MIN-HASHING AND LOCALITY SENSITIVE HASHING

Task: Finding Similar Documents

- **Goal:** **Given a large number (N in the millions or billions) of documents, find "near duplicate" pairs**
- **Applications:**
  - Mirror websites, or approximate mirrors
    - Don't want to show both in search results
  - Similar news articles at many news sites
    - Cluster articles by "same story"

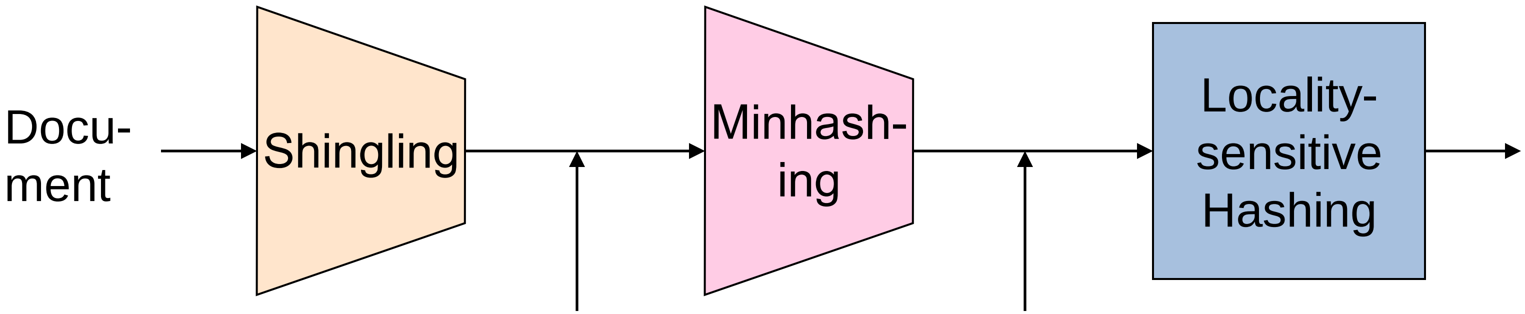
Main issues

- What is the right representation of the document when we check for similarity?
  - E.g., representing a document as a set of characters will not do (why?)
- When we have billions of documents, keeping the full text in memory is not an option.
  - We need to find a shorter representation

3 Essential Steps for Similar Docs

1. ***Shingling:*** Convert documents to sets
2. ***Min-Hashing:*** Convert large sets to short signatures, while preserving similarity
3. ***Locality-Sensitive Hashing:*** Focus on pairs of signatures likely to be from similar documents
   - **Candidate pairs**

The Big Picture



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.

Shingling Documents

Define: Shingles

- A *k*-shingle (or *k*-gram) for a document is defined to be any substring of length k found within the document.
- We associate with each document the **set of k-shingles** that appear one or more times within that document.
- **Example:**

  Suppose our document D is the string **abcdabd**, and **k = 2**.

Set of 2-shingles for D is {ab, bc, cd, da, bd}.

Option: Shingles as a bag (multiset), count ab twice: **$S'(D_1)$ =** {ab, bc, ca, ab}

