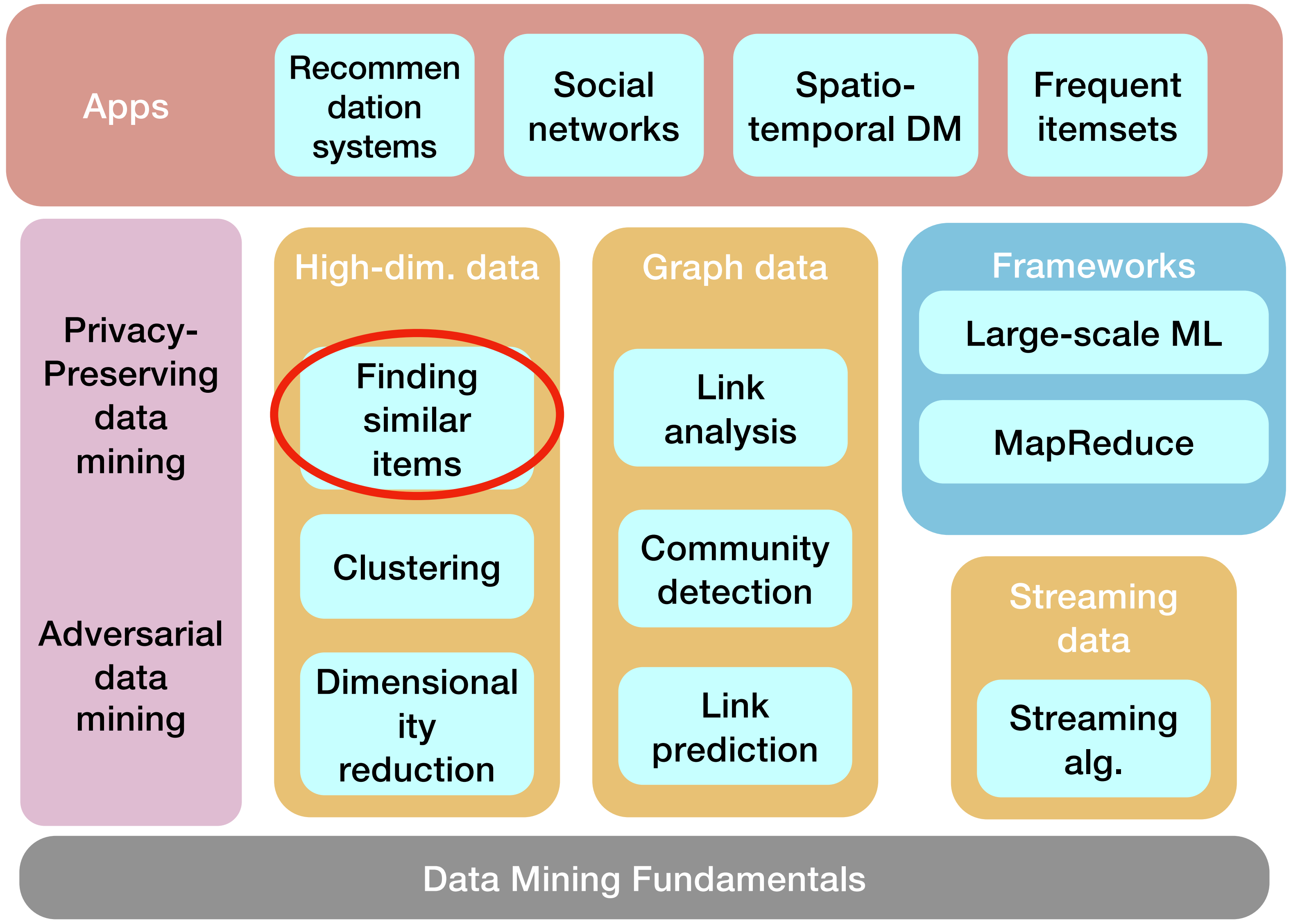
EE359 Data Mining Lecture 3

Finding Similar Items

Liyao Xiang
http://xiangliyao.cn/
Shanghai Jiao Tong University

Course Landscape



Outline

- Locality-Sensitive Hashing
- Applications of Locality-Sensitive Hashing
- Distance Measures
- Locality-Sensitive Functions
- Methods for High Degrees of Similarity

Outline

- Locality-Sensitive Hashing
- Applications of Locality-Sensitive Hashing
- Distance Measures
- Locality-Sensitive Functions
- Methods for High Degrees of Similarity

