CS3319 Foundations of Data Science

4.Locality Sensitive Hashing

Jiaxin Ding
John Hopcroft Center

Text Similarity

- Similarity check for a paper with all the published papers.


Running title • August 2012 • Vol. XXI, No. 1

Article Title

John Smith*

University of California
john@smith.com

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

I. Introduction

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

II. Methods

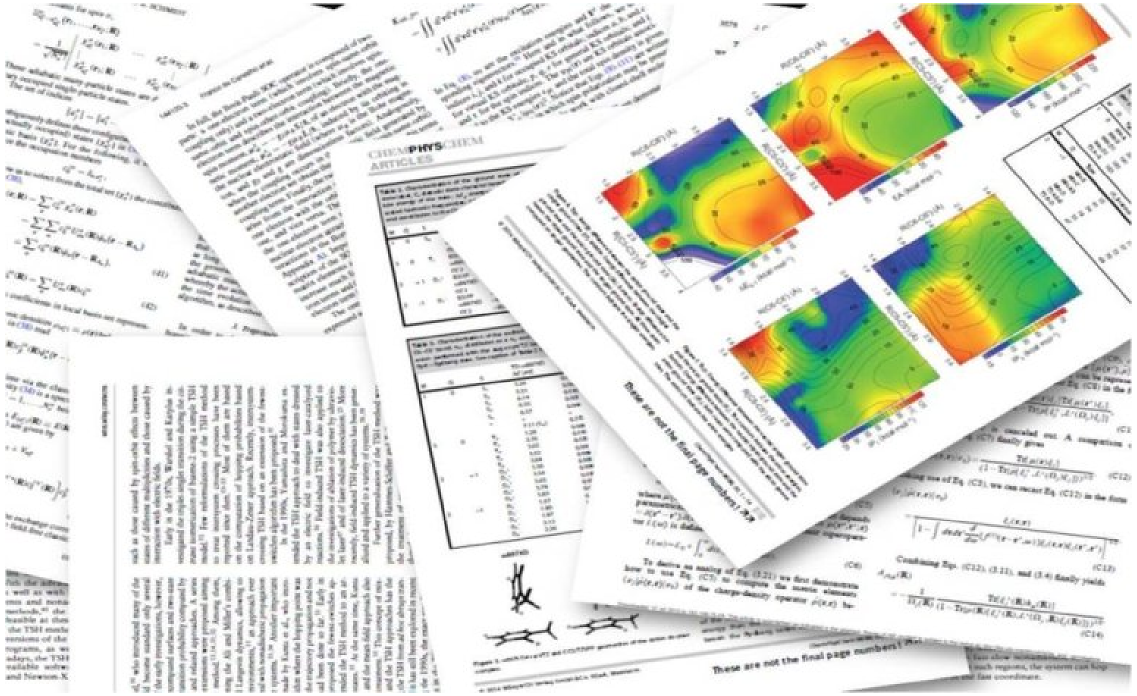
Maecenas sed ultricies felis. Sed imperdiet dictum arcu a egestas.

- Donec dolor arcu, rutrum id molestie in, viverra sed diam
- Curabitur feugiat
- turpis sed auctor facilisis
- arcu eros accumsan lorem, at posuere mi diam sit amet tortor
- Fusce fermentum, mi sit amet euismod rutrum
- sem lorem molestie diam, iaculis aliquet sapien tortor non nisi
- Pellentesque bibendum pretium aliquet

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultri-

*A thank you or further information

1



Task: Finding Similar Documents

- **Goal**: Given a large number ($\boldsymbol{N}$ in the millions or billions) of documents, find "near duplicate" pairs

- Challenges:
  - How to define the **similarity**?
    - Many small pieces of one document can appear **out of order** in another.
  - How to compute **efficiently**?
    - Documents are so **large** or so many that they cannot fit in main memory
    - **Too many** documents to compare **all pairs**. E.g. 1 million documents, we have 10^{12} pairs, if we compare 10^6 per second, it takes about 10 days.

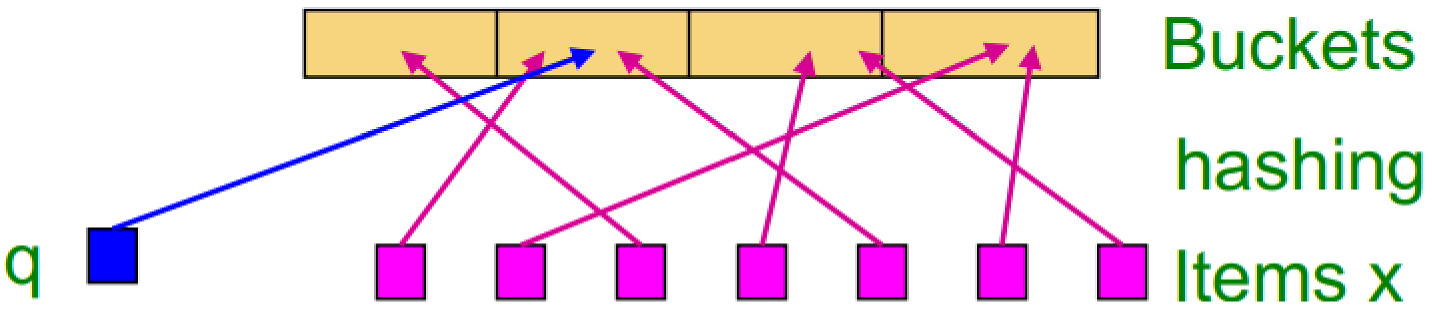
Problem Definition

- **Given**: High dimensional data points (e.g. Bag of Words) $x_1, x_2, \ldots$
- A **distance function** $d(x_1, x_2)$