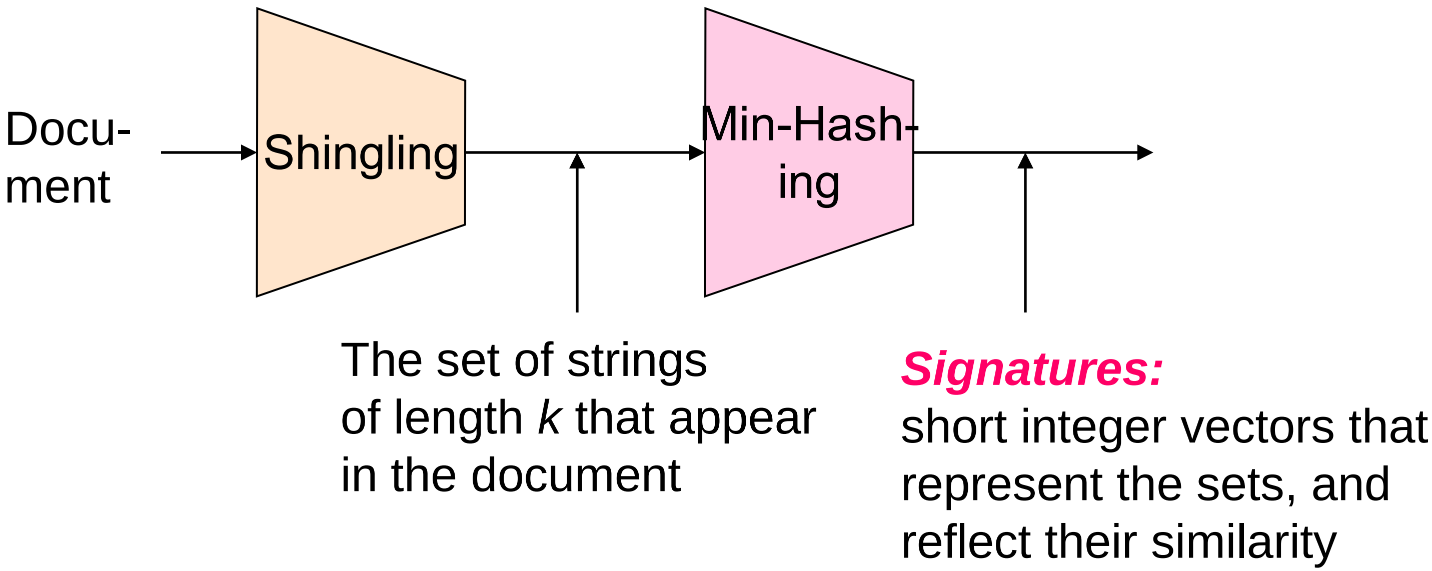
Working Assumption

- **Documents that have lots of shingles in common have similar text, even if the text appears in different order**

- **Caveat:** You must pick ***k*** large enough, or most documents will have most shingles
  - ***k*** = 5 is OK for short documents
  - ***k*** = 10 is better for long documents

Motivation for Minhashing / LSH

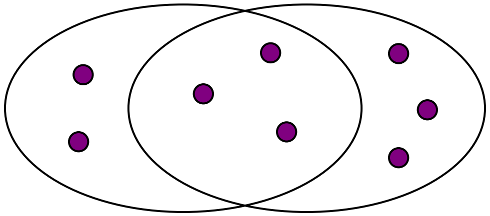
- **Suppose we need to find near-duplicate documents among $N = 1$ million documents**
- Naïvely, we would have to compute **pairwise Jaccard similarities** for **every pair of docs**
  - $N(N-1)/2$ ≈ **5*10^{11}** comparisons
  - At 10^5 secs/day and 10^6 comparisons/sec, it would take **5 days**
- For $N = 10$ million, it takes more than a year…



MINHASHING

Minhashing: Convert **large sets** to **short signatures**, while <u>**preserving similarity**</u>

Encoding Sets as Bit Vectors



- Many similarity problems can be formalized as **finding subsets that have significant intersection**
- **Encode sets using 0/1 (bit, boolean) vectors**
  - One dimension per element in the universal set
- Interpret set intersection as bitwise **AND**, and set union as bitwise **OR**

- **Example:** $\mathbf{C_1}$ = 10111; $\mathbf{C_2}$ = 10011
  - Size of intersection **= 3**; size of union **= 4**,
  - **Jaccard similarity** (not distance) **= 3/4**
  - **Distance: $\mathbf{d(C_1,C_2)}$ = 1 – (Jaccard similarity) = 1/4**

Representing Sets as Boolean Matrices

- **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: 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 – (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

Matrix Representation of Set

- The columns of the matrix correspond to the sets, and the rows correspond to elements of the universal set from which elements of the sets are drawn.
- A value 1 in row r and column c if the element for row r is a member of the set for column c.
- Otherwise the value in position (r, c) is 0.

Element	S_1	S_2	S_3	S_4
a	1	0	0	1
b	0	0	1	0
c	0	1	0	1
d	1	0	1	1
e	0	0	1	0

Finding Similar Columns

- **Next Goal: Find similar columns, Small signatures**
- **Naïve approach:**
  - **1) Signatures of columns:** small summaries of columns
  - **2) Examine pairs of signatures** to find similar columns
    - **Essential:** Similarities of signatures and columns are related
  - **3) Optional:** Check that columns with similar signatures are really similar
- **Warnings:**
  - Comparing all pairs may take too much time: **Job for LSH**
    - These methods can produce false negatives, and even false positives (if the optional check is not made)

Hashing Columns (Signatures)

- **Key idea:** "hash" each column ***C*** to a small ***signature h(C)***, such that:
  - **(1)** ***h(C)*** is small enough that the signature fits in RAM
  - **(2)** $\boldsymbol{sim(C_1, C_2)}$ is the same as the "similarity" of signatures $\boldsymbol{h(C_1)}$ and $\boldsymbol{h(C_2)}$

- **Goal: Find a hash function *h(·)* such that:**
  - If $\boldsymbol{sim(C_1,C_2)}$ is high, then with high prob. $\boldsymbol{h(C_1) = h(C_2)}$
  - If $\boldsymbol{sim(C_1,C_2)}$ is low, then with high prob. $\boldsymbol{h(C_1) \neq h(C_2)}$

- **Hash docs into buckets. Expect that "most" pairs of near duplicate docs hash into the same bucket!**

Min-Hashing

- **Goal: Find a hash function *h(·)* such that:**
  - if $sim(C_1,C_2)$ is high, then with high prob. $h(C_1) = h(C_2)$
  - if $sim(C_1,C_2)$ is low, then with high prob. $h(C_1) \neq h(C_2)$

- **Clearly, the hash function depends on the similarity metric:**
  - Not all similarity metrics have a suitable hash function
- **There is a suitable hash function for the Jaccard similarity:** It is called **Min-Hashing**

Min-Hashing

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

$$h_\pi(C) = min_\pi\, \pi(C)$$

- Use several (e.g., 100) independent hash functions (that is, permutations) to create a signature of a column
- The **minhash value** of any column **is the number of the first row**, in **the permuted order**, in which the column **has a 1**.