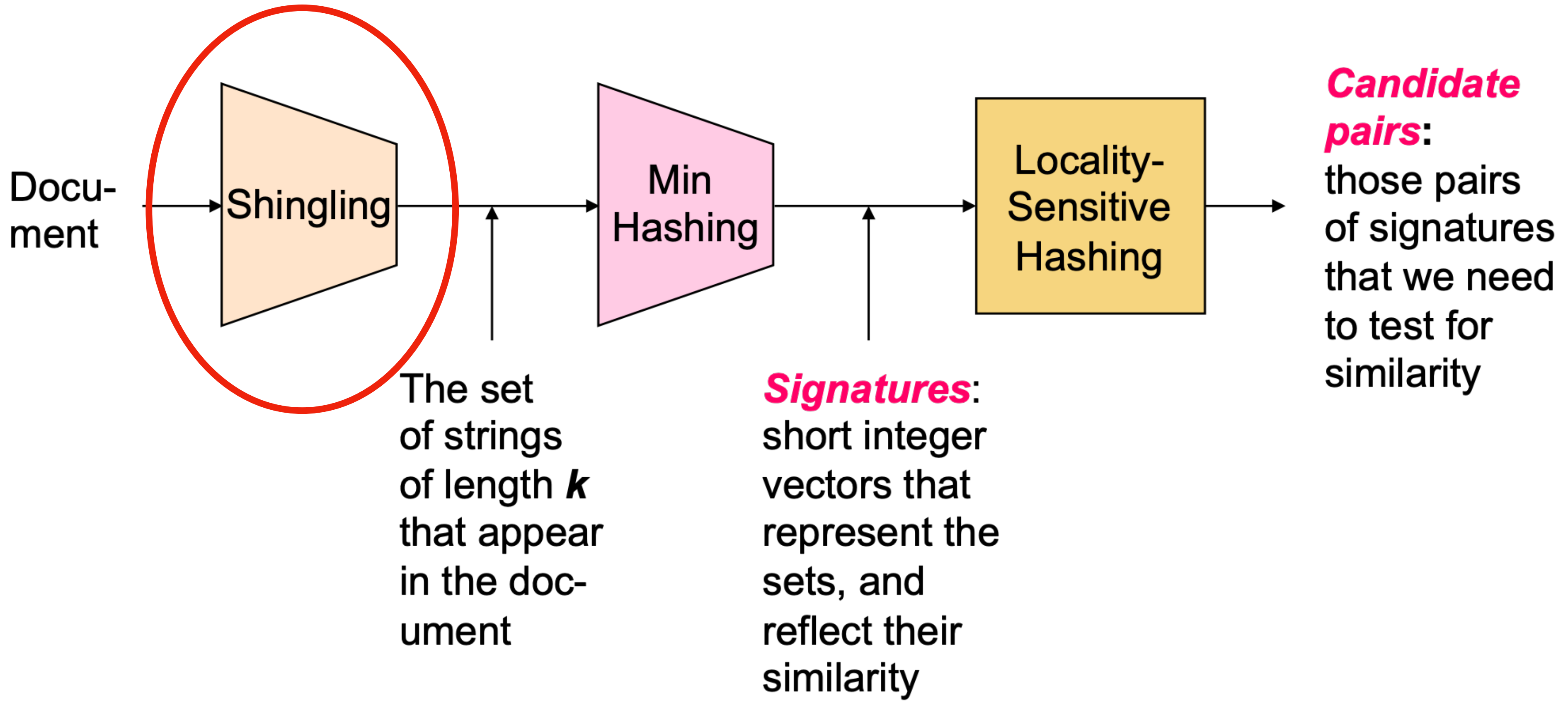
Finding Similar Documents

- **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'
- **Problems:**
  - Many small pieces of one document can appear out of order in another
  - Too many documents to compare all pairs
  - Documents are so large or so many that they cannot fit in main memory

Locality-Sensitive Hashing

- Focus on pairs that are likely to be similar
- **Idea:** hash items using many different hash functions
  - similar pairs wind up in the same bucket
  - examine only the candidate pairs
- **Example:** finding similar documents
  - **Shingling:** convert documents into sets
  - **Min-Hashing:** Convert large sets to short signatures, while preserving similarity
  - **Locality-Sensitive Hashing**

Locality-Sensitive Hashing


Docu-
ment
Shingling
Min
Hashing
Locality-
Sensitive
Hashing
Candidate pairs:
those pairs of signatures that we need to test for similarity
The set of strings of length *k* that appear in the doc-
ument
Signatures:
short integer vectors that represent the sets, and reflect their similarity

Shingling

- Convert documents to sets
- Simple approaches:
    - Document = set of words appearing in document
    - Document = set of `important’ words
    - Don’t work well. Why?
- Need to account for **ordering** of words!
- 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 else. E.g., k=2; document D1 = abcab;

Set of 2-shingles: S(D1) = {ab, bc, ca}

Compressing Shingles

- To compress long shingles, we can hash them to a few bytes
- Represent a document by the set of **hash values** of its k-shingles
  - E.g., k=2; document D1 = abcab
    - set of 2-shingles: S(D1) = {ab, bc, ca}
    - hash the singles: h(D1) = {1, 5, 7}
    - document D1 is a set of its k-shingles C1 = S(D1)

Compressing Shingles

- Construct the set of 9-shingles for a document and map each of those 9-shingles to a bucket number in $[0, 2^{32}-1]$
  - each shingle is represented by 4 bytes instead of 9
  - **save space**!
- Why not use 4-shingles?
  - assume only 20 characters are frequently used, the number of different 4-shingles that are likely to occur is only $(20)^4 = 160000$
  - if use 9-shingles, there are more than 2^{32} likely shingles

Shingles

- Each document is a 0/1 vector in the space of k-shingles
  - Each unique shingle is a dimension
  - Vectors are very sparse
- Documents that have lots of shingles in common have similar text, even if the text appears in different order
- Choice of k:
  - k = 5 is OK for short documents
  - k = 10 is better for long documents

Shingles Built from Words

- Find web **pages** that had the same **articles**, regardless of the **surrounding elements**
- Need to bias the set of shingles in favor of the article
  - e.g., pages with the same article but different surrounding material have higher similarity
- News articles, prose have many stop words: “and,” “you,” “to,”…
  - Not contribute to the topic
  - But useful to distinguish from the surrounding elements
- Define a shingle to be a stop word followed by the next two words

Question

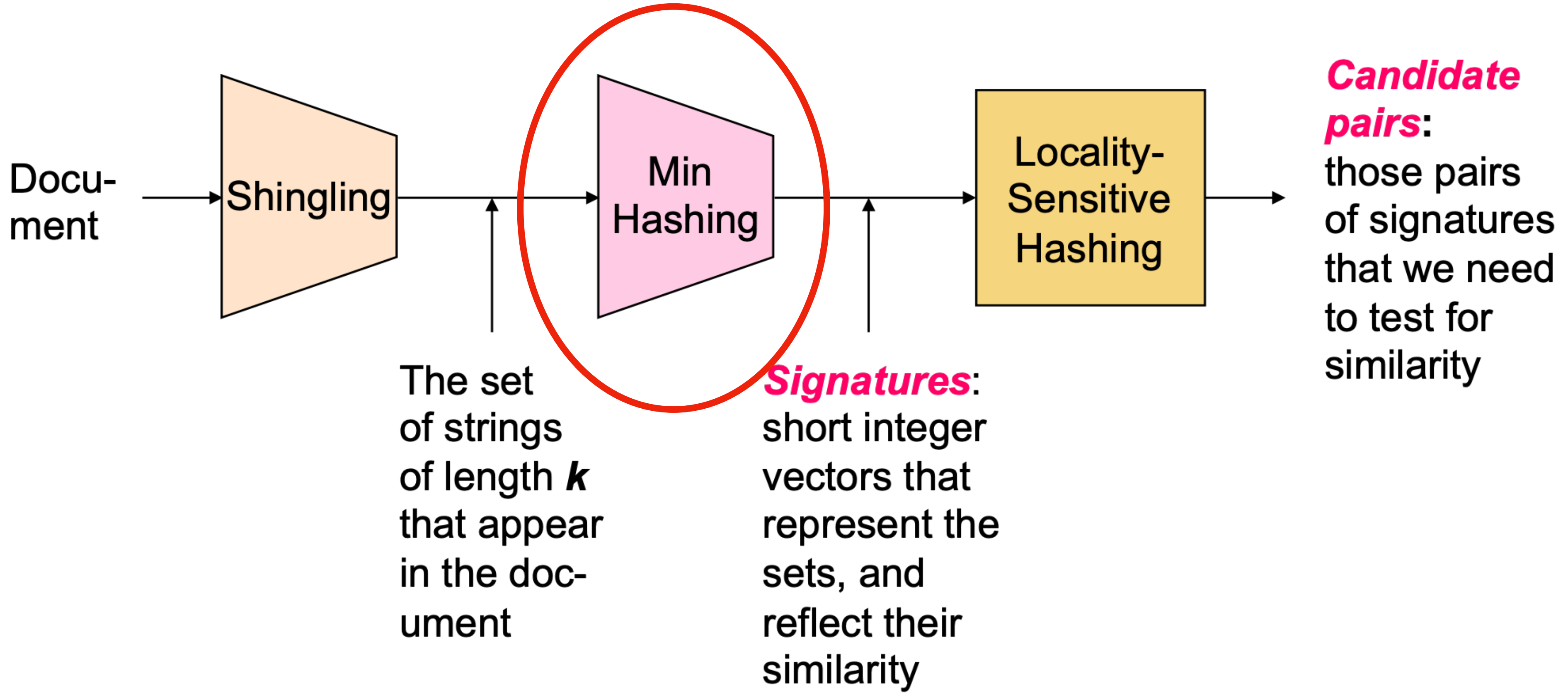
- An ad might have the simple text “Buy Sudzo.” However, a news article with the same idea might read something like “A spokesperson for the Sudzo Corporation revealed today that studies have shown it is good for people to buy Sudzo products.” We have underscored the stop words. Please list the shingles in the ad and the news article.

