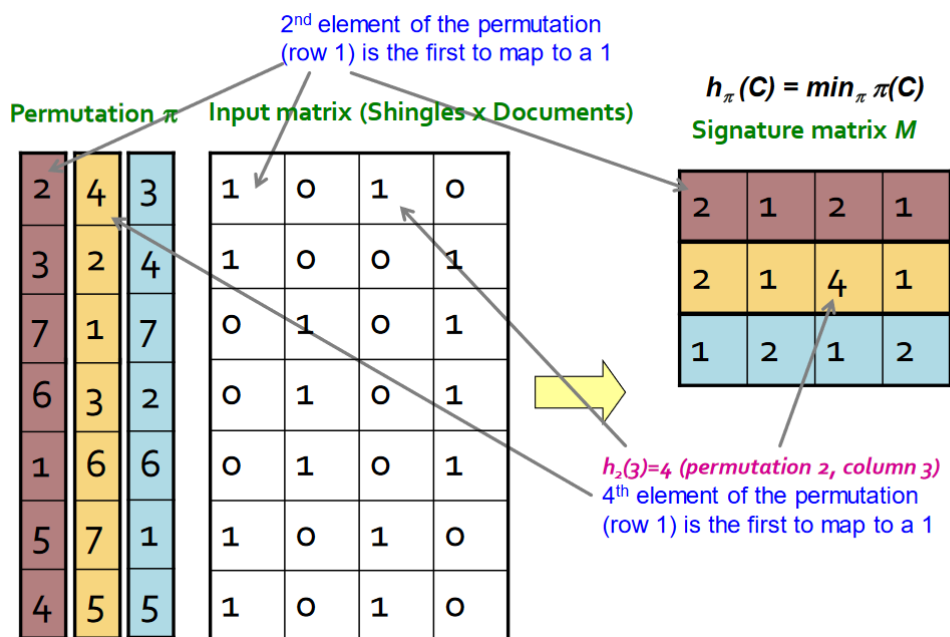
- 如果 $\text{sim}(C_1, C_2)$ 很高，那么映射后的hashing函数结果 $h(C_1) = h(C_2)$ 的概率也应该很高
- 如果 $\text{sim}(C_1, C_2)$ 很低，那么映射后的hashing函数结果 $h(C_1) \neq h(C_2)$ 的概率也应该很高

因此对于向量相似性的度量会影响到Hashing函数的选择，并且并不是所有的相似性度量都会有与支配的Hashing函数。对于Jaccard相似性而言，与之对应的Hashing函数是Min-Hashing函数。

Min-Hashing大致流程如下：

- 将Shingling后形成的01矩阵使用随机排列 π 替换
- 随机排列 π 代表一串随机顺序的正整数数列，长度与Shingling矩阵的行数相同。
- 对于矩阵的每一列 C 和一个随机排列数列 π ， $h_{\pi}(C) = \min_{\pi}(\pi(C))$ ，即Hashing的结果是这个随机排列 π 对应的最小的 C 中为1的数。

一个随机排列 π 对于 C 会形成一个整数，因此Min-Hashing后的signature向量维度就等于我们选择的 π 个数。



上图给出了三个随机排列 π ，长度与矩阵的行数相同。我们根据矩阵列向量为1的行，找到对应 π 的整数值，找到其中最小的为Hashing的结果。上图有三个排列，所以得到的signature结果为三维。

Min-Hashing的性质

- $Pr[h_{\pi}(C_1) = h_{\pi}(C_2)] = sim(C_1, C_2)$ 即对于同一个随机排列 π ，Min-Hashing结果相同的概率等于 C_1 C_2 的相似度。

The Min-Hash Property (1)

- Choose a random permutation π
- Claim:** $\Pr[h_{\pi}(C_1) = h_{\pi}(C_2)] = \text{sim}(C_1, C_2)$
- Why?**
 - Let X be a doc (set of shingles), $z \in X$ is a shingle
 - Then:** $\Pr[\pi(z) = \min(\pi(X))] = 1/|X|$
 - It is equally likely that any $z \in X$ is mapped to the *min* element
 - Now,** let y be s.t. $\pi(y) = \min(\pi(C_1 \cup C_2))$
 - Then either:** $\pi(y) = \min(\pi(C_1))$ if $y \in C_1$, **or** $\pi(y) = \min(\pi(C_2))$ if $y \in C_2$
 - One of the two cols should have 1 at position y
 - So the prob. that **both** are true is the prob. $y \in C_1 \cap C_2$
 - $\Pr[\min(\pi(C_1)) = \min(\pi(C_2))] = |C_1 \cap C_2| / |C_1 \cup C_2| = \text{sim}(C_1, C_2)$