Example:

Element	S_1	S_2	S_3	S_4
b	0	0	1	0
e	0	0	1	0
a	1	0	0	1
d	1	0	1	1
c	0	1	0	1

Element	S_1	S_2	S_3	S_4
a	1	0	0	1
b	0	0	1	0
c	0	1	0	1
d	1	0	1	1
e	0	0	1	0

- Suppose, we pick the order of rows **beadc** for the matrix given
- This permutation defines a **minhash function h** that maps sets to rows.

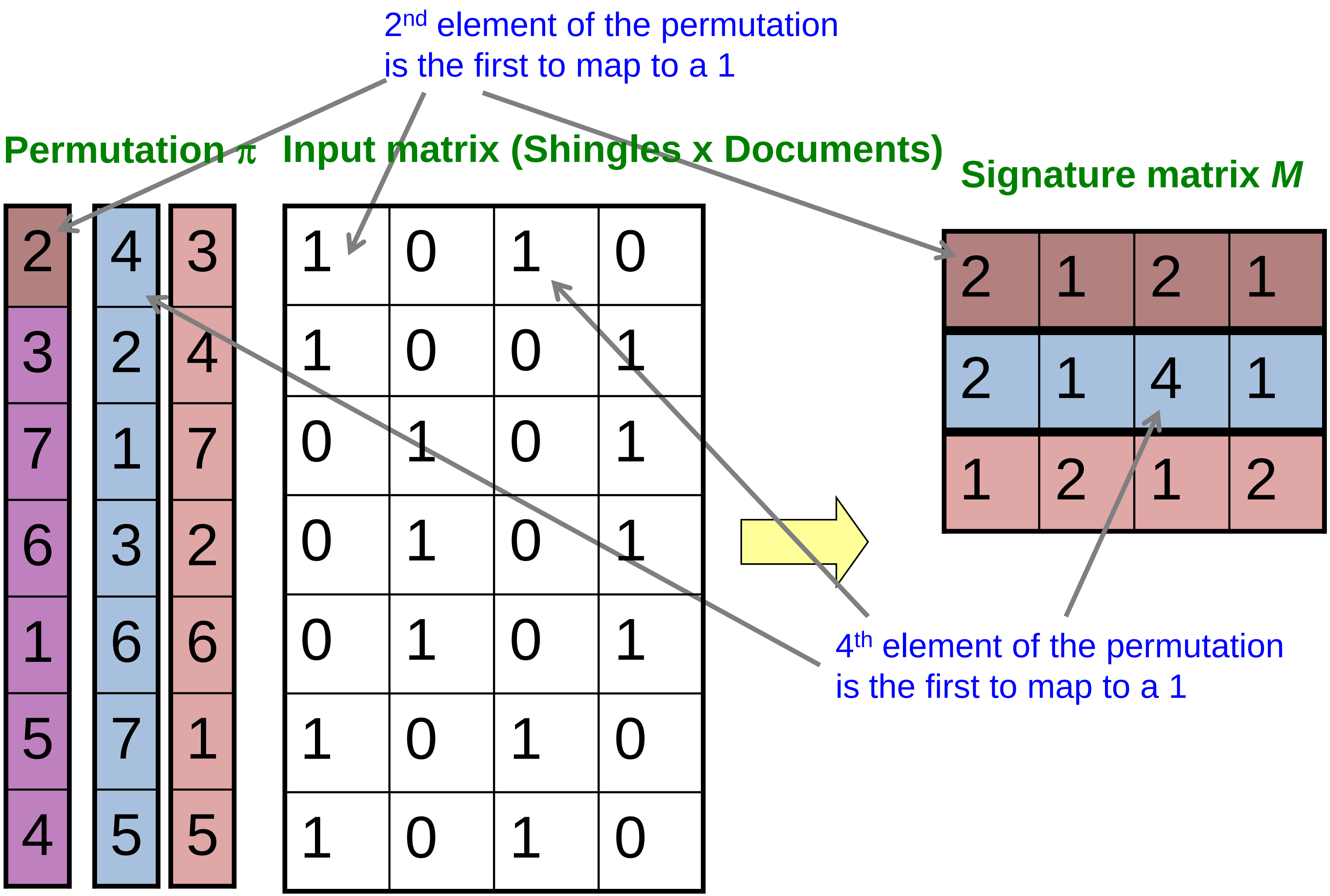
Computing minhash value of set S1 according to h:

- First column, column for set S1, has 0 in row b, 0 in row e, and a 1 in row a, so **h(S1) = a**
- Similarly, **h(S2) = c, h(S3) = b, and h(S4) = a.**

Min-Hashing

- Pick a random permutation of the rows (the universe U).
- Define "hash" function for set S
  - h(S) = the index of the first row (in the permuted order) in which column S has 1.
  - OR
  - h(S) = the index of the first element of S in the permuted order.
- Use k (e.g., k = 100) independent random permutations to create a signature.