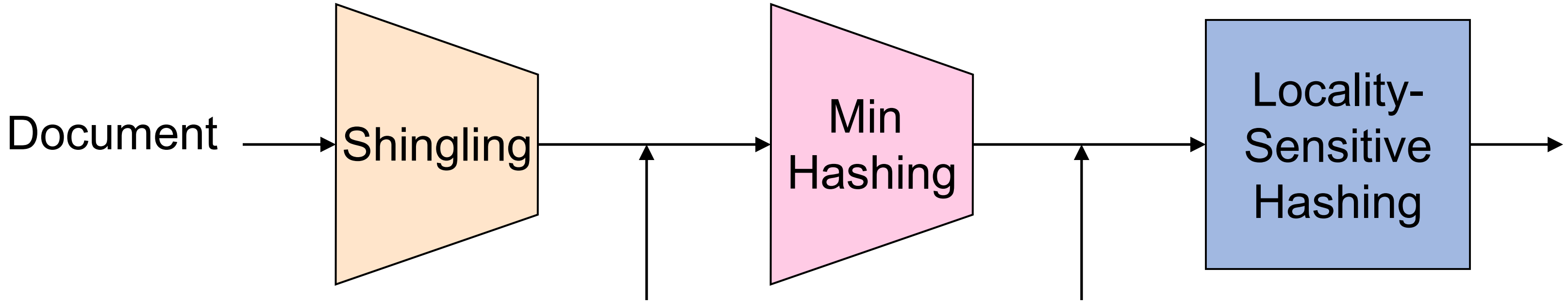
- **Goal:** Find all pairs of data points (x_i, x_j) that are within some distance threshold $d(x_i, x_j) \leq s$
- Naïve solution would take $O(N^2)$
  - where N is the number of data points
- This can be done in $O(N)$!

Key Idea

- **Hashing**
  - Throw items into buckets using several different **hash functions**.
  - Examine only those pairs of items that **share a bucket** for at least one of these hashings.



The Big Picture: 3 Steps for Similar Documents



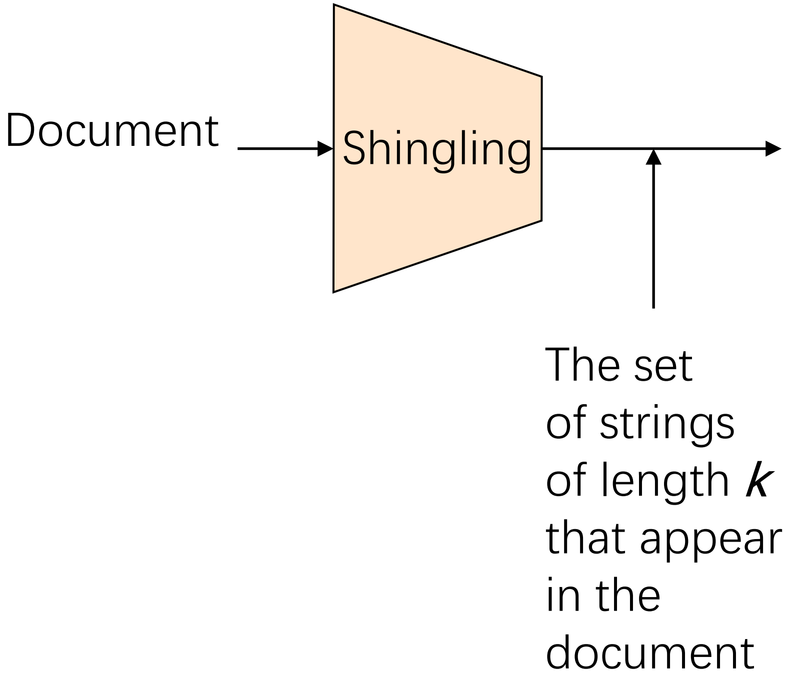
Candidate pairs: those pairs of signatures that we need to test for similarity

The set of strings of length ***k*** that appear in the document

Signatures: short integer vectors that represent the sets, and reflect their similarity

Shingling

- Step 1: Shingling: Convert documents to sets



Documents as High Dimensional Data

- Step 1: **Shingling**: Convert documents to sets

- Simple approaches:
  - Document = set of words appearing in document
  - Document = set of "important" words

- Need to account for **ordering** of words!
- A different way: Shingles!

Define: Shingles

- A **k-shingle** (or k-gram) for a document is a sequence of k tokens that appears in the document
  - Tokens can be characters, or words, depending on the application
- Represent a document by the set of its k-shingles

- Example: $k = 2$; document $D_1 = abcab$
  Set of 2-shingles: $S(D_1) = \{ab, bc, ca\}$

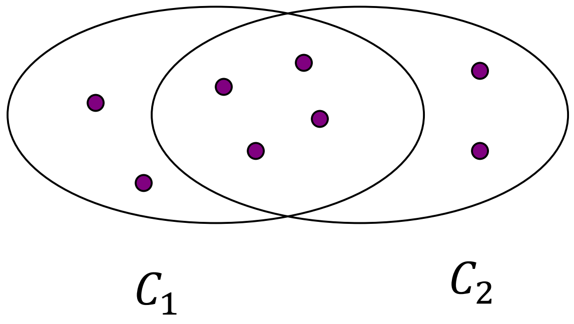
Similarity Metric for Shingles

- Document D_1 is a set of its k-shingles $C_1 = S(D_1)$
- A natural similarity measure is the **Jaccard similarity**:

$$sim(D_1, D_2) \; = \; |C_1 \cap C_2|/|C_1 \cup C_2|$$

- **Jaccard distance**:

$$\mathrm{d}(D_1, D_2) = 1 - sim(D_1, D_2)$$



$E.g.\ |C_1 \cup C_2| = 8$

$|C_1 \cap C_2| = 4$

$sim(D_1, D_2) = 0.5$

Set Representation

- Encode sets with 0/1 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
  - Typical matrix is sparse!