Answer

- Ad: None
- News article: A spokesperson for, for the Sudzo, the Sudzo Corporation, that studies have , have shown it ….

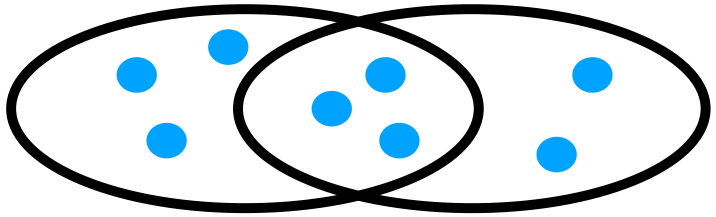
- Suppose we need to find near-duplicate documents among N = 1 million documents
  - compute pairwise similarities for every pair of docs
  - how to compare a pair of set?

Locality-Sensitive Hashing


Docu-
ment
Shingling
Min
Hashing
Locality-
Sensitive
Hashing
The set of strings of length ***k*** that appear in the doc-ument
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

Set Similarity

- Jaccard similarity/distance
  - Jaccard similarity of two sets is the size of their intersection divided by the size of their union:
  - $sim(C_1, C_2) = |C_1 \cap C_2| / |C_1 \cup C_2|$
  - Jaccard distance: $d(C_1, C_2) = 1 - |C_1 \cap C_2| / |C_1 \cup C_2|$



3 in intersection

8 in union

Jaccard similarity= 3/8

Jaccard distance = 5/8

Motivation for Minhash

- Suppose we need to find near-duplicate documents among N = 1 million documents
  - compute pairwise similarities for every pair of docs
  - $N(N-1)/2 \approx 5 * 10^{11}$ comparisons
  - At 10^5 secs/day and 10^6 comparisons/sec, it would take 5 days. **Too long**!
- Formulate the problem as **finding subsets that have significant intersection**
  - encode sets using 0/1 (bit, boolean) vectors
  - interpret set intersection as **bitwise AND**, and set union as **bitwise OR**
  - e.g., C1=10111, C2=10011; d(C1, C2)=?

Boolean Matrices

- Rows = elements (singles)
- Columns = sets (documents)
  - 1 in row **i** and column **j** if and only if **i** is a member of **j**
  - column similarity is the **Jaccard similarity** of the corresponding sets (rows with value 1)
  - sparse matrix!
- E.g., sim(C1, C2) = ?
  - size of intersection = 3; size of union = 6; 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

Finding Similar Columns

- So far:
  - documents -> sets of shingles
  - represent sets as boolean vectors in a matrix
- **Goal:** finding similar columns while computing small signatures
  - Naive approach:
    - **signatures of columns:** small summaries of columns
    - **relate similarities of signatures and columns:** examine pairs of signatures to find similar columns
  - Comparing all pairs may take too much time!

Hashing Columns (Signatures)

- Idea: `hash’ each column C to a small **signature** h(C) s.t.:
  - h(C) is small enough that the signature fits in RAM
  - sim(C1, C2) is the same as the `similarity’ of signatures h(C1) and h(C2)
- Goal: **find a hash function h(·)** s.t.:
  - 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) ≠ h(C2)
- Hash docs into **buckets**. Expect that `most’ pairs of near duplicate docs hash into the same bucket!

Min-Hashing

- Clearly, the hash function depends on the similarity metric: **not all similarity metrics have a suitable hash function**
- For **Jaccard similarity**, we have a suitable hash function:
- **Min-hashing**
  - rows of the boolean matrix permuted under random permutation p
  - 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