Min-Hashing Example



Four Types of Rows

- **Given cols C_1 and C_2, rows may be classified as:**

	C_1	C_2
A	1	1
B	1	0
C	0	1
D	0	0

  - **a** = # rows of type A, etc.
- **Note: sim(C_1, C_2) = a/(a +b +c)**
- **Then: Pr**[$h(C_1) = h(C_2)$] = $Sim(C_1, C_2)$
  - Look down the cols C_1 and C_2 until we see a 1
  - If it's a type-*A* row, then $h(C_1) = h(C_2)$
    If a type-*B* or type-*C* row, then not

Minhash Signatures

- Collection of sets represented by some characteristic matrix M.
- Call the minhash functions determined by these permutations h1, h2, . . . , hn.
- From the column representing set S, construct the minhash signature for S, the vector [h1(S), h2(S), . . . , hn(S)], normally represented as a column.
- Thus, we can form a signature matrix from matrix M , in which the ith column of M is replaced by the minhash signature for (the set of) the ith column.

Computing Minhash Signature

- Let **SIG(i, c)** be the element of the signature matrix for the ith hash function and column c.
- Initially, set SIG(i, c) to ∞ for all i and c.
- We handle row r by doing the following:

1. Compute $h_1(r), h_2(r), \ldots, h_n(r)$.
2. For each column c do the following:
   (a) If c has 0 in row r, do nothing.
   (b) However, if c has 1 in row r, then for each $i = 1, 2, \ldots, n$ set $\text{SIG}(i, c)$ to the smaller of the current value of $\text{SIG}(i, c)$ and $h_i(r)$.

Row	S_1	S_2	S_3	S_4	$x + 1 \mod 5$	$3x + 1 \mod 5$
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

Steps of Signature matrix

Row	S_1	S_2	S_3	S_4	$x+1 \mod 5$	$3x+1 \mod 5$
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

- Step 1:

	S_1	S_2	S_3	S_4
h_1	∞	∞	∞	∞
h_2	∞	∞	∞	∞

- Step 2:

	S_1	S_2	S_3	S_4
h_1	1	∞	∞	1
h_2	1	∞	∞	1

- Step 3:

	S_1	S_2	S_3	S_4
h_1	1	∞	2	1
h_2	1	∞	4	1

Contd…

Row	S_1	S_2	S_3	S_4	$x+1 \mod 5$	$3x+1 \mod 5$
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

- Step 4:

	S_1	S_2	S_3	S_4
h_1	1	3	2	1
h_2	1	2	4	1

- Step 5:

	S_1	S_2	S_3	S_4
h_1	1	3	2	1
h_2	0	2	0	0

- Step 6:

