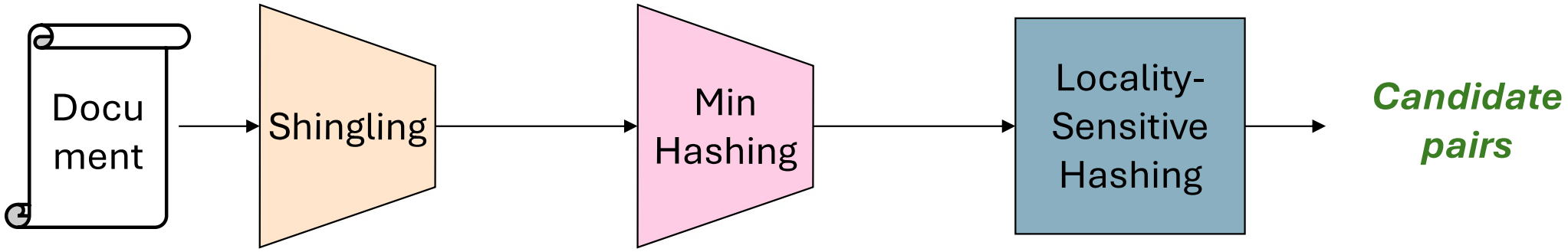


Locality Sensitive Hashing

Focus on pairs of signatures likely to be from similar documents

The key idea

Generate a **sketch** for every object that:

1) is ***much shorter*** than its # features (i.e. **d)**

2) transforms ***similarity*** between two feature vectors into ***equality*** *of their shorter sketches*.

- ✓ It is ***randomized***, correct ***with high probability***

  (good if this is the only way to approach the problem !!)

- ✓ It guarantees ***local access*** to data, which is good for speed in disk/distributed setting

The hamming case

- Consider vectors p, q of **d** *binary* features

- Hamming distance

  D(p,q)= #bits where *p* and *q* differ

- Define hash function *h* by choosing a set *I* of *r* random coordinates

  h(p) = projection of vector *p* on *I's* coordinates

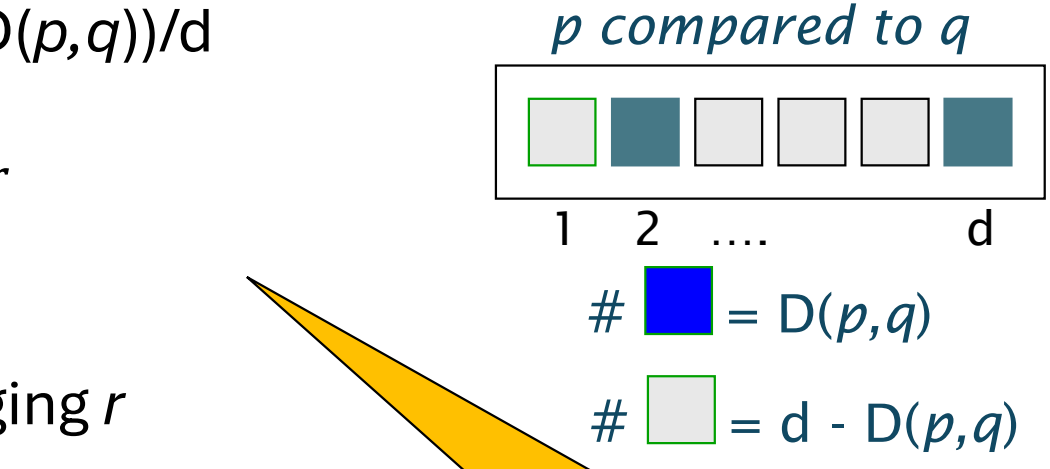
Example: If r=2, *assume I={1,4} then it is h(p=**0**10**1**1) = 01*