Min-Hashing

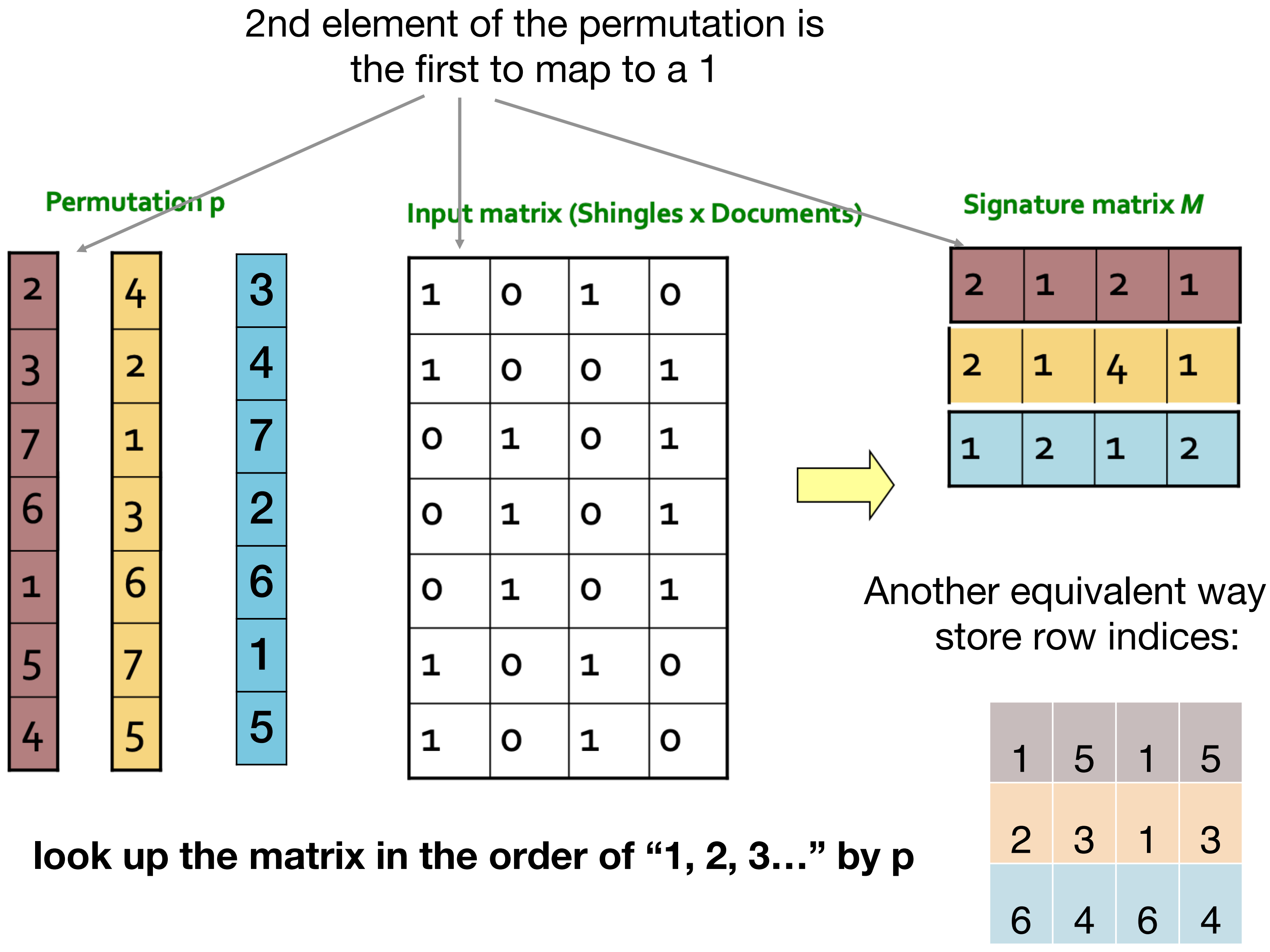
- To minhash a set represented by a matrix:
  - pick a permutation of rows p
  - compute the minhash value of any column as **the number of the first row**, in order p, in which the column **has a 1**

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

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

p：beadc **h(S1) = a， h(S2) = c**

Min-Hashing Example


2nd element of the permutation is the first to map to a 1
Permutation p
Input matrix (Shingles x Documents)
Signature matrix M
Another equivalent way to store row indices:

Permutation p:

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

Another equivalent way to store row indices:

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

look up the matrix in the order of “1, 2, 3…” by p

The Minhash Property

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

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

one of the two columns had to have 1 at position y

Four Types of Rows

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

	C1	C2
A	1	1
B	1	0
C	0	1
D	0	0

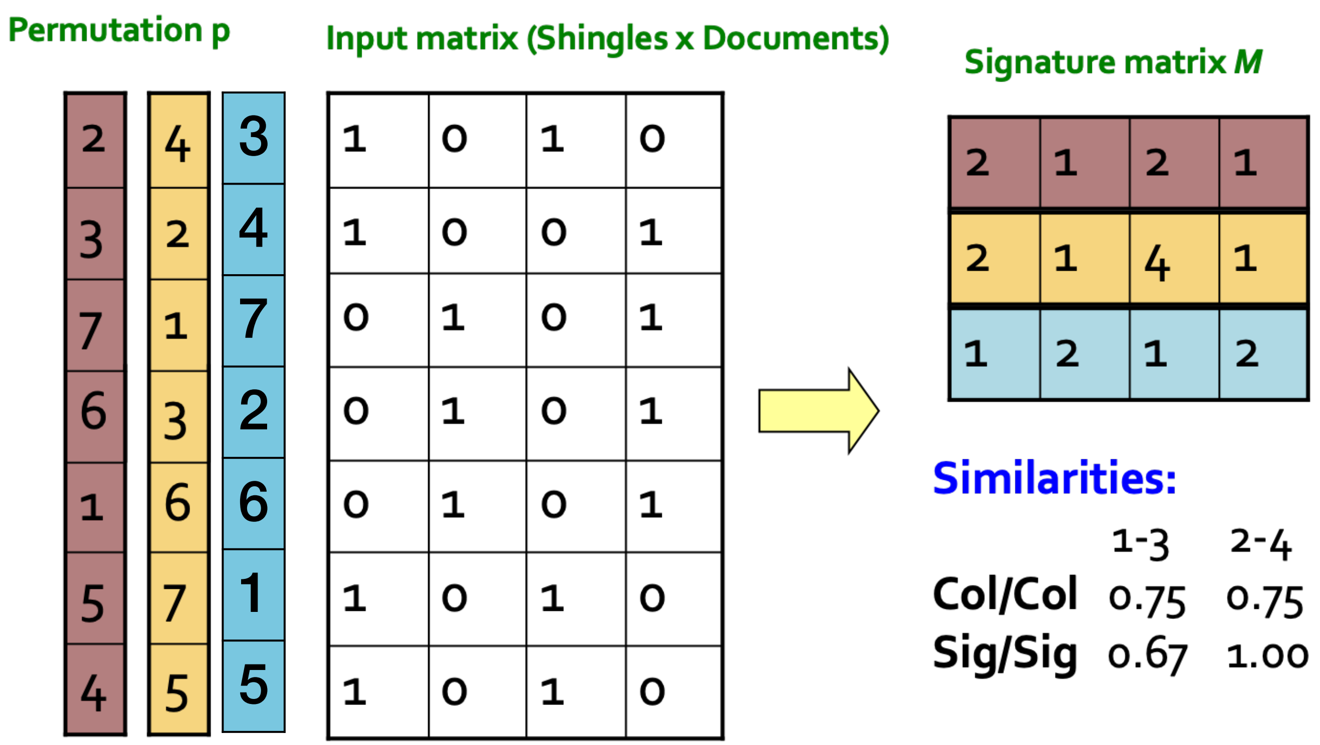
- a = # rows of type A, etc.
  - $\text{sim}(C_1, C_2) = a/(a + b + c)$
  - $\Pr[h(C_1) = h(C_2)] = \text{Sim}(C_1, C_2)$
- Look down the cols C_1 and C_2 until we see a 1
- If it's a type-A row, then $h(C_1) = h(C_2)$
- If a type-B or type-C row, then not