	S_1	S_2	S_3	S_4
h_1	1	3	0	1
h_2	0	2	0	0

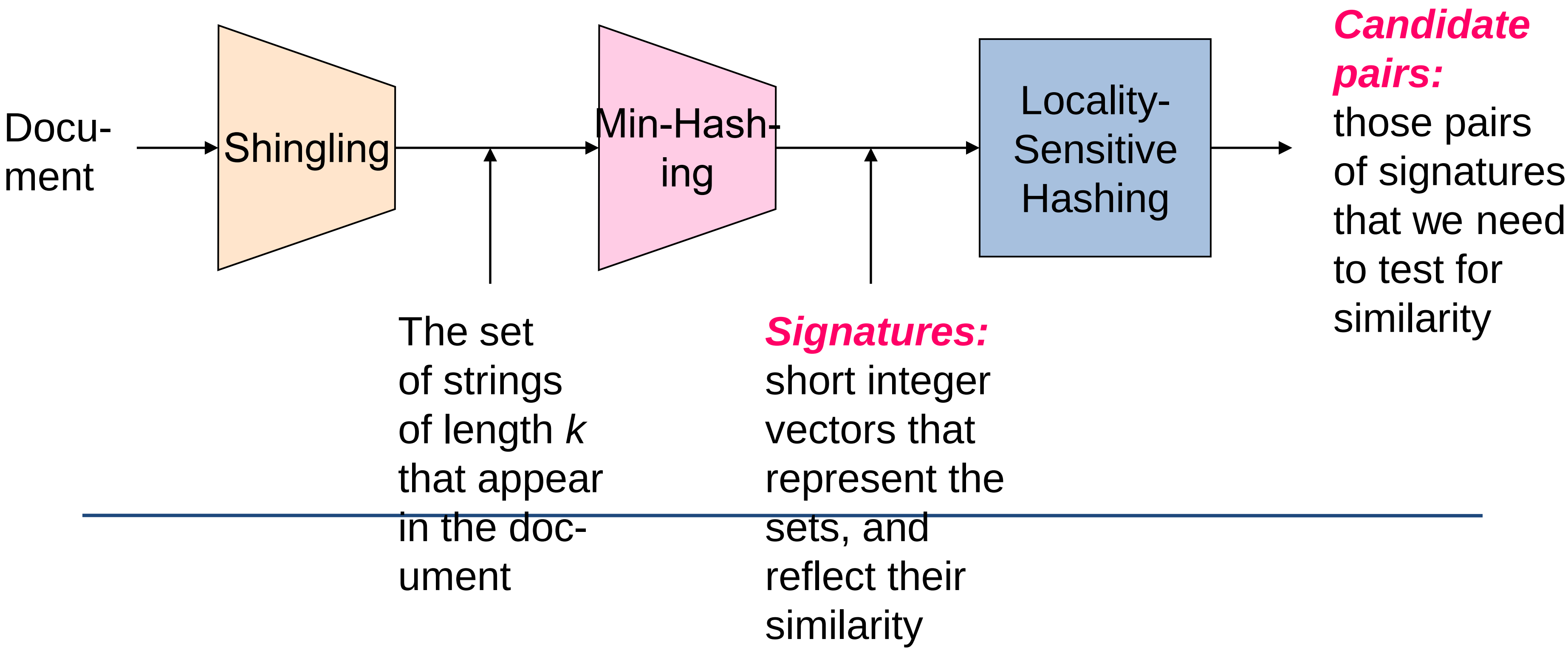
Example

- Using the data given, determine signatures of the columns the values of the following hash functions:

(a) $h_3(x) = 2x + 4 \mod 5.$

(b) $h_4(x) = 3x - 1 \mod 5.$

Element	S_1	S_2	S_3	S_4
0	0	1	0	1
1	0	1	0	0
2	1	0	0	1
3	0	0	1	0
4	0	0	1	1
5	1	0	0	0



Locality-Sensitive Hashing:

Focus on pairs of signatures likely to be from similar documents

Locality Sensitive Hashing

- One general approach to LSH is to “**hash**” items several times, in such a way that similar items are more likely to be hashed to the same bucket than dissimilar items are.
- Any pair that hashed to the same bucket for any of the hashings is termed to be a **candidate pair**

LSH:

- **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**

Candidates from Min-Hash

- **Pick a similarity threshold *s* (0 < s < 1)**

- Columns ***x*** and ***y*** of ***M(Signature matrix)*** are a **candidate pair** if their signatures agree on at least fraction ***s*** of their rows:
  M **(*i, x*) = *M* (*i, y*)** for at least frac. ***s*** values of ***i***
  - We expect documents ***x*** and ***y*** to have the same (Jaccard) similarity as their signatures

LSH for Min-Hash

- **Big idea: Hash columns of signature matrix *M* several times**

- Arrange that (only) **similar columns** are likely to **hash to the same bucket**, with high probability

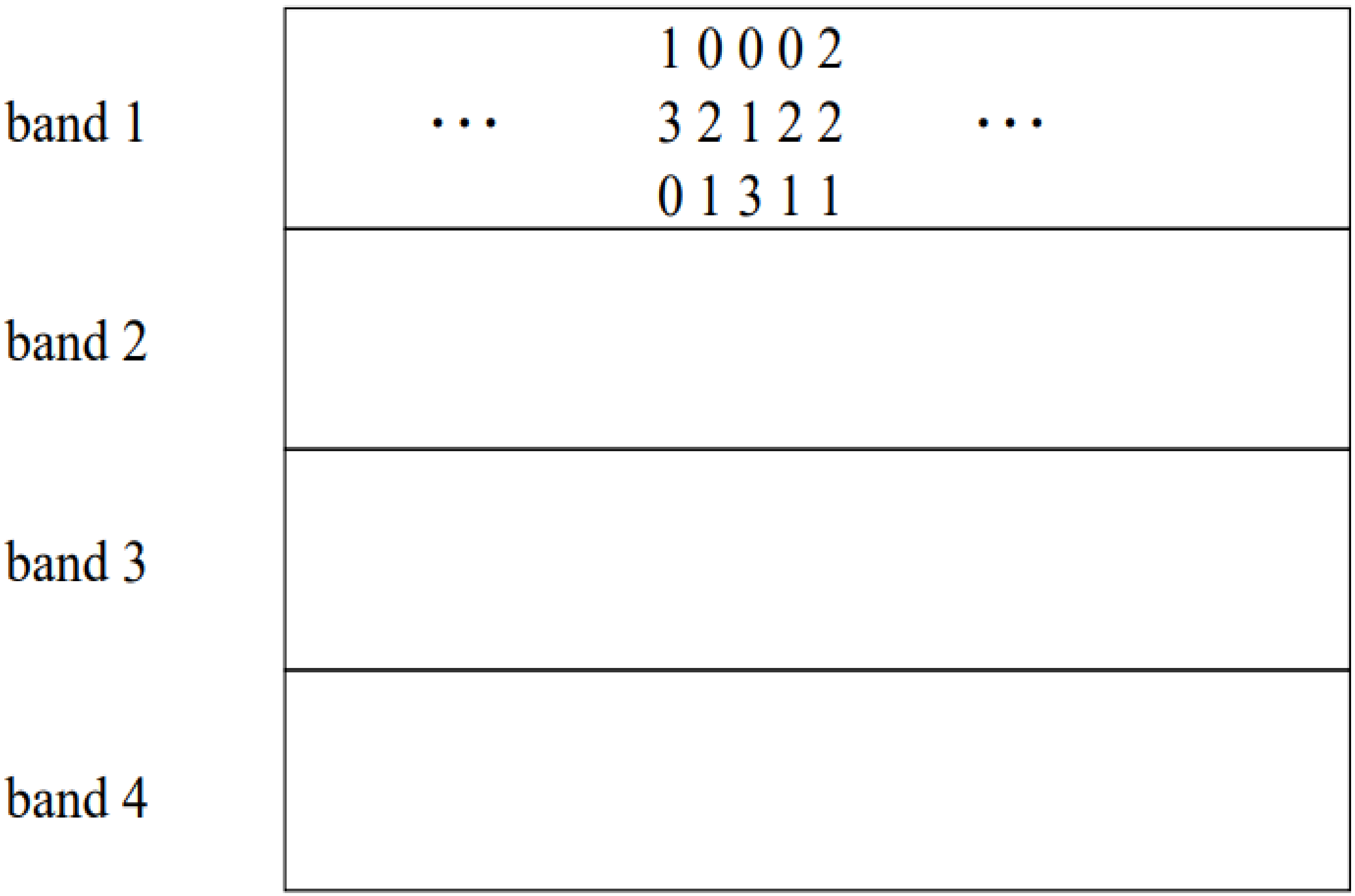
- **Candidate pairs are those that hash to the same bucket**

