بسم الله الرحمن الرحیم

جلسه بیست و پنجم

خلاصه‌سازی برای مه‌داده

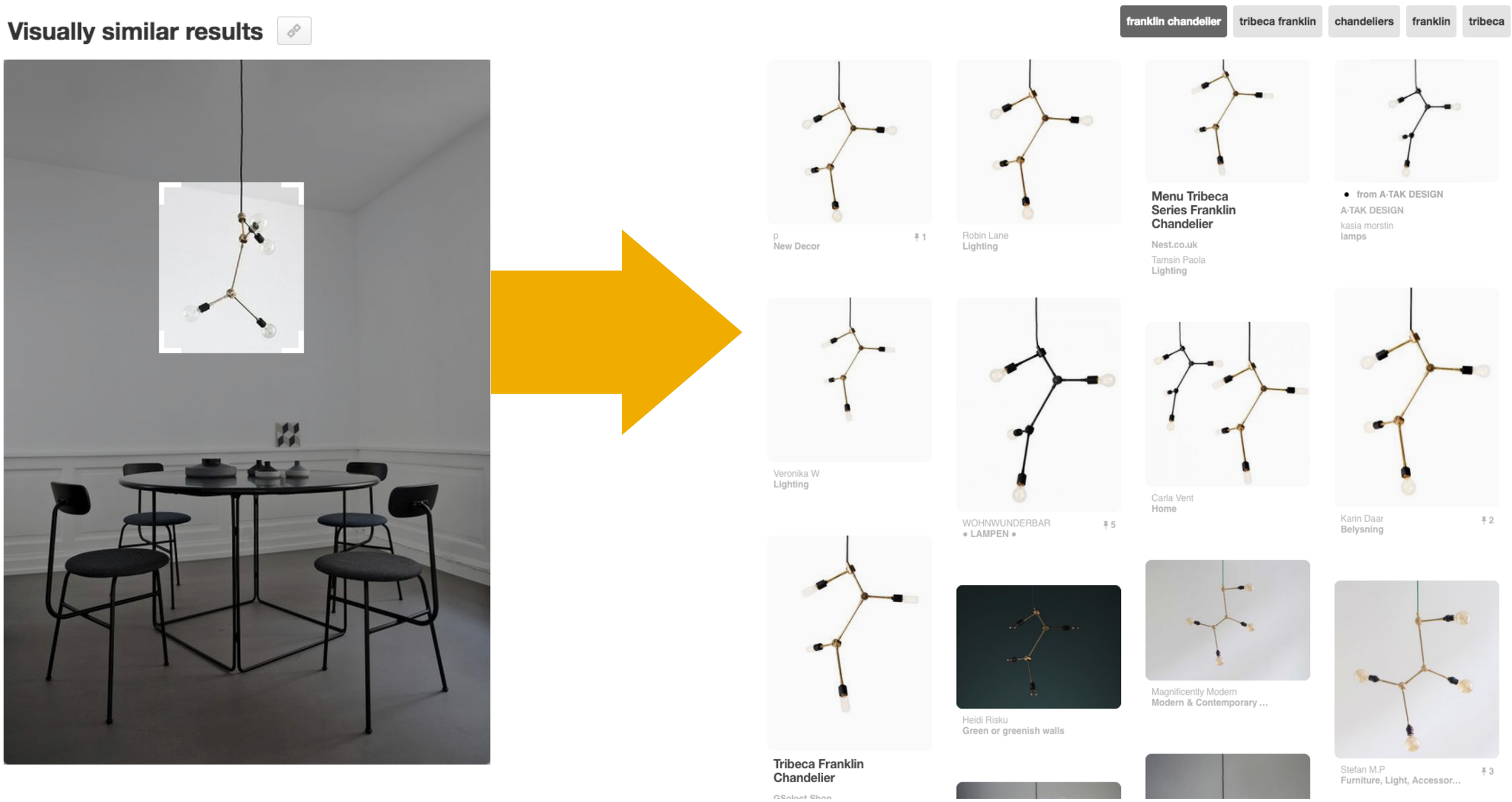
درهم‌سازی حساس به محلیت

Finding Similar Items: Locality Sensitive Hashing

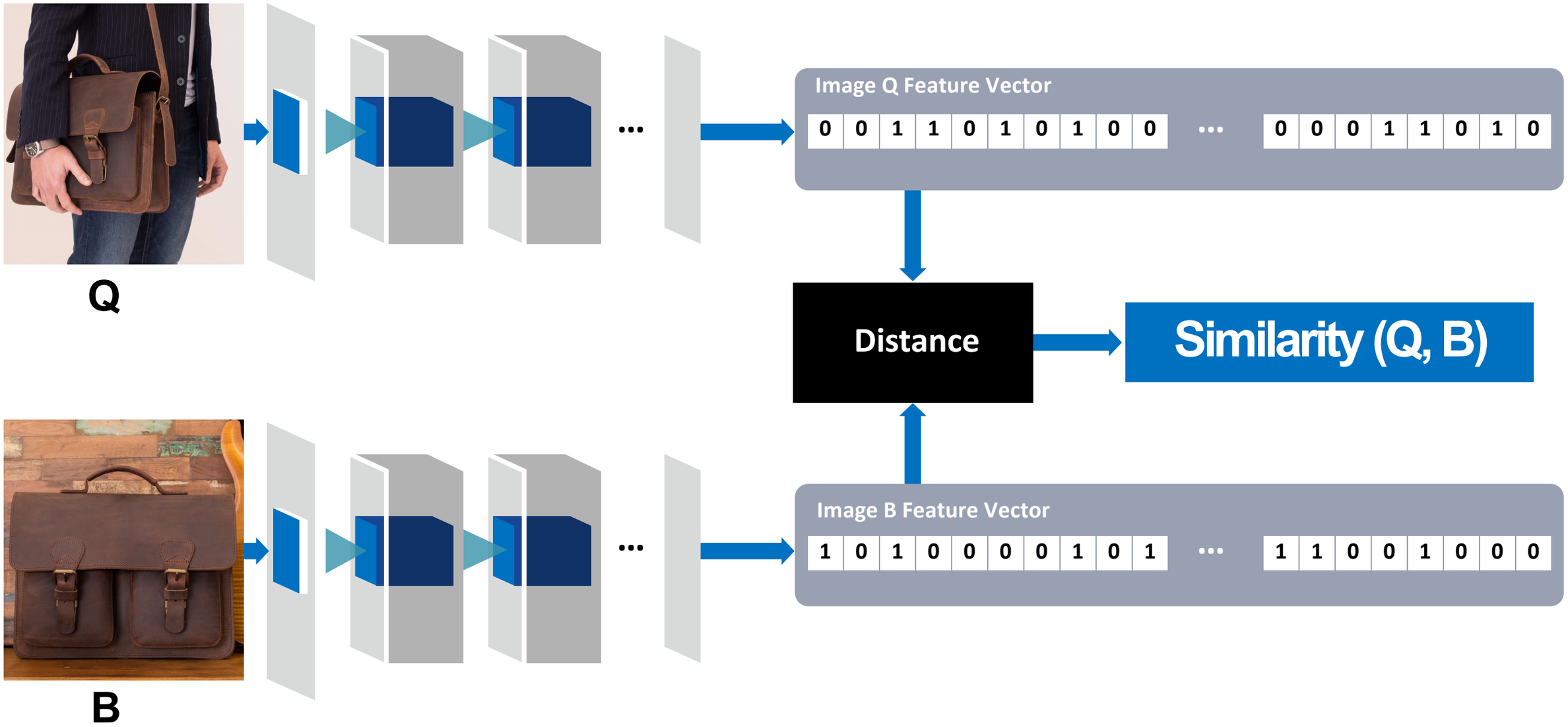
CS246: Mining Massive Datasets
Jure Leskovec, Stanford University
http://cs246.stanford.edu

Pinterest Visual Search

Given a **query image patch**, find similar images



How does it work?