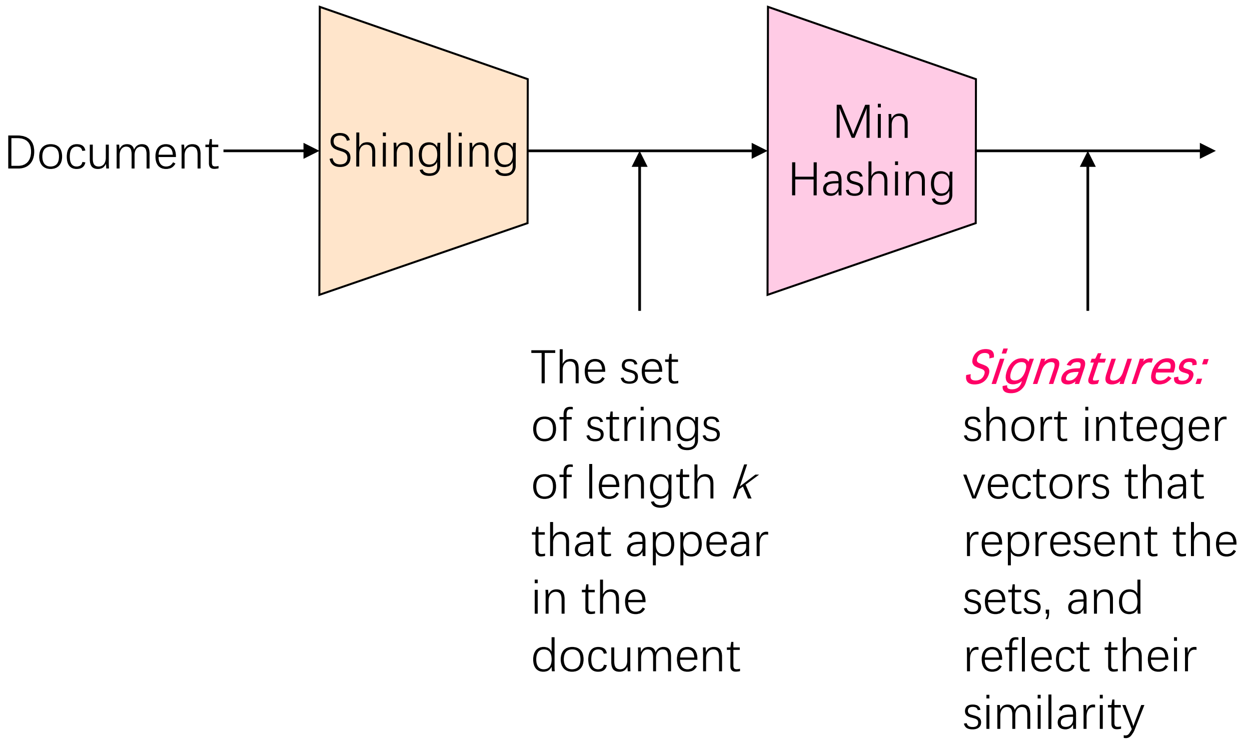
Documents

Shingles

	text1	text2		
ab	1	1	1	0
ac	1	1	0	1
	0	1	0	1
	0	0	0	1
	1	0	0	1
	1	1	1	0
	1	0	1	0

MinHashing

- Step 2: Min-hashing: Convert large sets to short signatures, while preserving similarity



Signatures

- Key idea: "hash" each column C to a small **signature** $h(C)$, such that:
  (1) $h(C)$ is small enough
  (2) $sim(C1, C2)$ is the same as the "similarity" of signatures $h(C1)$ and $h(C2)$

- **Goal**: Find a hash function h(·) such that:
  - If $sim(C1, C2)$ is high, then with high prob. $h(C1) = h(C2)$
  - If $sim(C1, C2)$ is low, then with high prob. $h(C1) \neq h(C2)$

- Hash function for the Jaccard similarity: **Min-Hashing**

Min-Hashing

2	1	0	1	0
3	1	0	0	1
7	0	1	0	1
6	0	1	0	1
1	0	1	0	1
5	1	0	1	0
4	1	0	1	0

- Imagine the rows of the boolean matrix permuted under **random permutation** π
- Define minhash function $h_{\pi}(C)$ = the index of the first (in the permuted order π) row in which column C has value 1:

$$h_{\pi}(C) = \min \pi(C)$$

- Use independent hash functions to create a signature of a column

Min-Hashing Example

2nd element of the permutation
is the first to map to a 1

Permutation π

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

Input matrix (Shingles x Documents)

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

4th element of the permutation
is the first to map to a 1

The Min-Hash Property

- Choose a random permutation π
- Claim: $\boldsymbol{Pr[h_\pi(C_1) = h_\pi(C_2)] = sim(C_1, C_2)}$
- Why?
  - Let X be a doc (set of shingles), $y \in X$ is a shingle
  - Then: $\Pr[\pi(y) = \min(\pi(X))] = 1/|X|$
    - It is equally likely that any $y \in X$ is mapped to the min element
  - Let y satisfy $\pi(y) = \min(\pi(C_1 \cup C_2))$
  - Then either: $\pi(y) = \min(\pi(C_1))$ if $y \in C_1$, or
    $\pi(y) = \min(\pi(C_2))$ if $y \in C_2$
  - So the prob. that both are true is the prob. $y \in C_1 \cap C_2$
  - $\Pr[\pi(y) = \min(\pi(C_1)) = \min(\pi(C_2))] = |C_1 \cap C_2| / |C_1 \cup C_2| = sim(C_1, C_2)$

0	0
0	0
1	**1**
0	0
0	1
1	0

One of the two cols had to have 1 at position ***y***

Similarity for Signatures

- $We\ know$: $\Pr[h_\pi(C_1) = h_\pi(C_2)] = sim(C_1, C_2)$
- Now generalize to multiple hash functions

- The similarity of two signatures is the fraction of the hash functions in which they agree

- **Note**: Because of the Min-Hash property, the similarity of columns is the same as the expected similarity of their signatures, with **expected error** of $O\left(\frac{1}{\sqrt{k}}\right)$, k is the number of hash functions.

Min-Hashing Example

Permutation π

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

Input matrix (Shingles x Documents)

1	2	3	4
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

Similarity

	1-3	2-4	1-2	3-4
Jaccard	0.75	0.75	0	0
Sig.	0.67	1.00	0	0

How to pick a random hash function h(x)?