Minhash Signatures

- We pick at random some number n of permutations of the rows of M (perhaps 100 permutations): h_1, h_2, …, h_n
- Construct the **minhash signature** for S: [h_1(S), h_2(S), …, h_n(S)]
- Form a **signature matrix**
  - has the same number of columns as M but only n rows
  - much smaller than M!
  - the expected number of rows in which two columns agree equals the Jaccard similarity of the corresponding sets
  - more rows in the signature matrix, the smaller the error of estimating the Jaccard similarity

Similarity for Signatures

- We have 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
- Because of Min-Hash property, the similarity of columns is the same as the expected similarity of their signatures. What are Jaccard similarities and signature similarities for 1 vs 3, 2 vs 4, 1 vs 2, 3 vs 4?



Permutation p

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

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

Minhash Signatures

- Pick k=100 random permutations of the rows
- Think of sig(C) as a column vector
- sig(C)[i] = according to the i-th permutation, the index of the first row that has a 1 in column C
    - **sig(C)[i] = min (p_i(C))**
    - the signature of document C is small ~100 bytes!
- We **compressed** long bit vectors into short signatures!
- Pick a random permutation of millions or billions of rows is **time-consuming**!
- Simulate the effect of a random permutation by **a random hash function** that maps row numbers to buckets

Implementation

- **Row hashing** instead of permutation!
  - pick k = 100 hash functions $\boldsymbol{k_i}$
  - ordering under $\boldsymbol{k_i}$ gives a random row permutation
  - for each column ***C*** and hash-func. $\boldsymbol{k_i}$, keep a "slot" for the min-hash value
  - initialize all **sig(C)[i] =** ∞
  - scan rows looking for 1s
    - suppose row ***j*** has 1 in column ***C***
    - for each $\boldsymbol{k_i}$:
      - If $k_i(j) < \text{sig}(C)[i]$, then $\text{sig}(C)[i] \leftarrow k_i(j)$

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 are random integers

p is a prime number (p > N)

Question

- We consider two hash functions for the following matrix. Please estimate the Jaccard similarities of 1 vs 2, 1 vs 3, and 1 vs 4 according to the 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

Answer

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

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

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

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

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

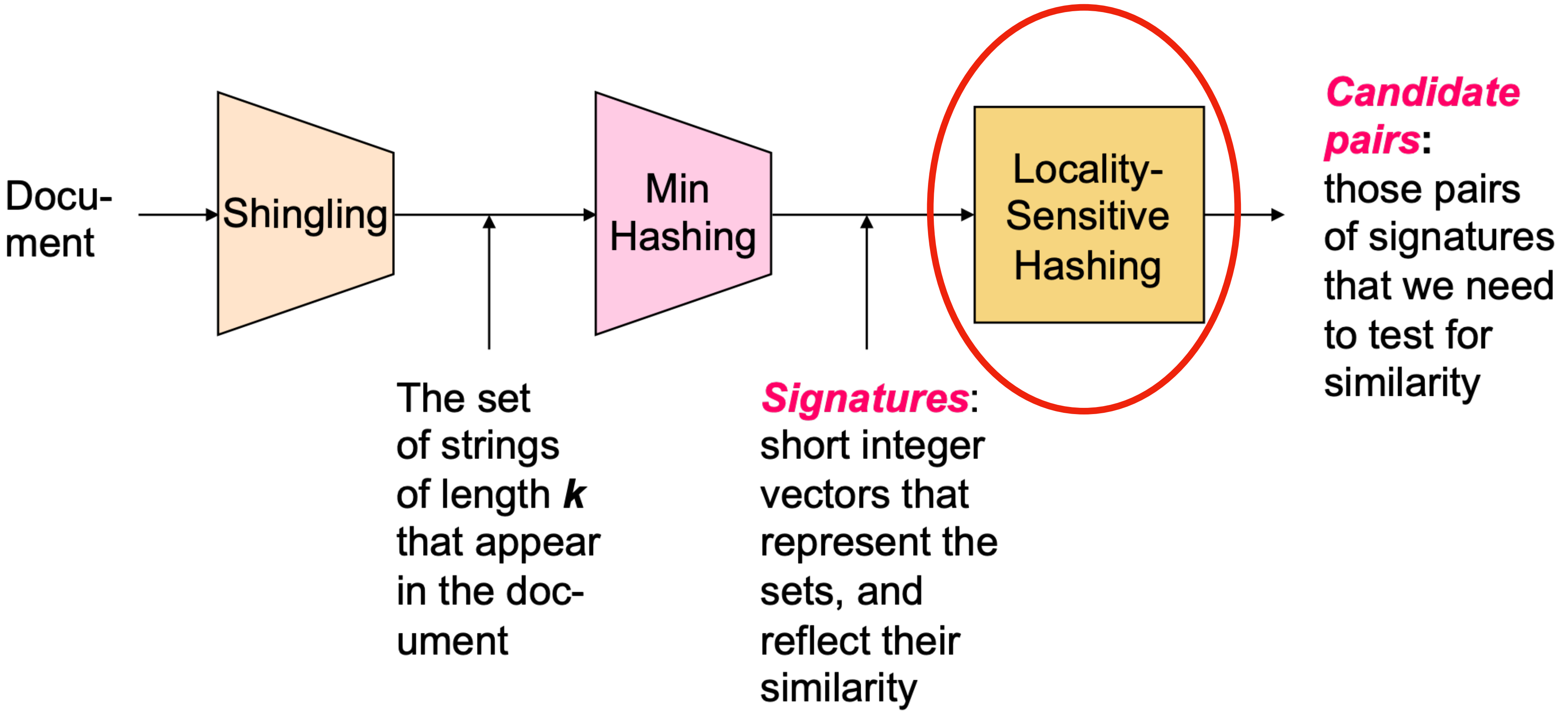
	S_1	S_2	S_3	S_4
h_1	1	3	2	1
h_2	0	2	0	0