A key property

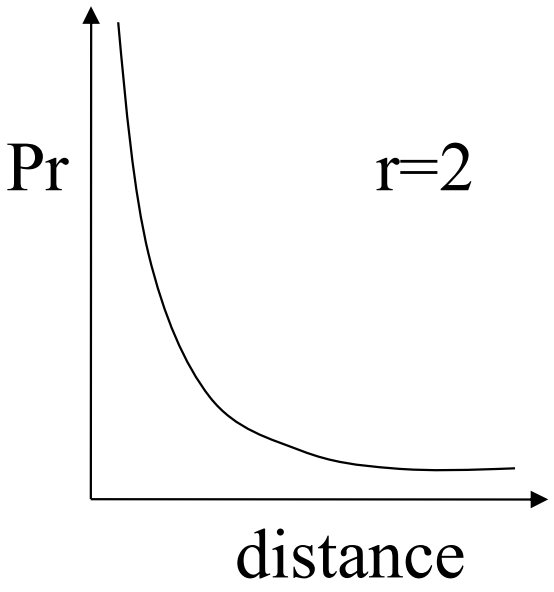
Pr[*picking x:p[x]=q[x]*] = # /d = (d - D(*p*,*q*))/d

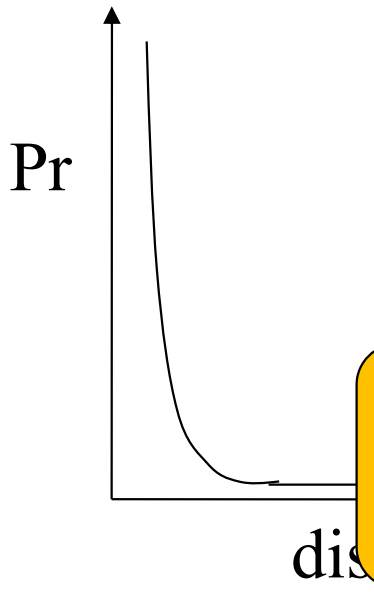
$$\Pr[\, h(p) = h(q)] = (1 - \frac{D(p,q)}{d})^r$$

We can vary the probability by changing *r*



= s^r
where s is the similarity between p and q





Larger r
Fewer False Positive
What about False Negatives?

Reiterate b times (calle band)

Larger b
Fewer False Negatives

1) Repeat **b times** the **r-projections** $h_i(p)$
2) We set $g(p) = < h_1(p), h_2(p), \ldots, h_b(p)>$
3) Declare «p matches q» if **at least** one $h_i(p)=h_i(q)$

Sketch(p)

Example:

*Let us set **r=2, b=3**, assume p = 01**0**01 and q = 01**1**01*

- *I1 = {3,4}, we have $h_1(p) = 00$* and *$h_1(q)=10$*
- *I2 = {1,3}, we have $h_2(p) = 00$* and *$h_2(q)=01$*
- *I3 = {1,5}, we have $h_3(p) = 01$* and *$h_3(q)=01$*

p and q declared to match !!