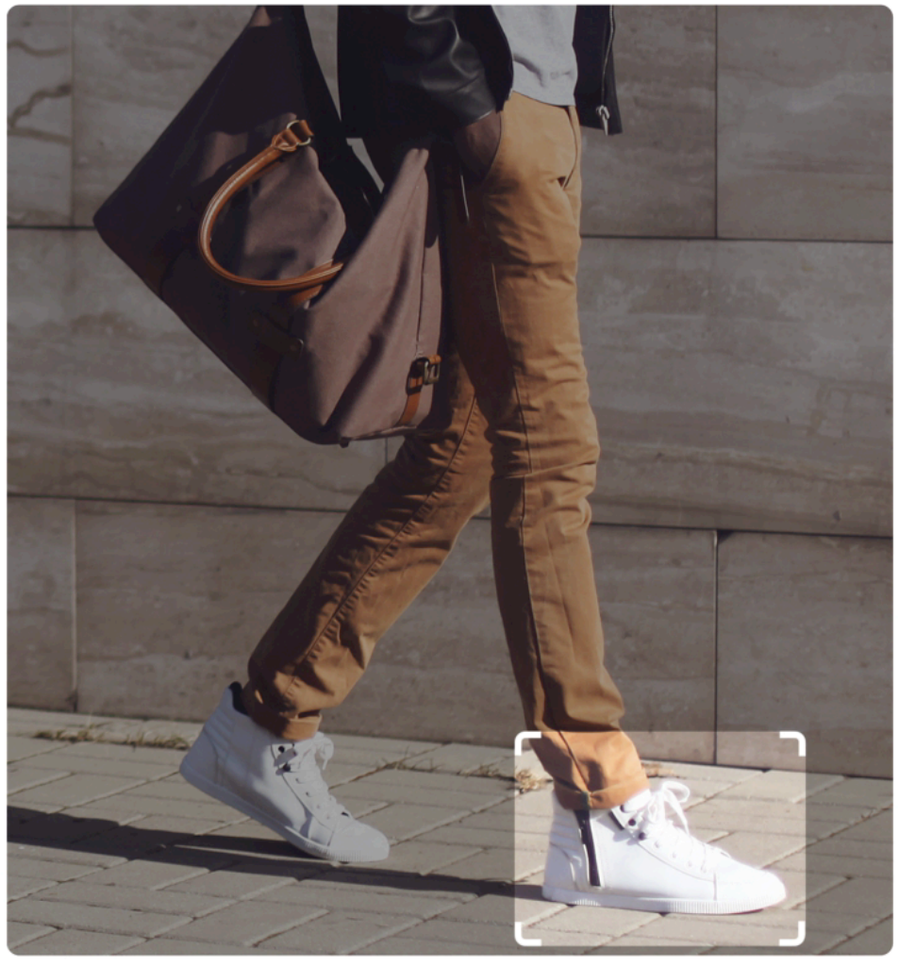
- Collect billions of images
- Determine feature vector for each image (4k dim)
- **Given a query Q, find nearest neighbors FAST**

How does it work?

Q

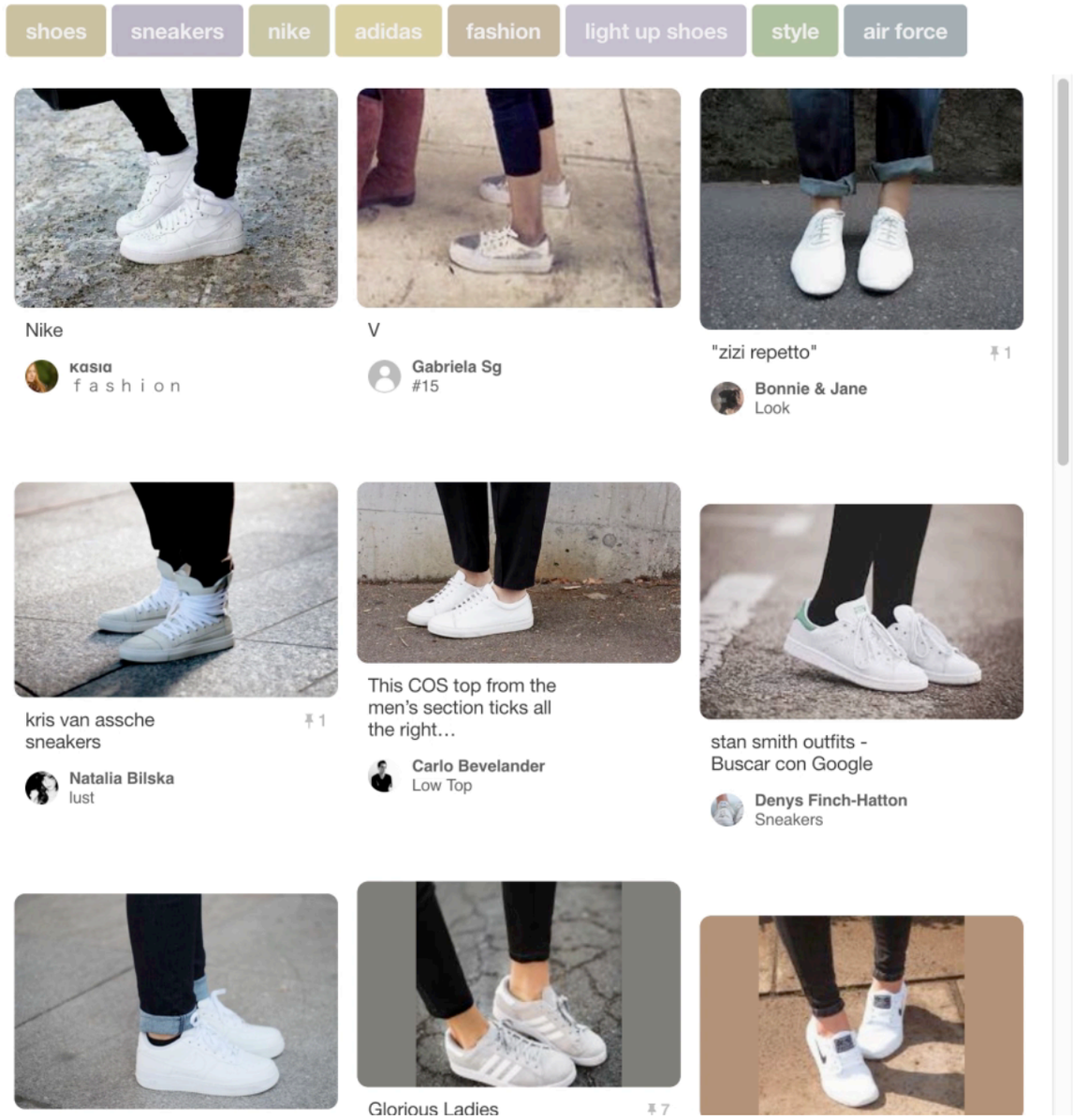
Nearest neighbor query
in the embedding space