Partition into Bands

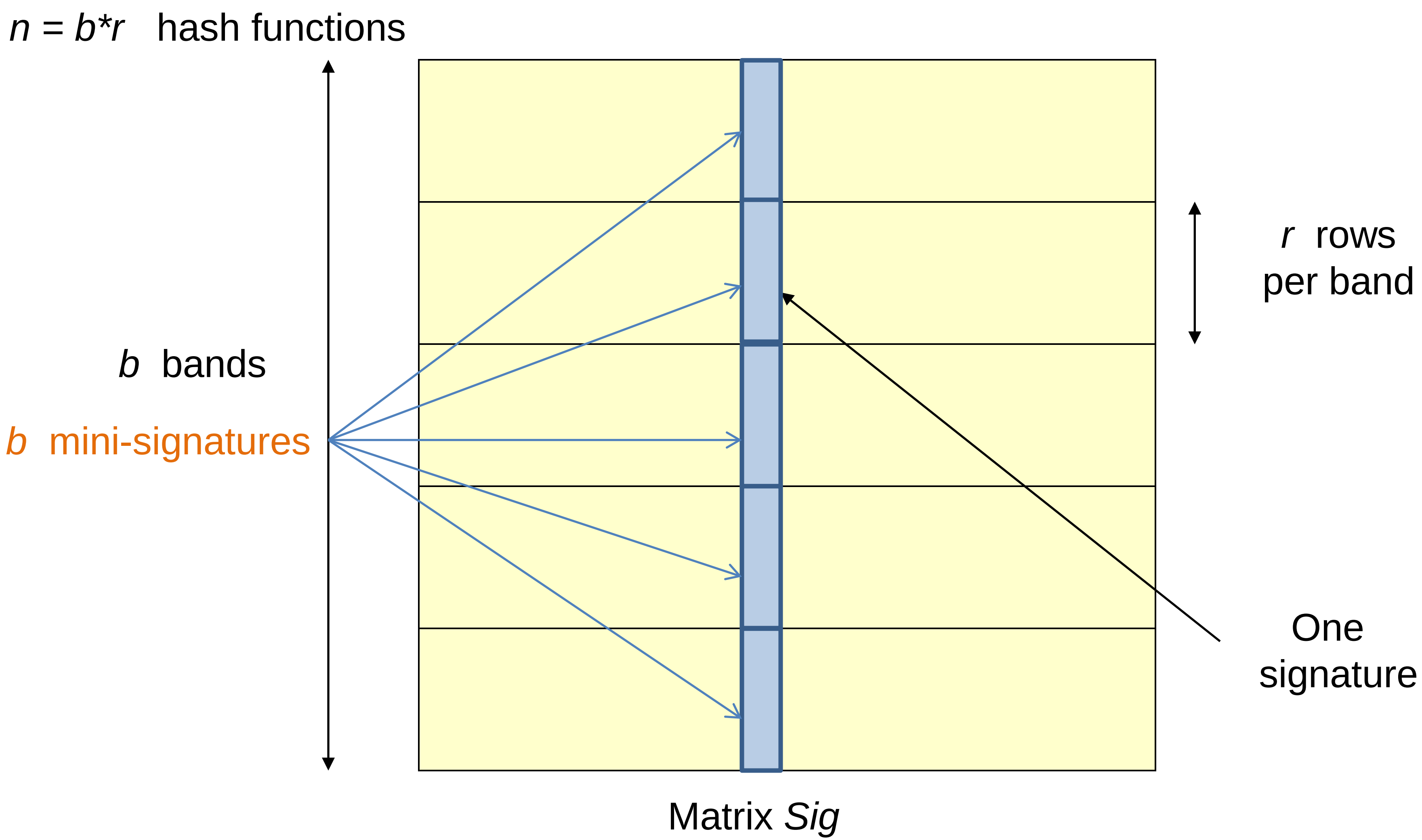
- Divide the signature matrix Sig into b bands of r rows.
  - Each band is a mini-signature with r hash functions.