Measuring the error probability

$$\Pr[\, h_i(p) = h_i(q)] = (1 - \frac{D(p,q)}{d})^r = s^r$$

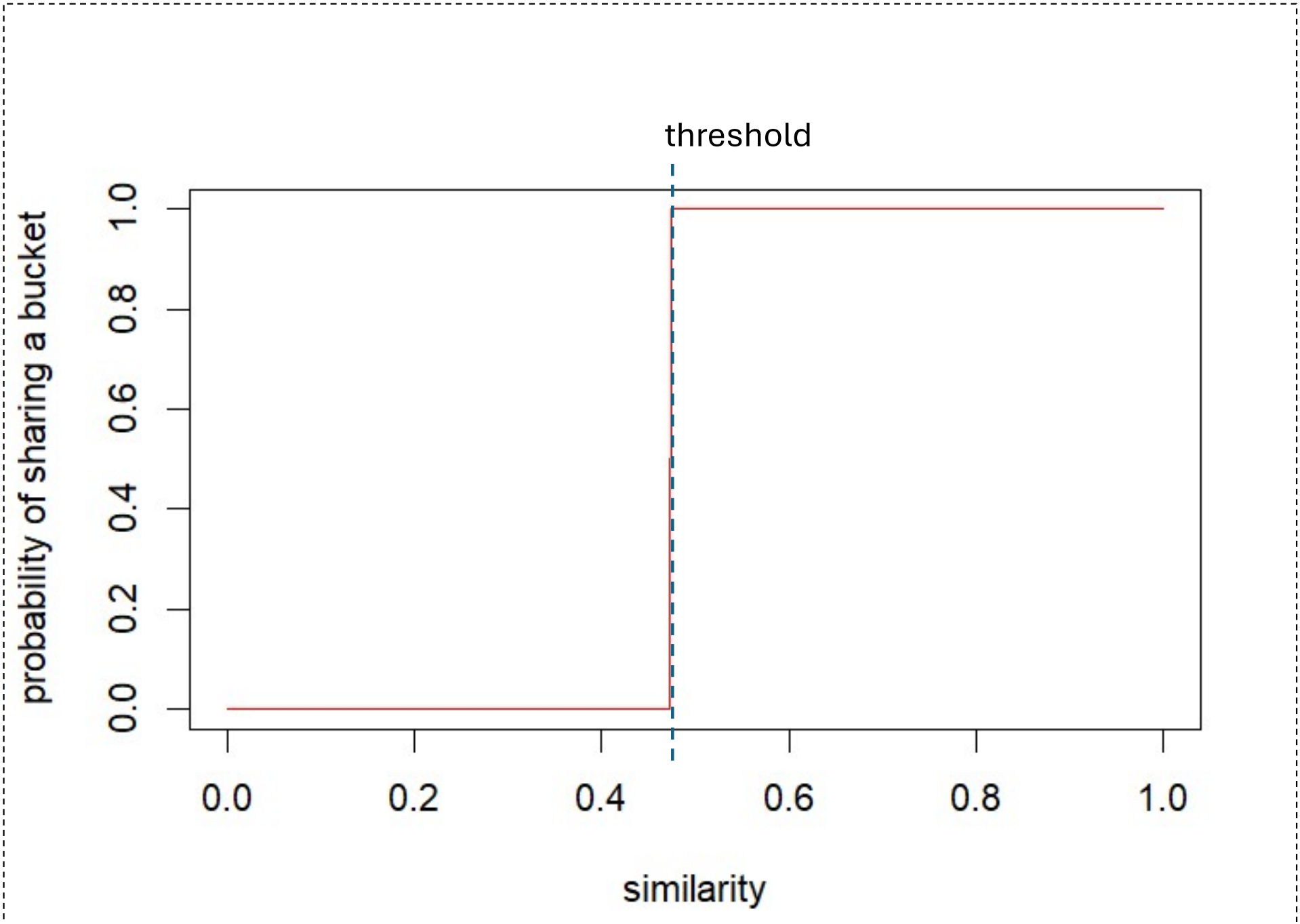
The g() consists of b independent hashes h_i

Pr[p not-similar q] = Pr[$h_i(p) \neq h_i(q)$, $\forall i=1, \ldots, b$]