Application: Visual Search



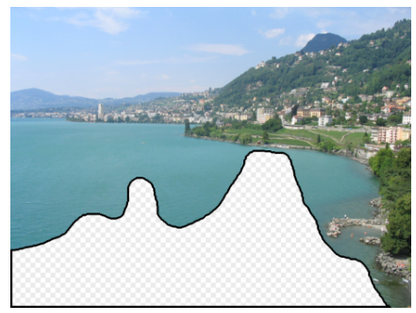
Q

Visually similar results



A Common Metaphor

- **Many problems can be expressed as finding "similar" sets:**
  - **Find near-neighbors in high-dimensional space**
- **Examples:**
  - **Pages with similar words**
    - For duplicate detection, classification by topic
  - **Customers who purchased similar products**
    - Products with similar customer sets
  - **Images with similar features**
    - Image completion
  - **Recommendations and search**



Problem for today's lecture

- **Given: High dimensional data points $x_1, x_2, \ldots$**
  - **For example:** Image is a long vector of pixel colors
- **And some distance function $d(x_1, x_2)$**
  - which quantifies the "distance" between x_1 and x_2
- **Goal:** Find **all pairs of data points (x_i, x_j)** that are within distance threshold $d(x_i, x_j) \leq s$
  - **Or (an easier problem): Find all x_j s.t. $d(q, x_j) \leq s$**
- **Note:** Naïve solution would take $O(N^2)$ where N is the number of data points
- **MAGIC: This can be done in $O(N)$!! How??**

Problem for today's lecture

- **Given: High dimensional data points $x_1, x_2, \ldots$**
  - **For example:** Image is a long vector of pixel colors
- **And some distance function $d(x_1, x_2)$**
  - which quantifies the "distance" between x_1 and x_2
- **Goal:** Find **all pairs of data points (x_i, x_j)** that are within distance threshold $d(x_i, x_j) \leq s$
  - **Or (an easier problem): Find all x_j s.t. $d(q, x_j) \leq s$**
- **Note:** Naïve solution would take $O(N^2)$ where N is the number of data points
- **MAGIC: This can be done in $O(N)$!! How??**

Problem for today's lecture

- **Given: High dimensional data points $x_1, x_2, \ldots$**
  - **For example:** Image is a long vector of pixel colors
- **And some distance function $d(x_1, x_2)$**
  - which quantifies the "distance" between x_1 and x_2
- **Goal:** Find **all pairs of data points (x_i, x_j)** that are within distance threshold $d(x_i, x_j) \leq s$
  - **Or (an easier problem): Find all x_j s.t. $d(q, x_j) \leq s$**
- **Note:** Naïve solution would take $O(N^2)$ where N is the number of data points
- **MAGIC: This can be done in $O(N)$!! How??**

Problem for today's lecture

- **Given: High dimensional data points $x_1, x_2, \ldots$**
  - **For example:** Image is a long vector of pixel colors
- **And some distance function $d(x_1, x_2)$**
  - which quantifies the "distance" between x_1 and x_2
- **Goal:** Find **all pairs of data points (x_i, x_j)** that are within distance threshold $d(x_i, x_j) \leq s$
  - **Or (an easier problem): Find all x_j s.t. $d(q, x_j) \leq s$**
- **Note:** Naïve solution would take $O(N^2)$ where N is the number of data points
- **MAGIC: This can be done in $O(N)$!! How??**