Universal hashing:

$h_{a,b}(x)=((a \cdot x+b) \bmod p) \bmod N$

where:

a,b … random integers

p … prime number (p > N)

Implementation Trick

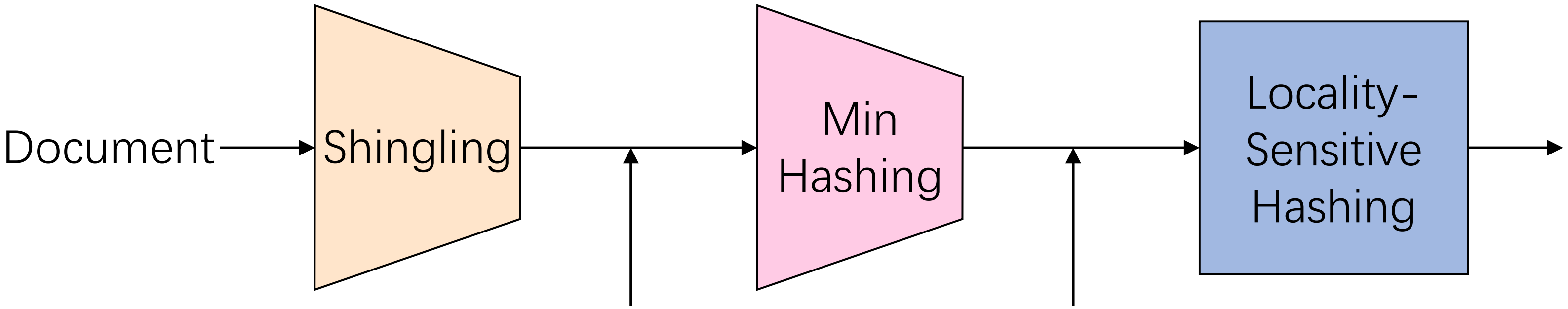
- Permuting rows is complicated, we only need the minimum hashing
- Row hashing
  - Pick $K = 100$ hash functions $\mathbf{h}_i$
  - Ordering under $\mathbf{h}_i$ gives a random row permutation
- One-pass implementation
  - For each column C and hash func. $\mathbf{h}_i$
  - Initialize all $M(i, C) = \infty$, to store the **smallest** hashing value of a document under $\mathbf{h}_i$
  - Scan rows looking for 1s
    - Suppose row j has 1 in column C
    - Then for each $\mathrm{h_i}$:
      - If $\mathrm{h_i}(j) < M(i, C)$,then $M(i, C) = \mathrm{h_i}(j)$.

Signature matrix M

$$\begin{array}{cc} & C \\ i & \begin{bmatrix} 2 & 1 & 2 & 1 \\ 2 & 1 & 4 & 1 \\ 1 & 2 & 1 & 2 \end{bmatrix} \end{array}$$

Locality Sensitive Hashing

- Step 3: Locality-Sensitive Hashing:
  Focus on pairs of signatures likely to be from similar documents



The set of strings of length *k* that appear in the document

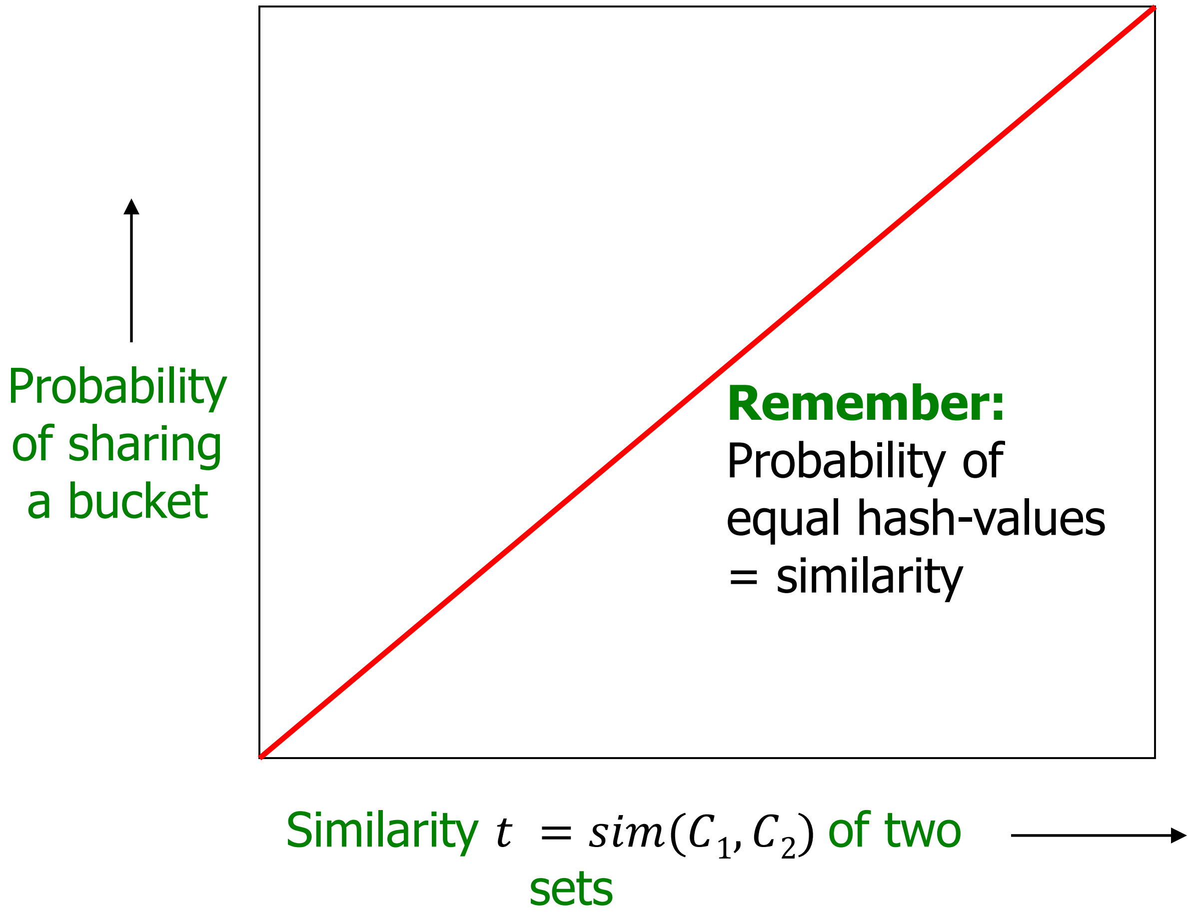
Signatures: short integer vectors that represent the sets, and reflect their similarity