- For each band, there is a hash function that takes vectors of r integers (the portion of one column within that band) and hash them to some large no of buckets.
- Same hash function can be used for all bands but use separate bucket for each band

Signature matrix divided into four bands of three rows per band



Partitioning into bands



Partition into Bands

- Divide the signature matrix Sig into *b* bands of *r* rows.
  - Each band is a mini-signature with r hash functions.
- For each band, hash the mini-signature to a hash table with *k* buckets.
  - Make *k* as large as possible so that mini-signatures that hash to the same bucket are almost certainly identical.
- Candidate column pairs are those that hash to the same bucket for at least 1 band.
- Tune *b* and *r* to catch most similar pairs, but few non-similar pairs.

Analysis of the Banding Technique

Suppose we use b bands of r rows each, and suppose that a particular pair of documents have Jaccard similarity s. We can calculate the probability that these documents (or rather their signatures) become a candidate pair as follows:

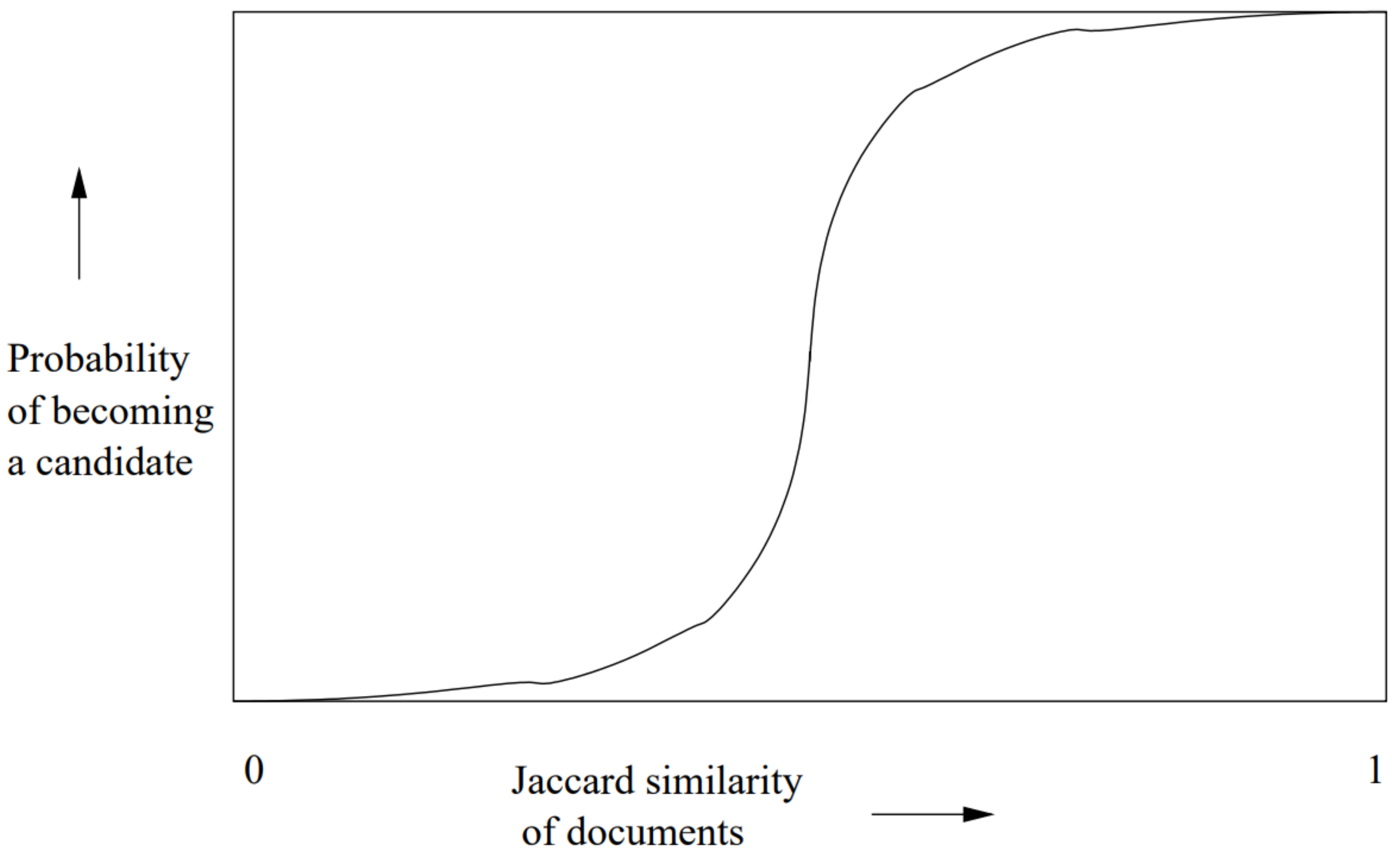
1. The probability that the signatures agree in all rows of one particular band is s^r.