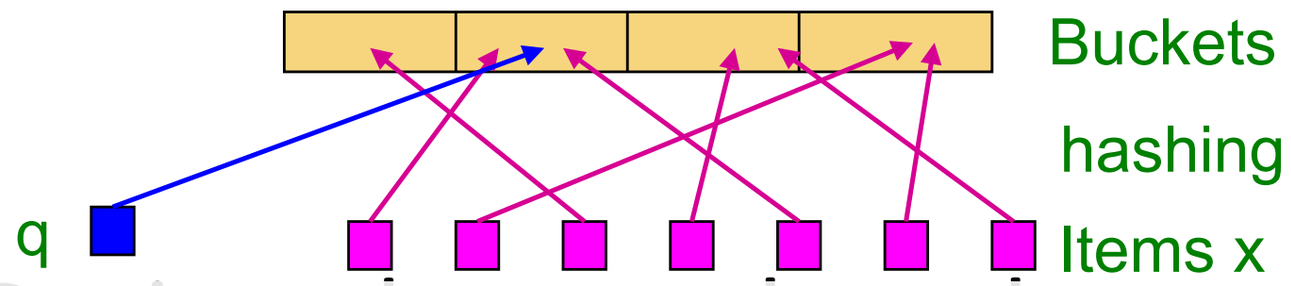
LSH: The Bigfoot of CS

- LSH is really a family of related techniques
- In general, one throws items into buckets using several different "hash functions"
- You examine only those pairs of items that share a bucket for at least one of these hashings

- **Upside:** Designed correctly, only a small fraction of pairs are ever examined
- **Downside:** There are *false negatives* – pairs of similar items that never even get considered

LSH: The Bigfoot of CS

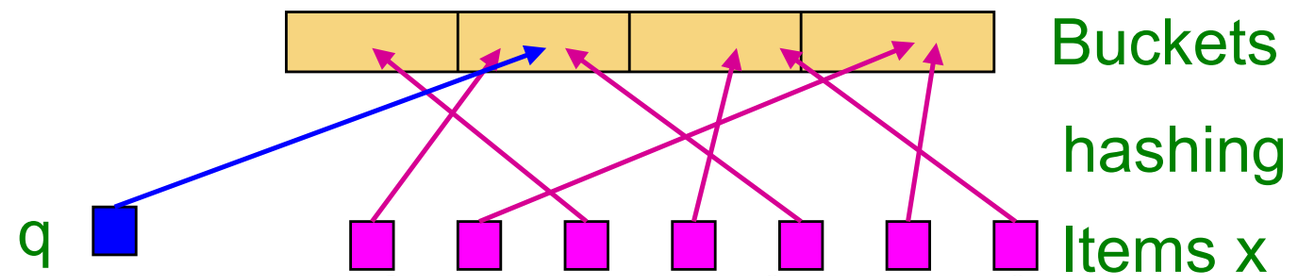
- LSH is really a family of related techniques
- In general, one throws items into buckets using several different "hash functions"
- You examine only those pairs of items that share a bucket for at least one of these hashings



- **Upside:** Designed correctly, only a small fraction of pairs are ever examined
- **Downside:** There are *false negatives* – pairs of similar items that never even get considered

LSH: The Bigfoot of CS

- LSH is really a family of related techniques
- In general, one throws items into buckets using several different "hash functions"
- You examine only those pairs of items that share a bucket for at least one of these hashings



- **Upside:** Designed correctly, only a small fraction of pairs are ever examined
- **Downside:** There are *false negatives* – pairs of similar items that never even get considered

Motivating Application: Finding Similar Documents

مشابهت‌یابی مستندات

- یک میلیون مستند
- مشابهت‌یابی دوبه‌دو
  - N= ۱ میلیون ==> ۵ روز
  - N= ۱۰ میلیون ==> یک سال

3 Essential Steps for Similar Docs

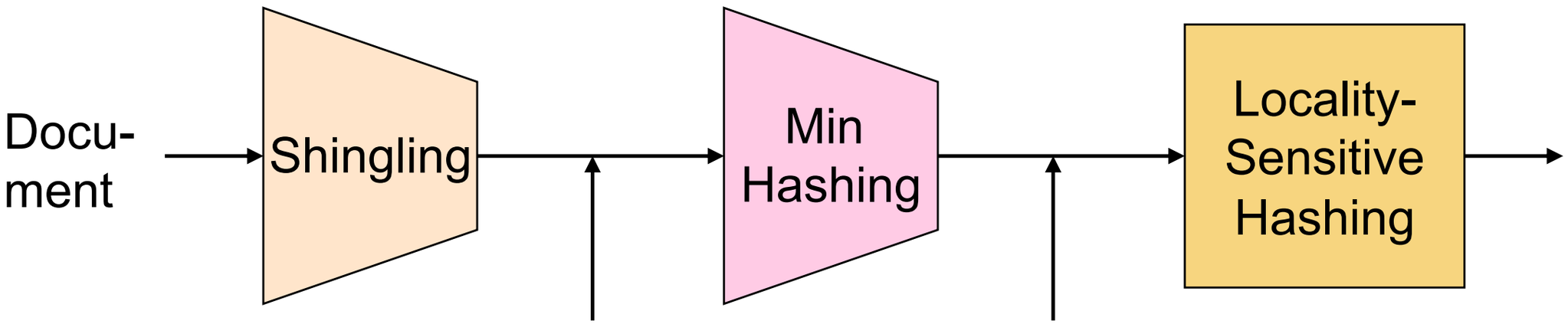
1. ***Shingling:*** Converts a document into a set representation (Boolean vector)