Candidate pairs: those pairs of signatures that we need to test for similarity

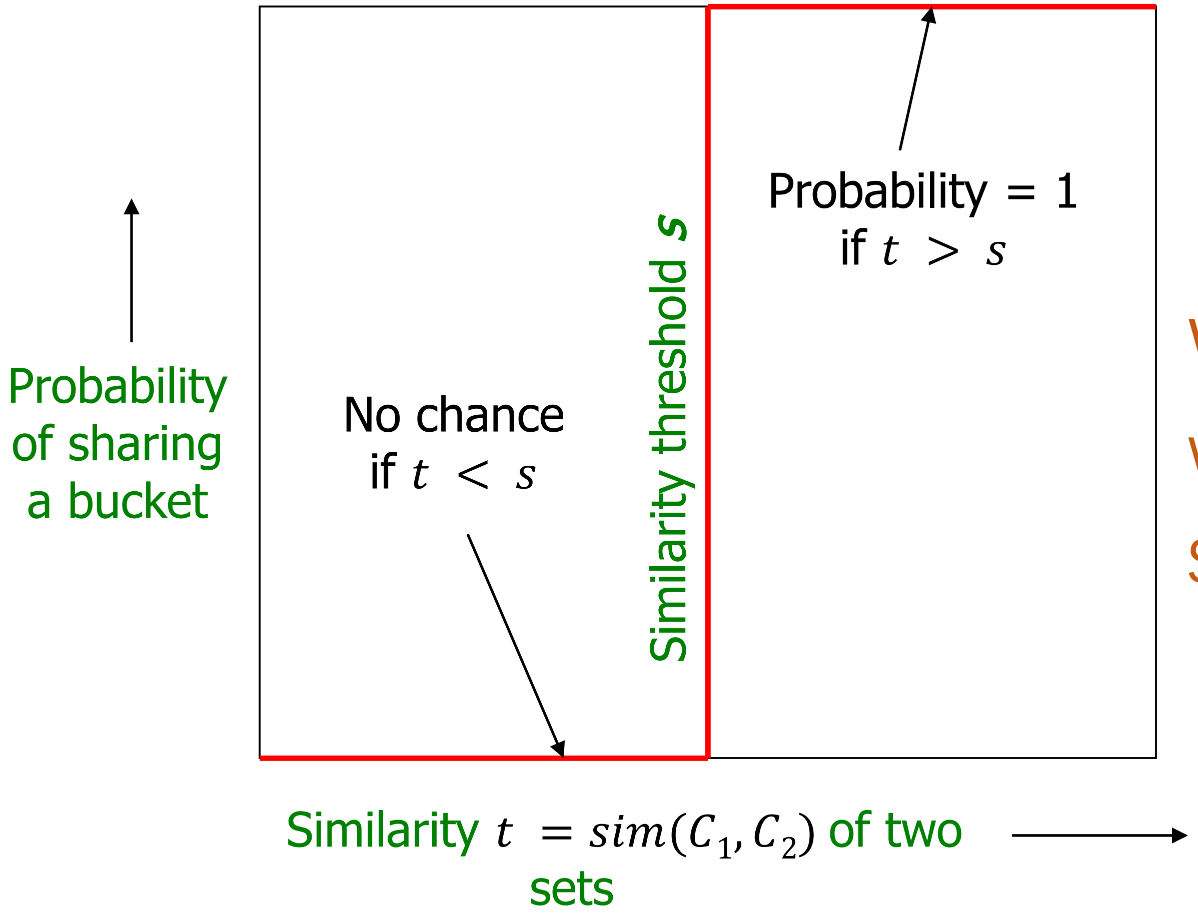
Locality Sensitive Hashing

- **Goal**: Find documents with Jaccard similarity at least s (for some similarity threshold, e.g., $s = 0.8$)
- **LSH – General idea**: Use a **function** $f(x, y)$ that tells whether x and y is a **candidate pair**
- **For Min-Hash matrices:**
  - **Hash** columns of signature matrix M to many buckets
  - Each pair of documents that hashes into the same bucket is a candidate pair

Jaccard Similarity Hashing (1 signature)