2. The probability that the signatures disagree in at least one row of a particular band is $1 - s^r$.

3. The probability that the signatures disagree in at least one row of each of the bands is $(1 - s^r)^b$.

4. The probability that the signatures agree in all the rows of at least one band, and therefore become a candidate pair, is $1 - (1 - s^r)^b$.

- The function **$1-(1-s^r)^b$** has the form of s-curve regardless of constants b and r



- The **threshold**, that is, the value of similarity **s** at which the probability of becoming a candidate is **1/2,** is a function of b and r.
- The threshold is roughly where the rise is the steepest, and for large b and r there we find that pairs with similarity above the threshold are very likely to become candidates, while those below the threshold are unlikely to become candidates.
- An approximation to the threshold is **(1/b)^1/r.** For example, if b = 16 and r = 4, then the threshold is approximately at s = 1/2, since the 4th root of 1/16 is 1/2.

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

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

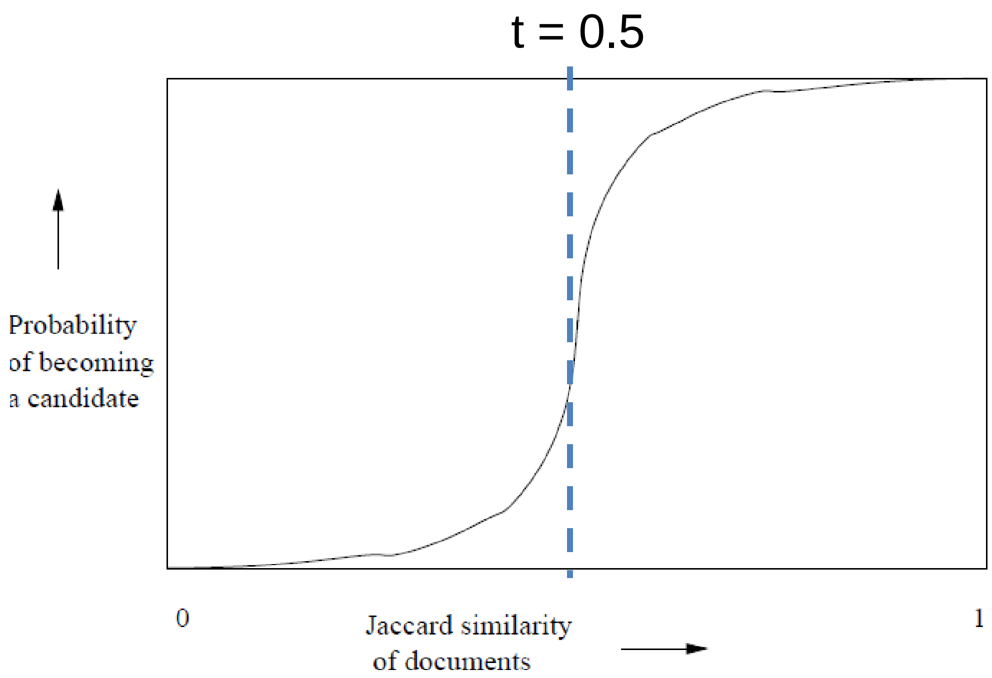


Figure 3.7: The S-curve

Suppose S_1, S_2 are 80% Similar

- We want all 80%-similar pairs. Choose 20 bands of 5 integers/band.

- Probability S_1, S_2 identical in one particular band:

$$(0.8)^5 = 0.328.$$

- Probability S_1, S_2 are not similar in any of the 20 bands:

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

  - i.e., about 1/3000-th of the 80%-similar column pairs are false negatives.

- Probability S_1, S_2 are similar in at least one of the 20 bands:

$$1-0.00035 = 0.999$$

C_1, C_2 are 30% Similar

- **Find pairs of $\geq$ *s***=0.8 similarity, 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 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**

LSH Summary

- Tune to get almost all pairs with similar signatures, but 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 *sets* .

Applications

- Entity Matching/Record Linkage
- Advertising Matching
- Image Search
- News Article
- Fingerprints