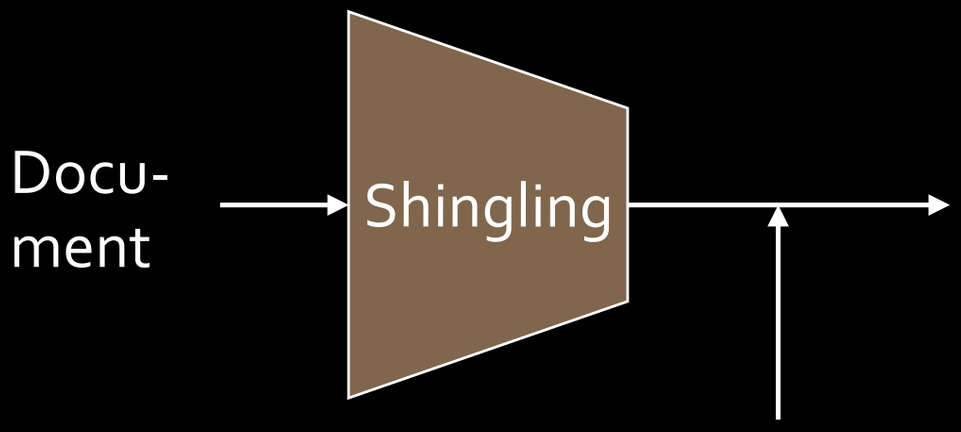
2. ***Min-Hashing:*** Convert large sets to short signatures, while preserving similarity

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

   - **Candidate pairs!**

The Big Picture





Shingling

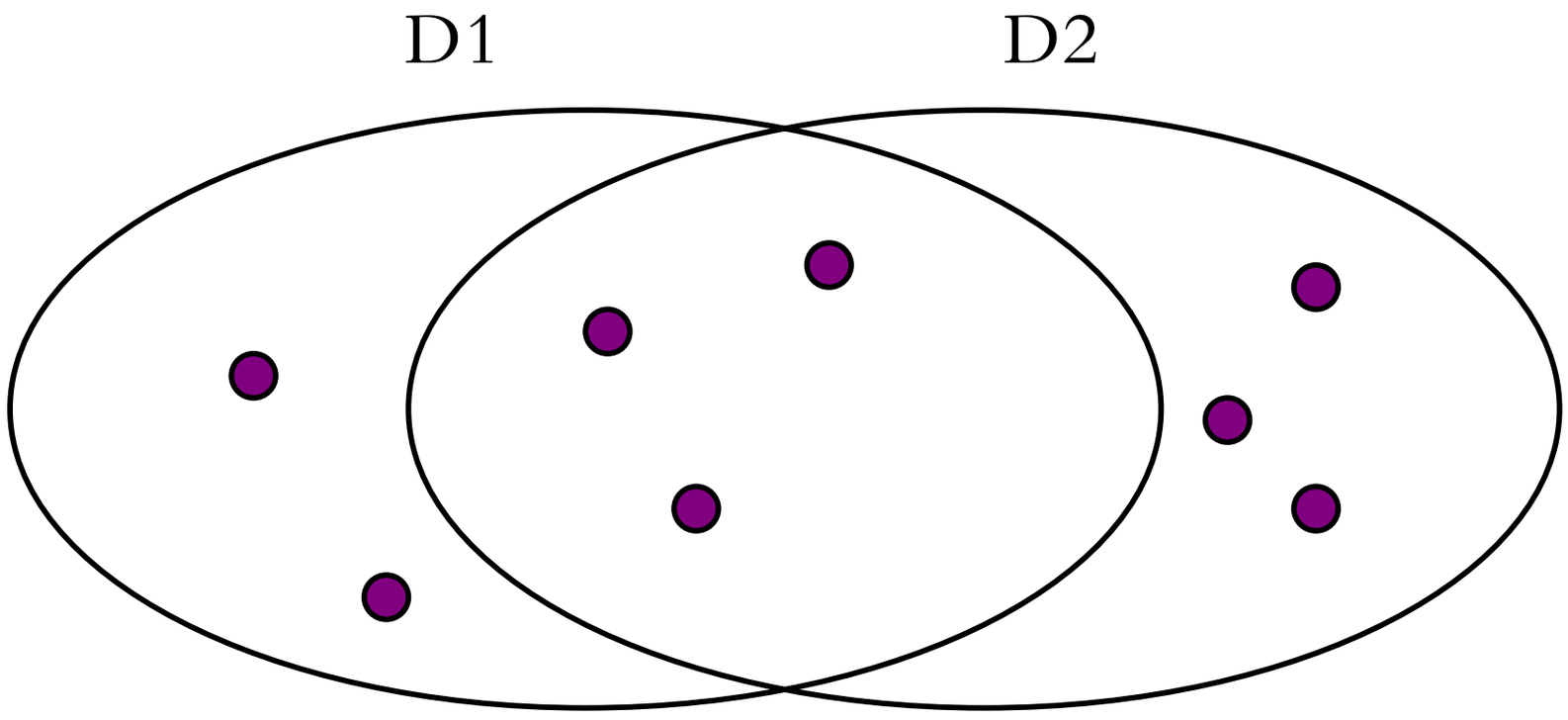
Step 1: ***Shingling:***
Convert a document into a set