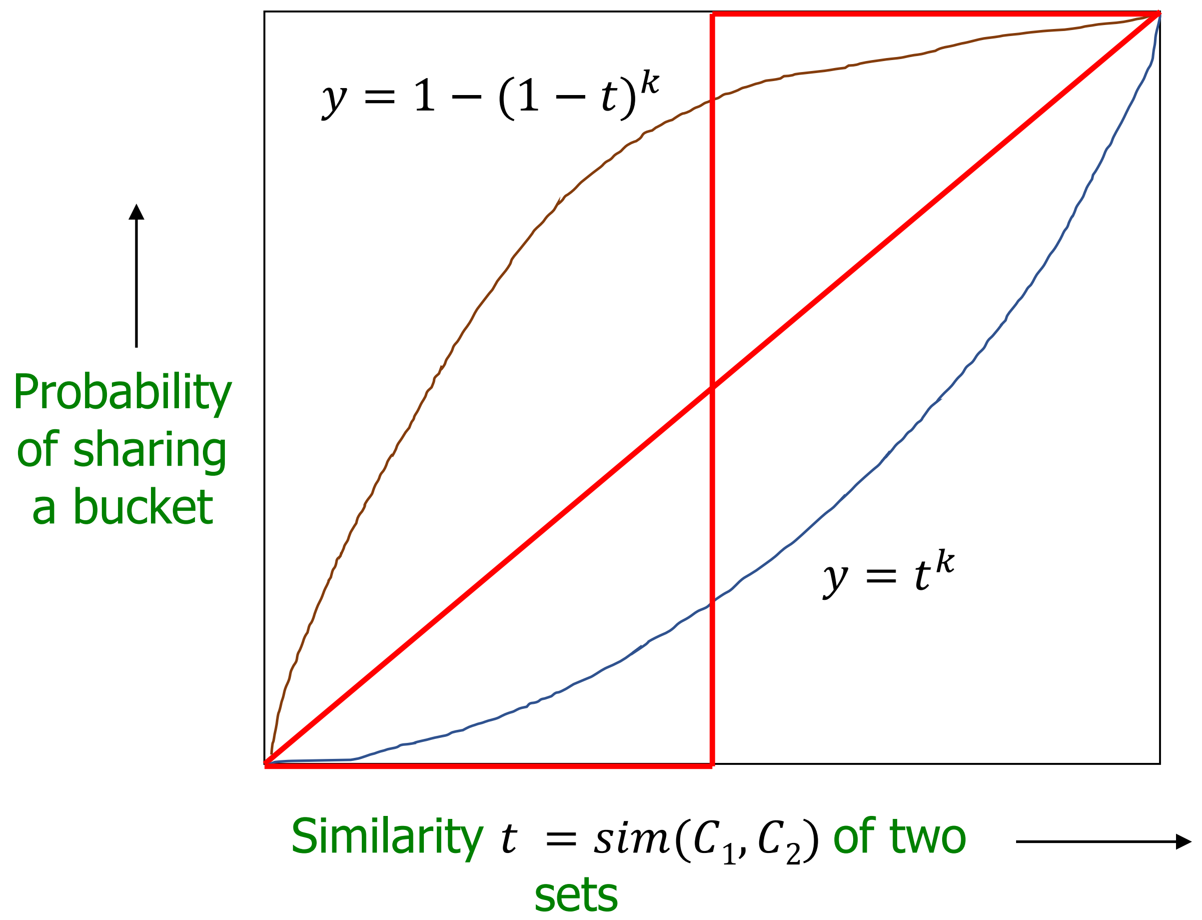


What We Want



What can we do with multiple minhash signatures?

Jaccard Similarity Hashing



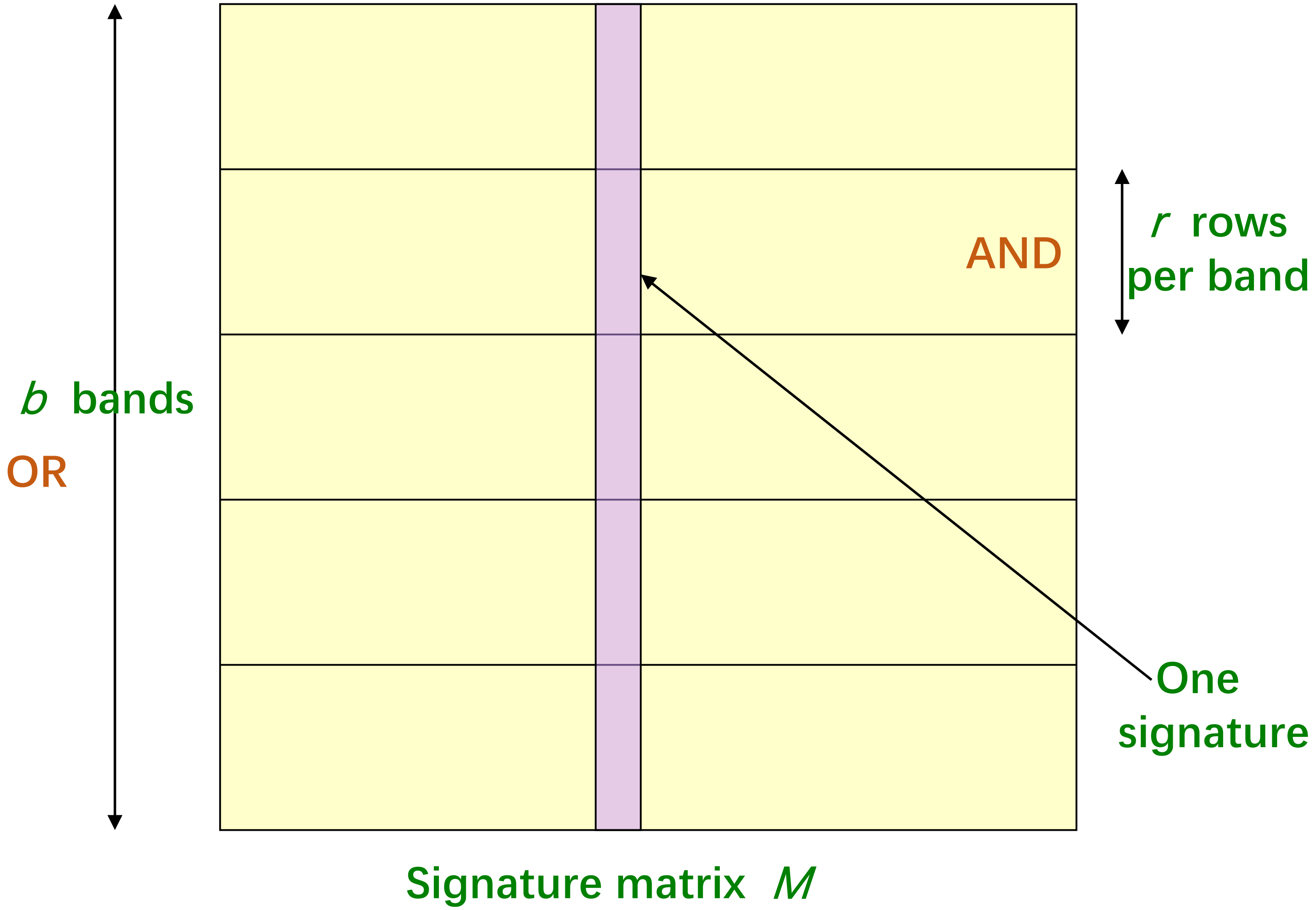
We have k hash functions.