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

sim(S1,S2) = 0
sim(S1,S3) = 1/2
sim(S1,S4) = 1

In fact,
sim(S1,S2) = 0
sim(S1,S3) = 1/4
sim(S1,S4) = 2/3

Locality-Sensitive Hashing



Locality-Sensitive Hashing

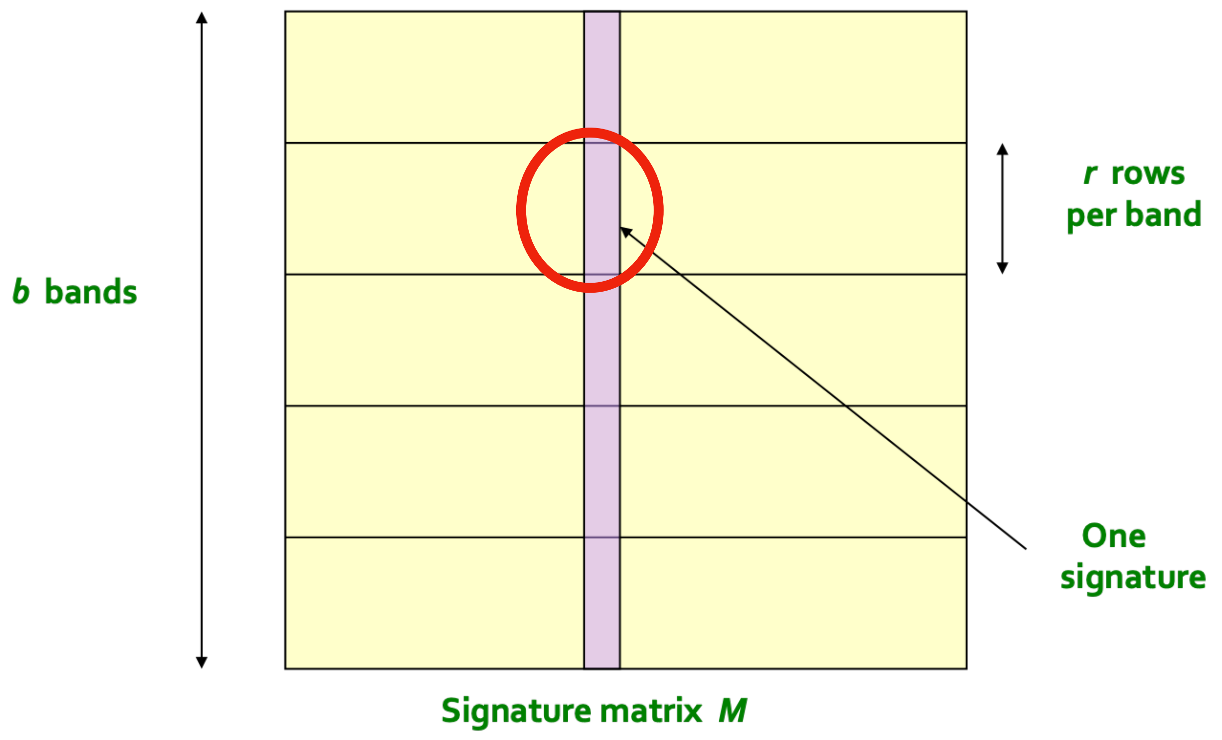
- Suppose we have 1,000,000 documents, and we use signatures of length 250. How many bytes do we need for the signature of one document?
- If we use 1000 bytes per document for the signatures, how many bytes in total do we need to store all signature matrices?
- 10^9 bytes (1 GB) are needed! How many pairs of documents we need to compare? $C^2_{1,000,000}$ **Too many! Can we do better?**
- We only need to focus on pairs that are likely to be similar, without investigating every pair
- Consider any pair that hashed to the same bucket to be a **candidate pair**

Locality-Sensitive Hashing

- **Goal**: Find documents with Jaccard similarity at least **s** (e.g. **s**=0.8)
- **Idea**: Use a function f(**x**, **y**) that tells whether **x** and **y** is a **candidate pair**— a pair of elements whose similarity must be evaluated
- 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**
  - Pick a similarity threshold **s** (0 < **s** < 1)
  - Columns **x** and **y** of **M** 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 fraction **s** values of **i**
  - We expect **x** and **y** to have the same Jaccard similarity as their signatures

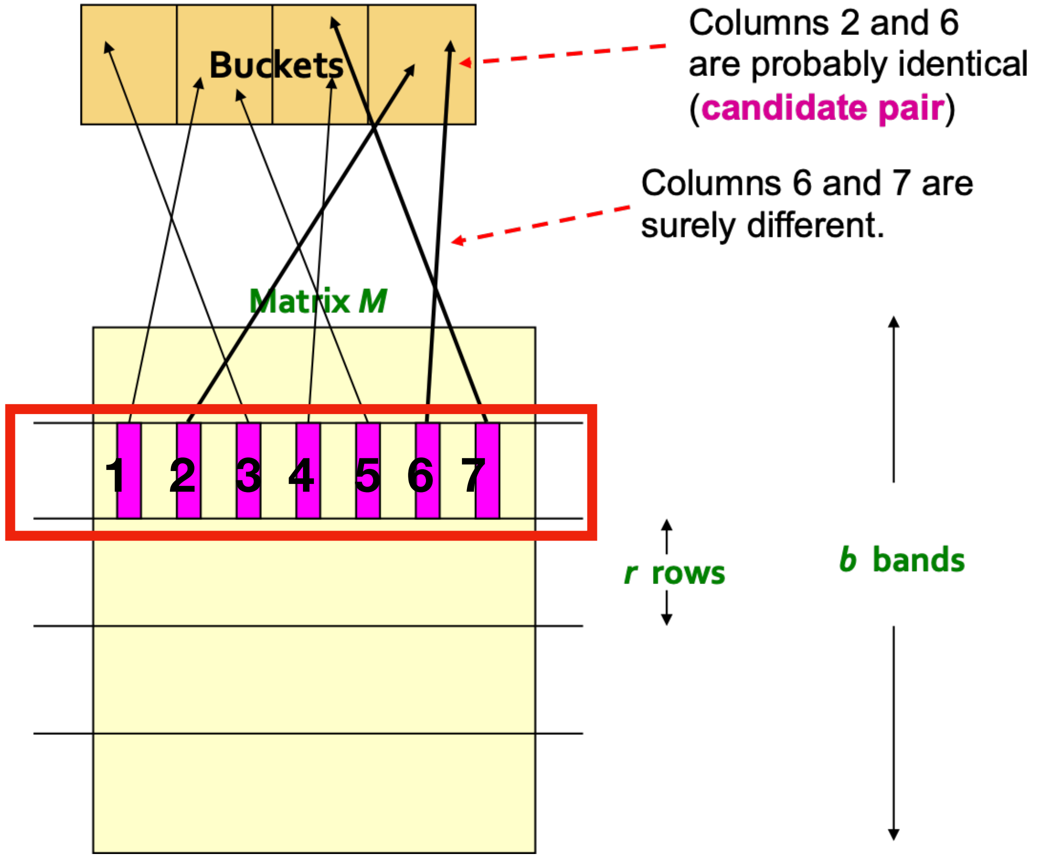
LSH for Min-Hash

- Idea: Hash columns of signature matrix **M** several times
- Arrange that **similar columns** are likely to **hash to the same bucket**, with high probability
- **Candidate pairs** are those that hash to the same bucket