= (Pr[$h_i(p) \neq h_i(q)$])^b

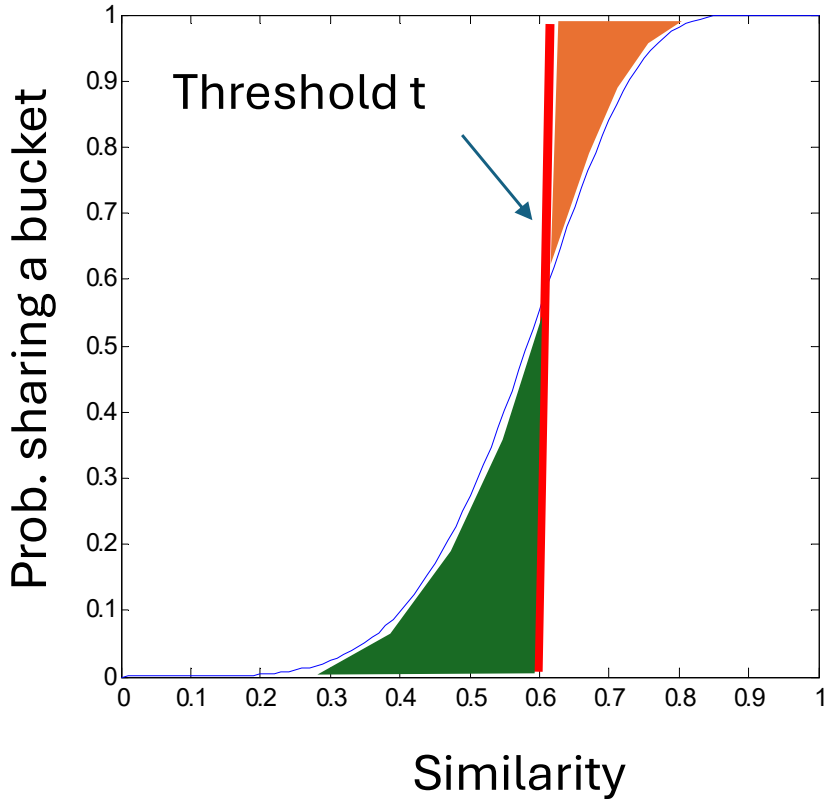
= (1- Pr[$h_i(p) = h_i(q)$])^b

The S-curve $1 - (1 - s^r)^b$



Picking r and *B*: The S-curve

- **Picking *r* and *b* to get the best S-curve**
  - 50 hash-functions (r=5, b=10)