- We consider C_1, C_2 to be a candidate pair, only if they share **all** the k Minhash values (**AND**)
- We consider C_1, C_2 to be a candidate pair, if they share at least **one** Minhash value (**OR**)

What can we do?

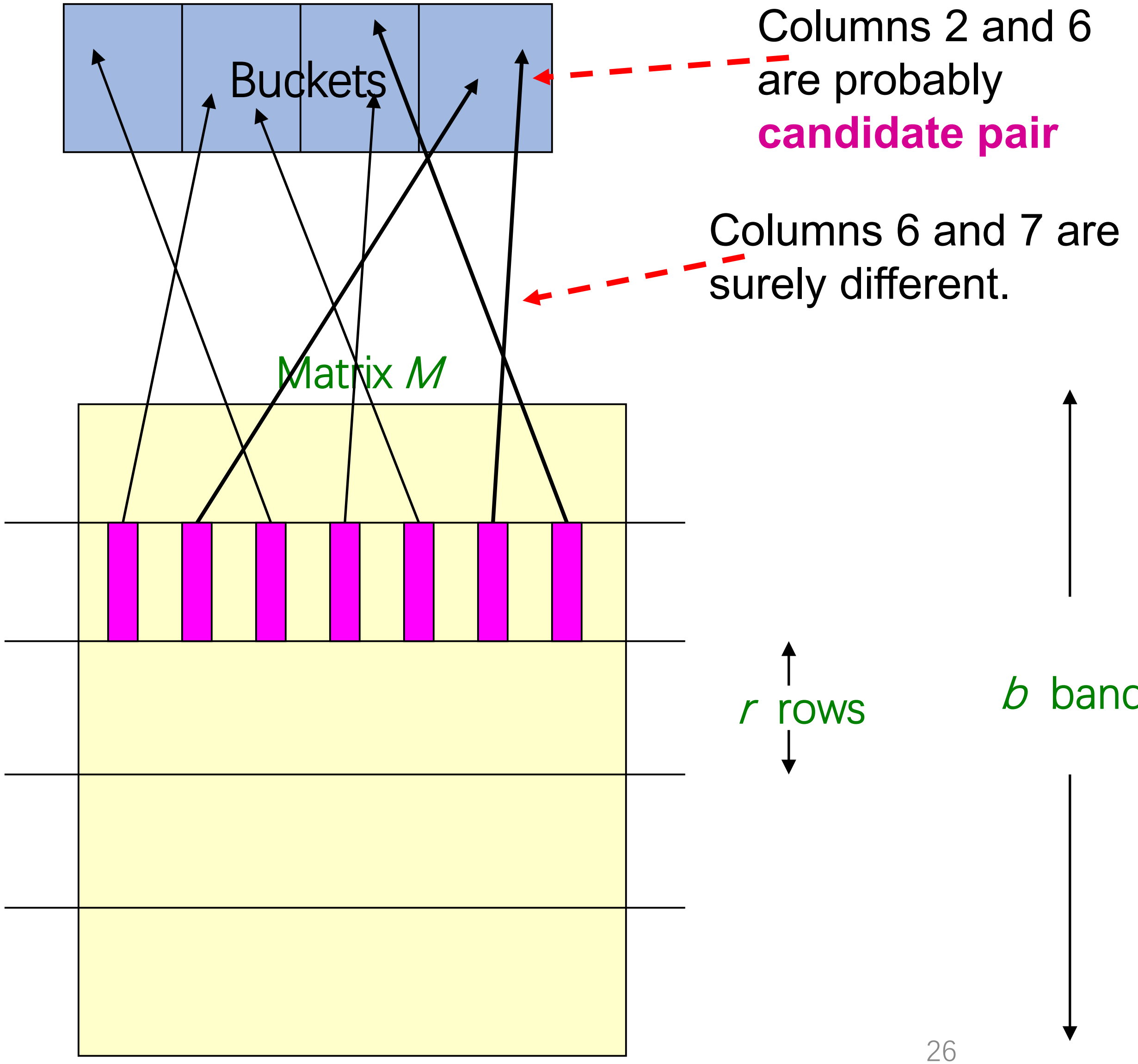
LSH for Min-Hash

- **Key Idea:**
  - Hash columns of signature matrix M several times
  - **Combine OR and AND**



Partition M into Bands

- Divide matrix M into b bands of r rows
- For each band, hash its portion of each column to a hash table (Buckets: as many as possible)
- Candidate column pairs are those that hash to the same bucket for $\geqslant 1$ band
- Tune b and r to catch most similar pairs, but few non-similar pairs



Example of Bands

- Assume the following case:
- Suppose 100,000 columns of M (100k docs)
- Signatures of 100 integers (rows)

- Choose b = 20 bands of r = 5 integers/band

- **Goal**: Find pairs of documents that are at least s = 0.8 similar

C_1, C_2 are 80% Similar, false negative rate?

- Find pairs, similarity $\geq s = 0.8$, set $b = 20, r = 5$
- Assume: $sim(C_1, C_2) = 0.8$
  - Since $sim(C_1, C_2) \geq s$, 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 of the 20 bands:

$$(1 - 0.328)^{20} = 0.00035$$

  - i.e., about 1/3000 of the 80%-similar column pairs are false negatives (we miss them)
  - We would find 99.965% pairs of truly similar documents

C_1, C_2 are 30% Similar, false positive rate?