0	0
0	0
1	1
0	0
0	1
1	0

Permutation π Input matrix (Shingles x Documents)

2	4	3
3	2	4
7	1	7
6	3	2
1	6	6
5	7	1
4	5	5

1	0	1	0
1	0	0	1
0	1	0	1
0	1	0	1
0	1	0	1
1	0	1	0
1	0	1	0

Signature matrix M

2	1	2	1
2	1	4	1
1	2	1	2



Similarities:

	1-3	2-4	1-2	3-4
Col/Col	0.75	0.75	0	0
Sig/Sig	0.67	1.00	0	0

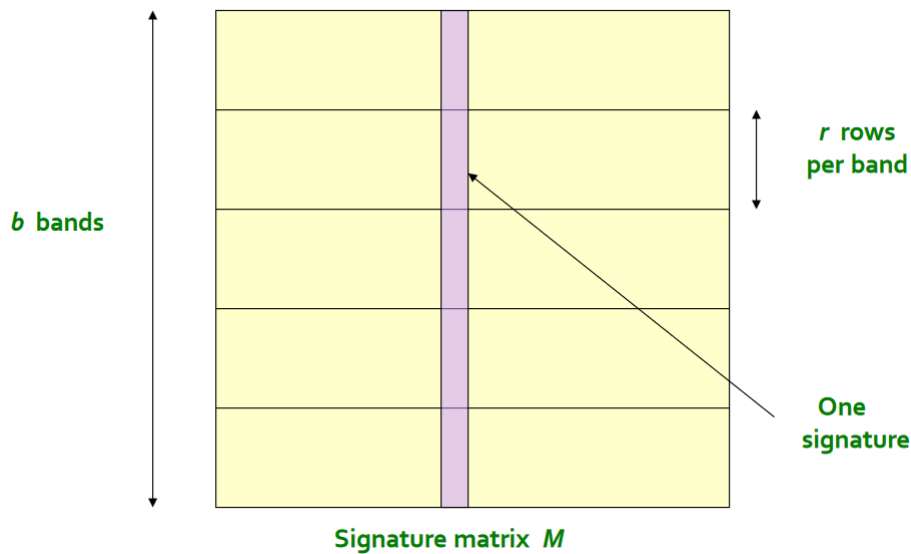
通过Min-Hashing我们可以在降低了doc表示向量的同时保持了向量之间的相似度。Signature向量的维度越接近Shingling向量，向量相似度就会保存的越好。

LSH

通过Shingling和Min-Hashing我们完成了将doc转化为01向量并且进行了降维的过程，但是如果有N篇文档且N很大时，计算复杂度仍然为 $O(N^2)$ 。LSH主要通过筛选出部分candidate向量来减小 N^2 的复杂度。