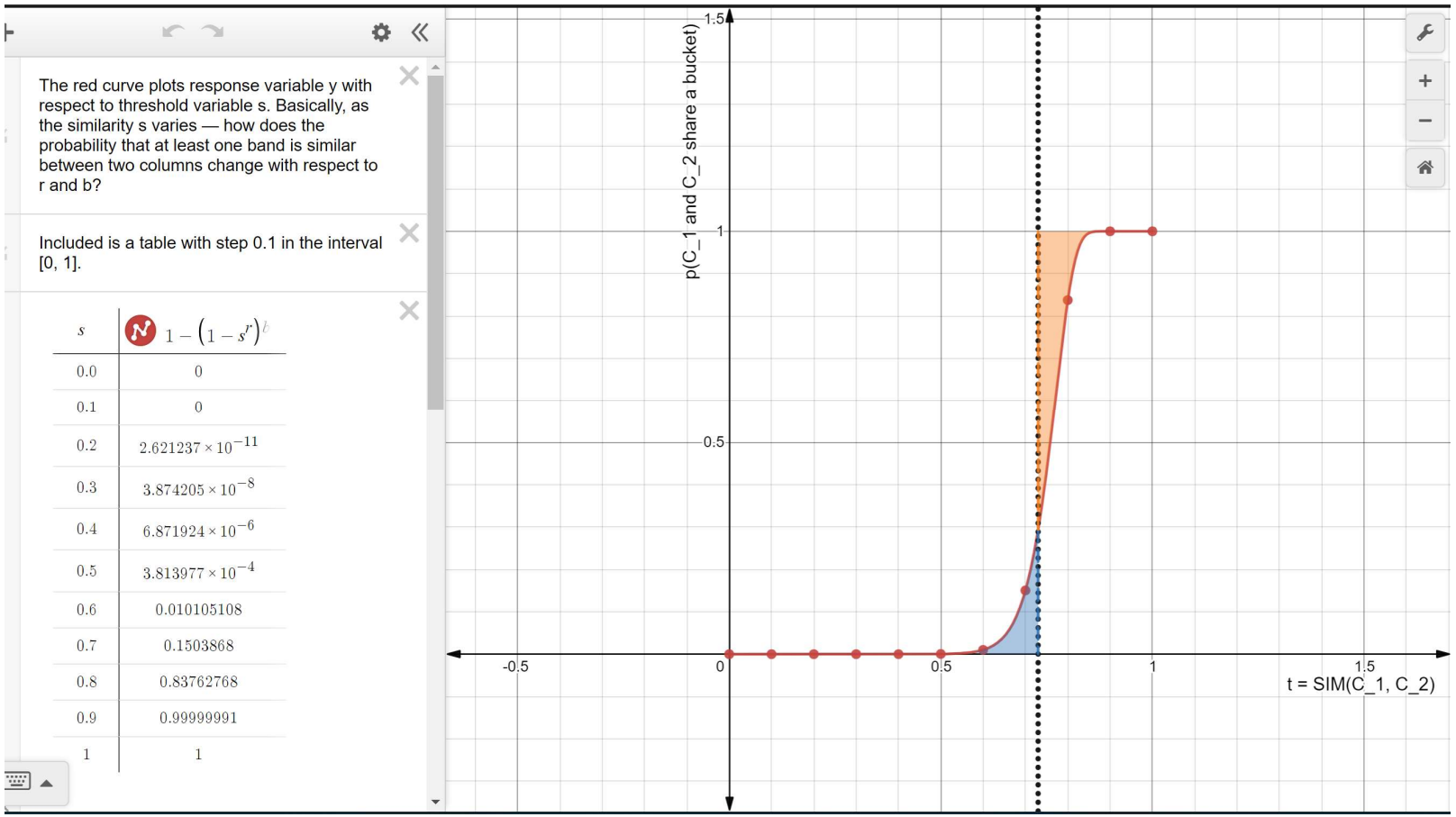


Red area: False Negative rate
Pairs with sim>t that will be never considered

Green area: False Positive rate
Pairs with sim<t, we can Discard these pairs once we Compute the exact distance

Demo

- https://www.desmos.com/calculator/lzzvfjiujn



Example

- Assume the following case:
  - Suppose 100,000 columns of M (100k docs)
  - Signatures of length 100, stored as integers (rows)
  - Therefore, signatures take 40MB
  - Goal: Find pairs of documents that are at least s = 0.8 similar
  - Choose b = 20 bands of r = 5 integers/band

Two columns highly similar

- Find pairs of $\geq$ s=0.8 similarity, let's set b=20, r=5
- Assume: sim(C1, C2) = 0.8
- Since sim(C1, C2) $\geq$ s, we want C1, C2 to be a candidate pair: We want them to hash to at least 1 common bucket (at least one band is identical):
- Prob. C1, C2 identical in one particular band: (0.8) $(0.8)^5$ = 0.328
- So, prob. C1, C2 are not similar in all 20 bands: $(1 - 0.328)^{20}$ = 0.00035
- That is, 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

Two column far away

- Find pairs of $\geq$ s=0.8 similarity, let's set b=20, r=5
- Assume: sim(C1, C2) = 0.3
- Since sim(C1, C2) $<$ s, we want C1, C2 to be a candidate pair: We want them to be not a candidate pair (all bands are differt):
- Prob. C1, C2 identical in one particular band: $(0.3)^5$ = 0.00243
- So, prob. C1, C2 are identical in at least one band is: 1- $(1 - 0.00243)^{20}$ = 0.0474
- We have that 4.75% pair of documents with similarity 0.3 will appear as candidate pairs.
- These are false positive.

Example: $b = 20$; $r = 5$

- Similarity threshold **s**
- Prob. that at least 1 of r proj of the sketch is identical:

s	**$1-(1-s^r)^b$**
.2	.006
.3	.047
.4	.186
.5	.470
.6	.802
.7	.975
.8	.9996

The (off-line) algorithm

- For every feature vector p, compute

$$g(p) = < h_1(p), h_2(p), \ldots, h_b(p)>$$

Sketch(p)

- For every i=1, 2, .., b, create the clustering C_i by putting in the same group vectors *p* and *q* iff $h_i(p) = h_i(q)$

Sort

- Create an undirected graph such that nodes *p* and *q* are **linked** iff their sketches are in the same cluster of C_i for some iteration *i*
- Compute the **connected components** because they provide groups of similar vectors

Notebook

Tutorial su MinHashing e LSH

https://colab.research.google.com/drive/1hjklFJM-1PMLNSl2MOqe7ZnLxkOLLc-A?usp=sharing

Next step: LSH Tuning

- Tune M, b, r to get almost all pairs with similar signatures, but eliminate most pairs that do not have similar signatures