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 with **k** buckets
- **Candidate column pairs** are those that hash to the same bucket for >= 1 band
- Tune **b** and **r** to catch most similar pairs, but few non-similar pairs


Buckets
Columns 2 and 6 are probably identical (candidate pair)
Columns 6 and 7 are surely different.
Matrix M
1 2 3 4 5 6 7
r rows
b bands

Assume: there are enough buckets that columns are unlikely to hash to the same bucket unless they are identical in a band; “same bucket” means “identical in that band”

Examples

- Suppose
  - 100,000 columns of **M** (100k docs)
  - Signatures of 100 integers (rows), 40MB in total
  - Choose **b**=20 bands for **r**=5 integers/band
  - Goal: Find pairs of documents that are at least **s**=0.8 similar
- If sim(C1, C2)>=**s**, we want C1, C2 to be a candidate pair — want to hash them to **at least 1 common bucket** (at least 1 band identical)
- Prob. C1, C2 identical in one particular band? $(0.8)^5=0.328$
- Prob. C1, C2 are not similar in all of the 20 bands? $(1-0.328)^{20}=0.00035$
  - about 1/3000th of the 80%-similar column pairs are **false negatives**
  - we would find 99.965% pairs of truly similar documents

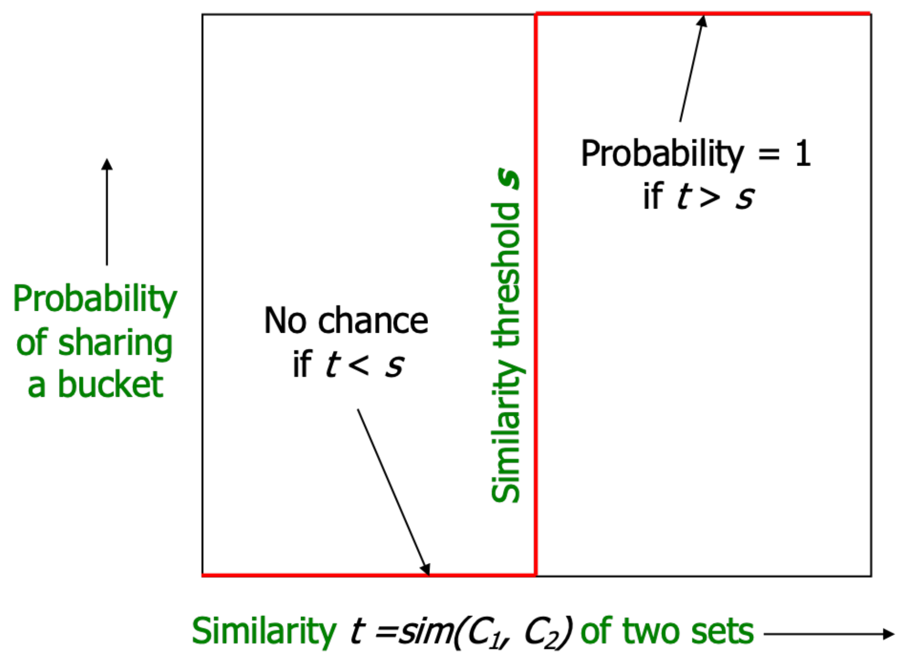
Examples

- Suppose
  - 100,000 columns of **M** (100k docs)
  - Signatures of 100 integers (rows), 40Mb in total
  - Choose **b**=20 bands for **r**=5 integers/band
  - Goal: Find pairs of documents that are at least **s**=0.3 similar
- If sim(C1, C2)<**s**, we want C1, C2 to hash to **NO common buckets** (all bands should be different)
- Prob. C1, C2 identical in one particular band? $(0.3)^5=0.00243$
- Prob. C1, C2 are identical in at least 1 of 20 bands? $1-(1-0.00243)^{20}=0.0474$
  - about 4.74% pairs of docs with similarity 30% end up becoming candidate pairs (**false positives**). We will have to examine them but their similarity <**s**

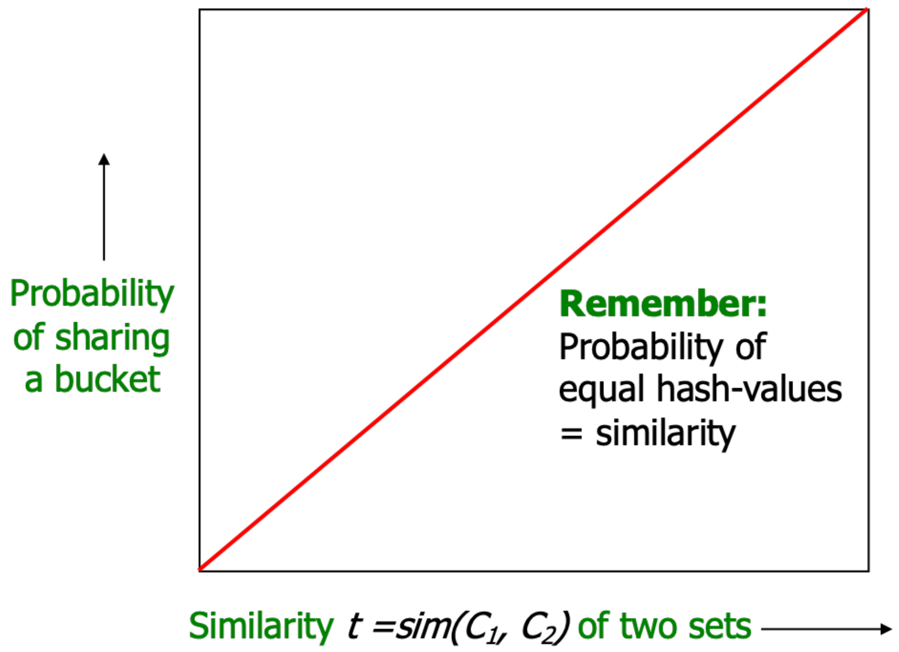
Tradeoff

- To balance false positives/negatives, we should pick:
  - The number of Min-Hashes (rows of **M**)
  - The number of bands **b**
  - The number of rows **r** per band