- Find pairs of similarity ≥ s=0.8, set $b = 20, r = 5$
- Assume: $sim(C_1, C_2) = 0.3$
  - Since $sim(C_1, C_2) < s$ 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 30% similarity end up becoming candidate pairs (false positive)
    - 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

b bands, r rows/band

Pick:

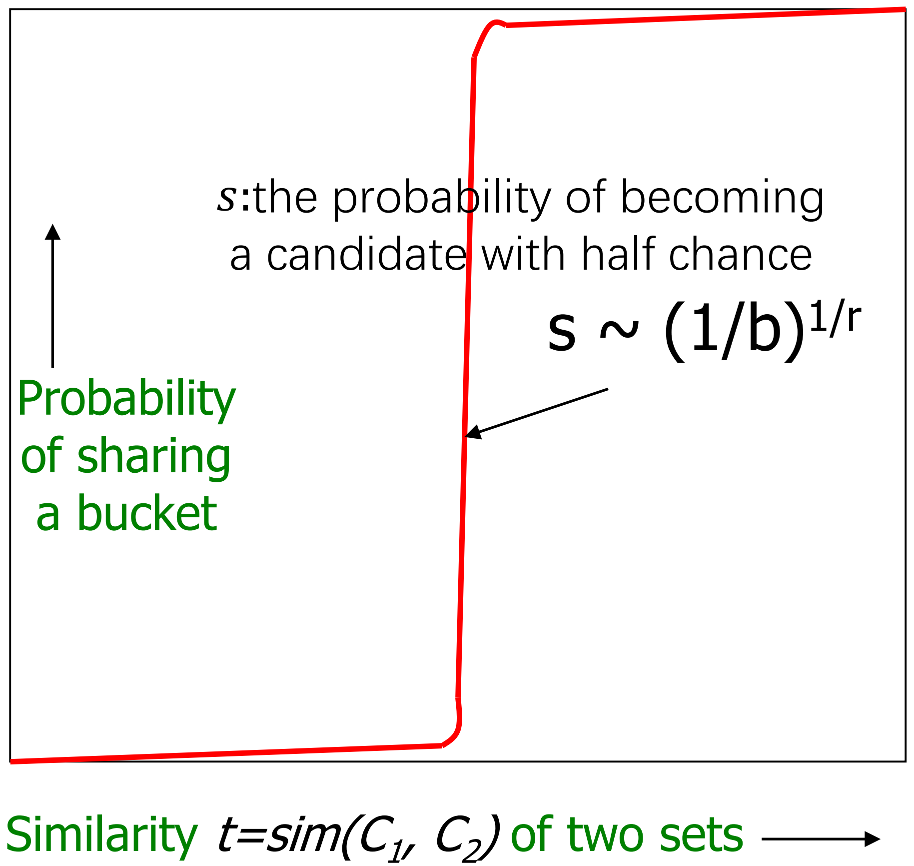
The number of Min-Hashes (rows of M)

The number of bands b, and

The number of rows r per band

to balance false positives/negatives

- Columns C_1 and C_2 have similarity t
- Pick any band (r rows)
  - Prob. that all rows in band equal $= t^r$
  - Prob. that some row in band unequal $= 1 - t^r$
- Prob. that no band identical $= (1 - t^r)^b$
- Prob. that at least 1 band identical $= 1 - (1 - t^r)^b$



At least one band identical

No bands identical

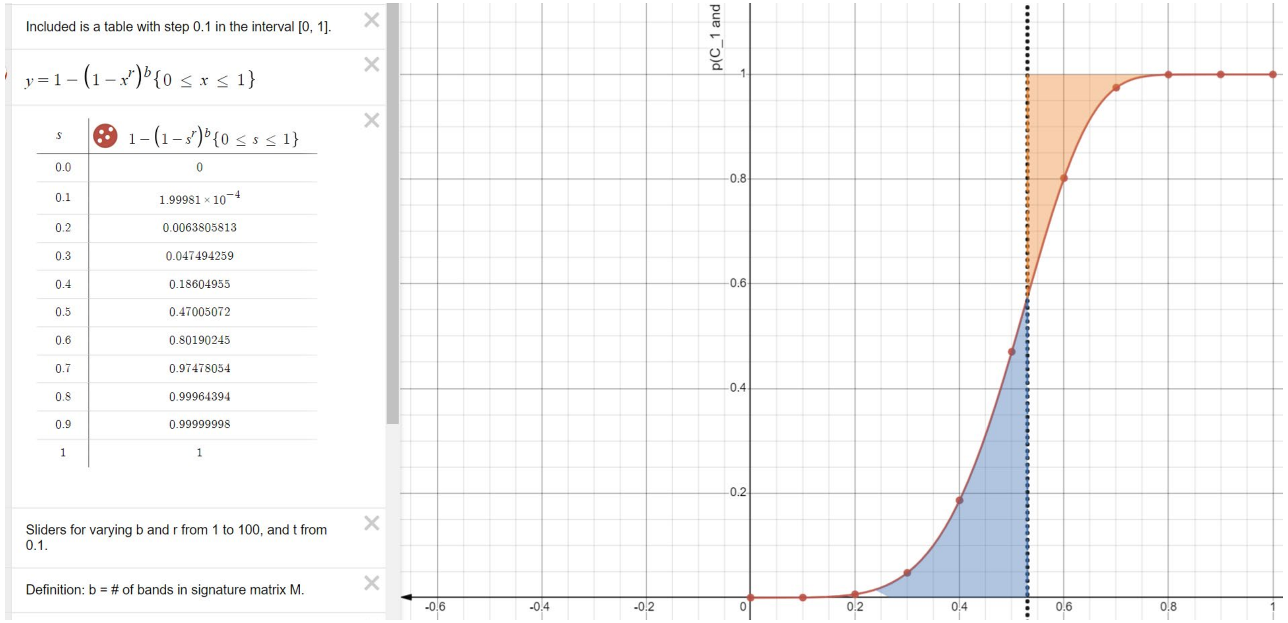
$$\mathbf{1 - (1 - t^r)^b}$$

Some row of a band unequal

All rows of a band are equal

Picking r and b: The S-curve

- Picking r and b to get the best S-curve
  - https://www.desmos.com/calculator/lzzvfjiujn?lang=zh-CN
  - r: hashed into the same bucket, b: identified as similar.



LSH Summary

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

- Check in main memory that candidate pairs really do have similar signatures

- Optional: In another pass through data, check that the remaining candidate pairs really represent similar documents