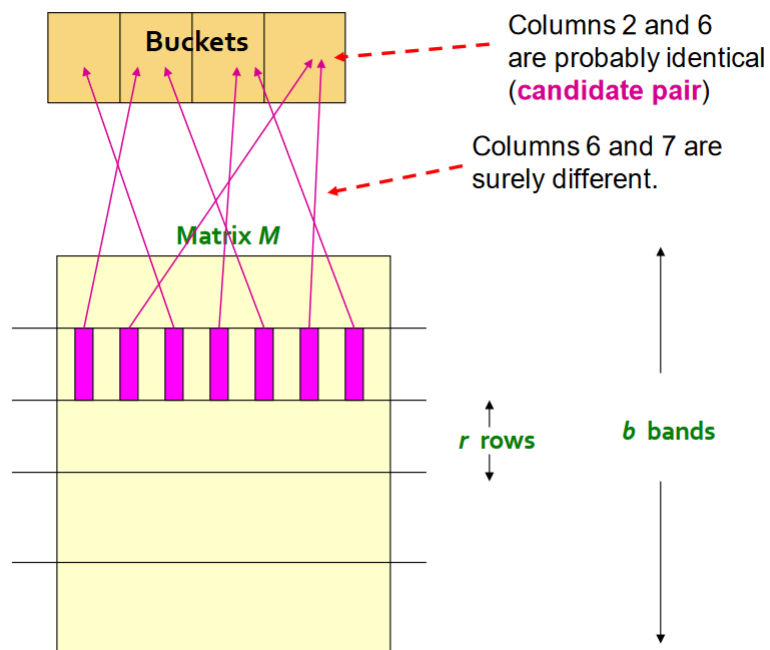
Partition M into b Bands

2	1	4	1
1	2	1	2
2	1	2	1



M 矩阵的每一列就是Min-Hashing后得到的signature向量。每一个向量在图中被分为了 b 段（每一列为一个向量），每一段有 r 行MinHash值。在任意一个band中分到了同一个桶内，就成为候选相似用户（拥有较大可能相似）。

我们设两个向量的相似度为 t ，则其任意一个band所有行相同的概率为 t^r ，至少有一行不同的概率为 $(1 - t^r)$ 则所有band都不同的概率为 $(1 - t^r)^b$ ，至少有一个band相同的概率为 $1 - (1 - t^r)^b$ 。



If C_1, C_2 are 80% Similar

1	2	1	2
2	1	2	1

- Find pairs of $\geq s=0.8$ similarity, set $b=20, r=5$
- Assume: $\text{sim}(C_1, C_2) = 0.8$
 - Since $\text{sim}(C_1, C_2) \geq s$, we want C_1, C_2 to be a **candidate pair**: We want them to hash to at **least 1 common bucket** (at least one band is identical)
- Probability C_1, C_2 identical in one particular band: $(0.8)^5 = 0.328$
- Probability C_1, C_2 are **not** similar in all 20 bands: $(1-0.328)^{20} = 0.00035$
 - i.e., about 1/3000th of the 80%-similar column pairs are **false negatives** (we miss them)
 - We would find **99.965%** pairs of truly similar documents

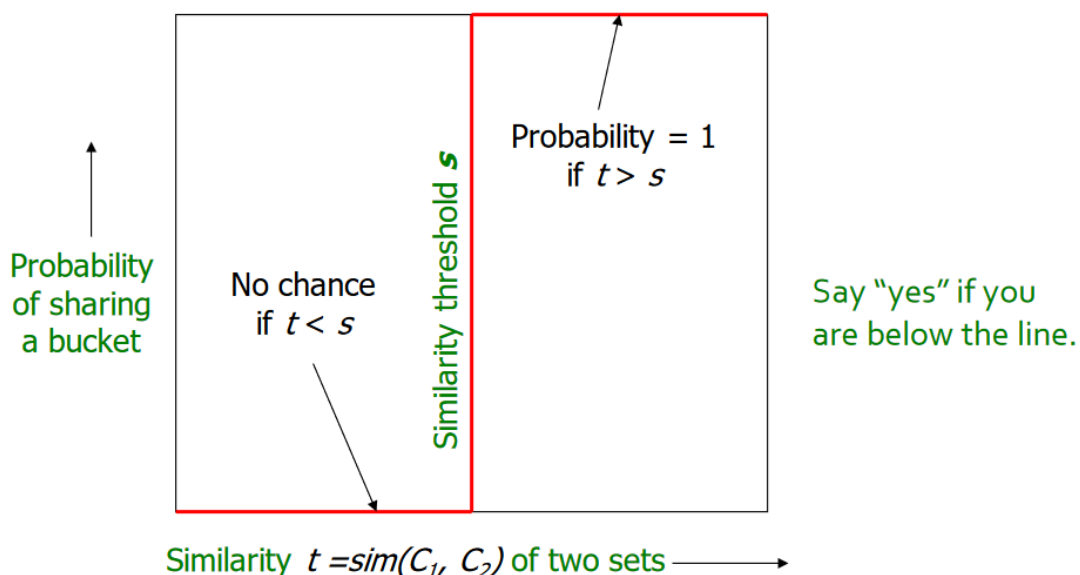
根据上图的假设，我们假设 C_1, C_2 的相似度为0.8，即大概率是真正相似的向量，signature向量维度为100，分为20个band，band内长度为5。经过计算我们可以得到我们根据LSH算法有99.9%的概率计算为candidate向量。