Ideal:

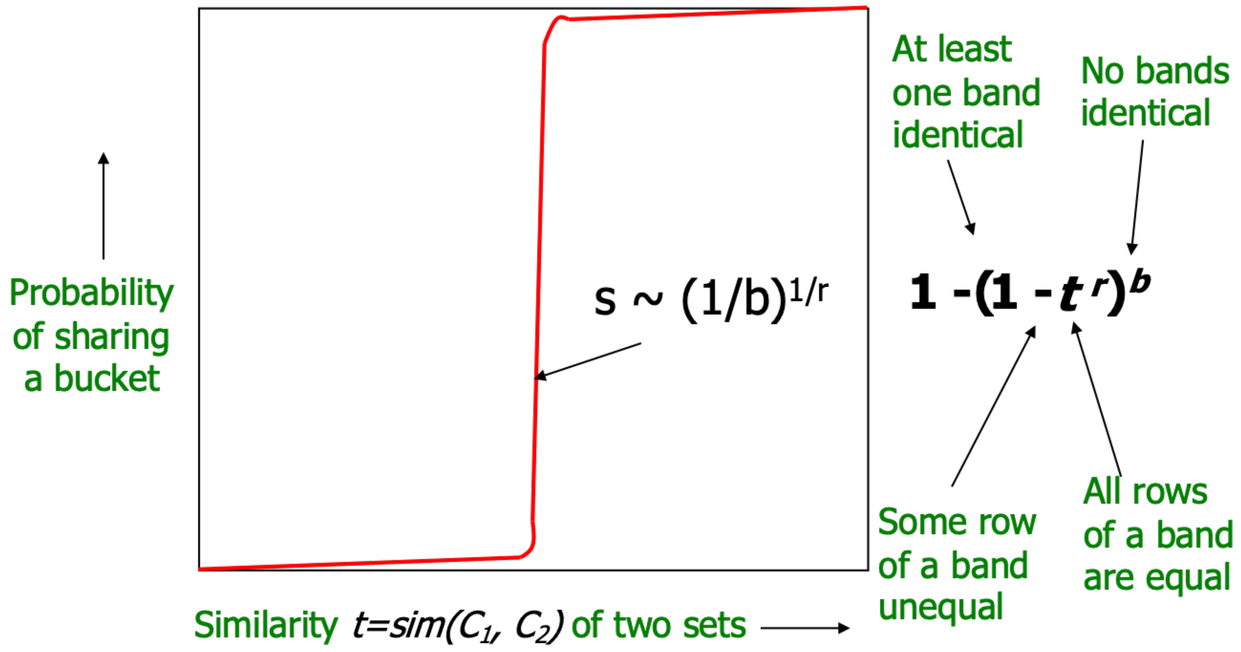


1 Band per Row:



Choices of b & r

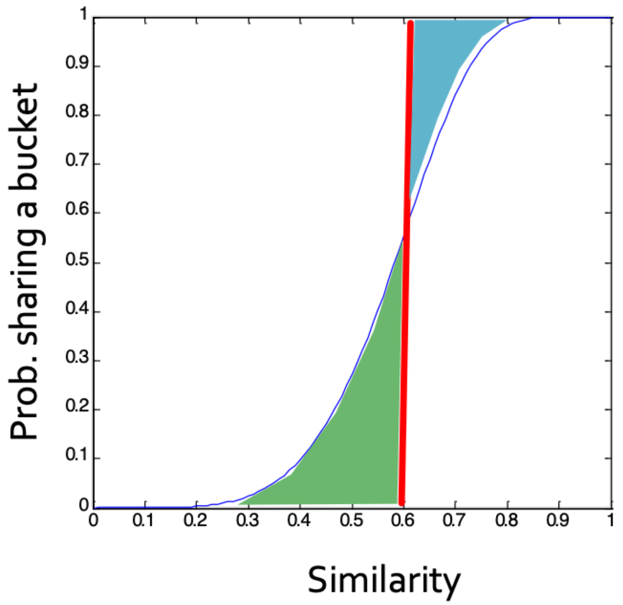
- Columns C1 and C2 have similarity **t**
- Pick any band (**r** rows)
  - Prob. that all rows in band equal = $\mathbf{t^r}$
  - Prob. that some row in band unequal = $\mathbf{1- t^r}$
- Prob. that no band identical = $\mathbf{(1- t^r)^b}$
- Prob. that at least 1 band identical = $\mathbf{1- (1- t^r)^b}$



Choices of b & r

- Tune **M, b, r** 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**

s	$1-(1-s^r)^b$
0.2	0.006
0.3	0.047
0.4	0.186
0.5	0.470
0.6	0.802
0.7	0.975
0.8	0.9996



50 hash functions
(b=10, r=5)

Blue area: False Negative rate
Green area: False Positive rate

Combining the Techniques

- Find the set of candidate pairs for similar documents and then discovering the truly similar documents among them

1. Pick a value of k and construct from each document the set of **k-shingles**
2. Sort the document-shingle pairs to order them by shingle
3. Pick a length n for the minhash signatures. Compute the **minhash signatures** for all the documents (sorted lists)
4. Choose a threshold **t** that defines how similar documents have to be in order for them to be regarded as “similar pair.” Pick a number of of bands **b** and a number of rows **r** such that **br = n**, and the threshold **t** is approximately $\mathbf{(1/b)^{1/r}}$.
5. Construct candidate pairs by applying the **LSH** technique
6. Examine each candidate pair’s signatures and determine whether the fraction of components in which they agree is at least **t**

Summary

- Locality-Sensitive Hashing
  - **Shingling**: convert documents to sets
  - **Minhashing**: convert large sets to short **signatures**, preserving similarity $\Pr[h_\pi(C_1) = h_\pi(C_2)] = sim(C_1, C_2)$
  - **Locality-Sensitive Hashing:** hash to find **candidate pairs** of similarity >= s