If C_1, C_2 are 30% Similar

2	1	4	1
1	2	1	2
2	1	2	1

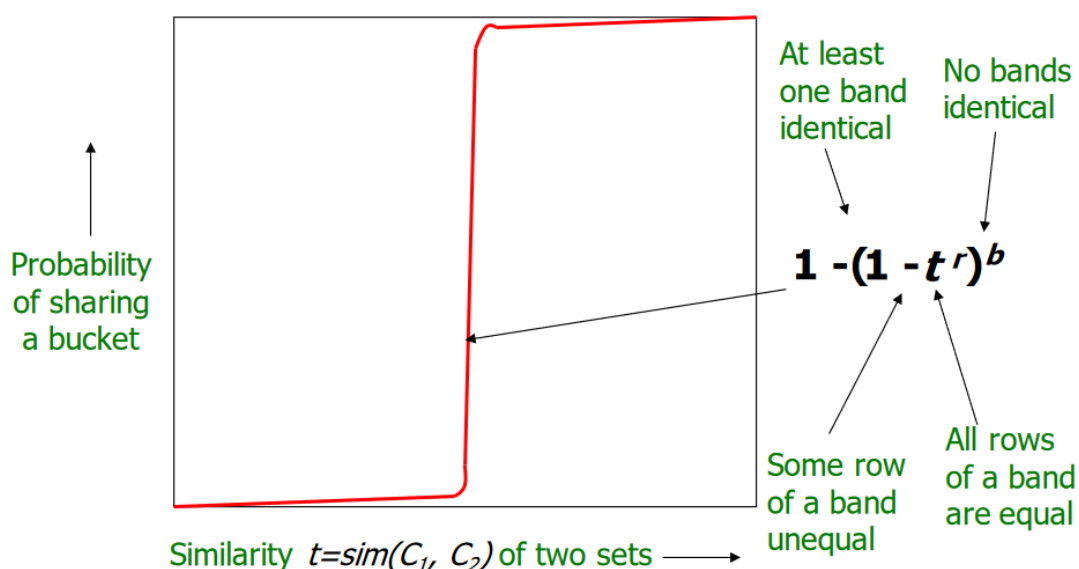
- Find pairs of $\geq s=0.8$ similarity, set $b=20, r=5$
- Assume: $\text{sim}(C_1, C_2) = 0.3$
 - Since $\text{sim}(C_1, C_2) < s$ we want C_1, C_2 to hash to **NO common buckets** (all bands should be different)
- Probability C_1, C_2 identical in one particular band: $(0.3)^5 = 0.00243$
- Probability C_1, C_2 identical in at least 1 of 20 bands: $1 - (1 - 0.00243)^{20} = 0.0474$
 - In other words, approximately 4.74% pairs of docs with similarity 0.3 end up **becoming candidate pairs**
 - They are **false positives** since we will have to examine them (they are candidate pairs) but then it will turn out their similarity is below threshold s

根据上图的假设，我们假设 C_1, C_2 ，即不是真正相似的向量，我们也可以计算得到他们有4.74%被计算为candidate向量。



上图是我们期望构造的判别函数，对于不同的 $\text{sim}(C_1, C_2)$ ，给定某个阈值后，小于阈值的向量组合不会被share到一个bucket中，大于阈值的向量组合会100%被share到一个bucket中。

What b Bands of r Rows Gives You



根据上文的计算，LSH算法构造的判别函数如上图所示，通过选择不同的超参数 r 和 b 。我们可以控制LSH的函数图像尽可能逼近理想的曲线。所以现在的问题转化为如何调参 b 和 r 来调整曲线的形状？

Summary: 3 Steps

- **Shingling:** Convert documents to set representation
 - We used hashing to assign each shingle an ID
- **Min-Hashing:** Convert large sets to short signatures, while preserving similarity
 - We used **similarity preserving hashing** to generate signatures with property $\Pr[h_{\pi}(C_1) = h_{\pi}(C_2)] = \text{sim}(C_1, C_2)$
 - We used hashing to get around generating random permutations
- **Locality-Sensitive Hashing:** Focus on pairs of signatures likely to be from similar documents
 - We used hashing to find **candidate pairs** of similarity $\geq s$

我们现在完成了Shingling,Min-Hashing和LSH的流程。关于LSH的原理和调参在下一节中阐述。