مرحله ۱: Shingling

- مستند: کلمه‌ها
- ژنوم: k_ تایی‌های متوالی

- **Example: k=2**; document D_1= abcab
  Set of 2-shingles: $S(D_1)$ = {ab, bc, ca}
  Hash the shingles: $h(D_1)$ = {1, 5, 7}
  - هر مستند = سبدی از واژه‌ها

معیار تشابه دو سند



Jaccard similarity:

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

Documents

Shingles

واژه‌ها —>

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

نگه‌داری:
بسیار حجیم

Documents

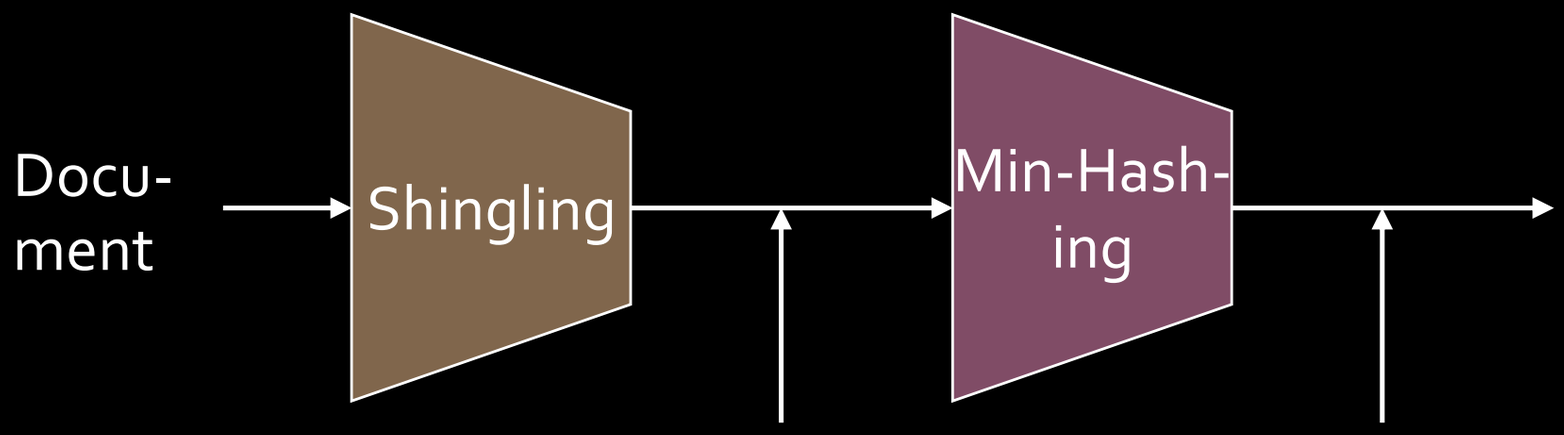
Shingles

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

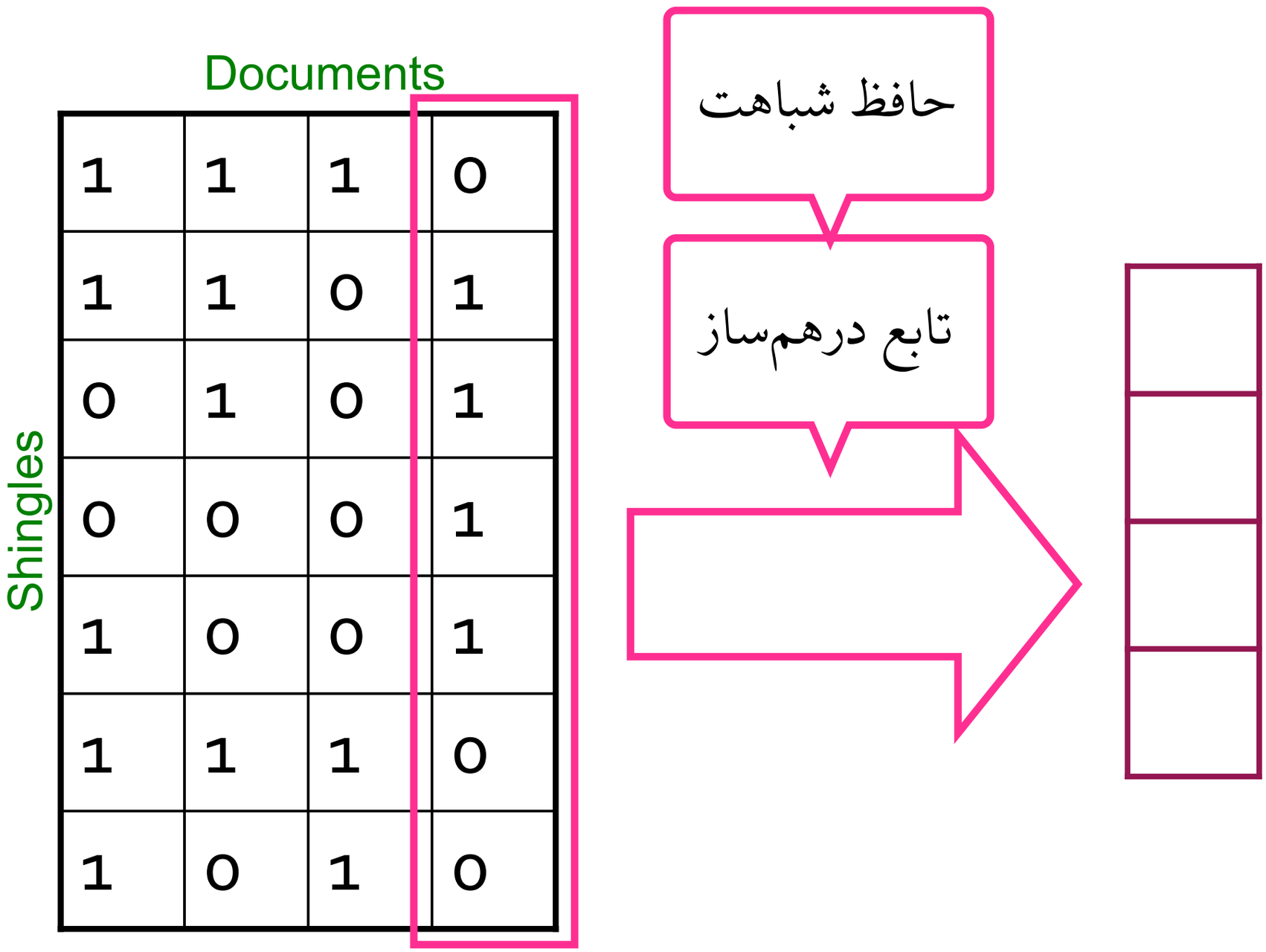
- **Next goal: Find similar columns while computing small signatures**
  - **Similarity of columns == similarity of signatures**

مقایسه دو_به_دو:
بسیار زمان‌بر



Min-Hashing

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



- **Goal: Find a hash function $h(\cdot)$ 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. $h(C_1) \neq h(C_2)$

- **Goal: Find a hash function $h(\cdot)$ 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. $h(C_1) \neq h(C_2)$

تابع درهم‌ساز برای Jackard:
Min-Hashing

۱ـ جایگشت تصادفی

Permutation π	Input matrix (Shingles x Documents)			
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

۱ـ جایگشت تصادفی

۲ـ اولین سطر غیرصفر

C1	C2	C3	C4

Permutation π **Input matrix (Shingles x Documents)**

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

Signature matrix *M*

2	1	2	1

۱ـ جایگشت تصادفی

۲ـ اولین سطر غیرصفر

C1	C2	C3	C4

Permutation π

Input matrix (Shingles x Documents)

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

Signature matrix *M*

2	1	2	1

۳_ چند بار تکرار

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

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

Permutation π

Input matrix (Shingles x Documents)

Signature matrix *M*

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

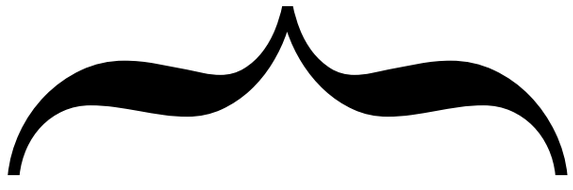
1	0	1	0
1	0	0	1
0	1	0	1
0	1	0	1
0	1	0	1
1	0	1	0
1	0	1	0

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

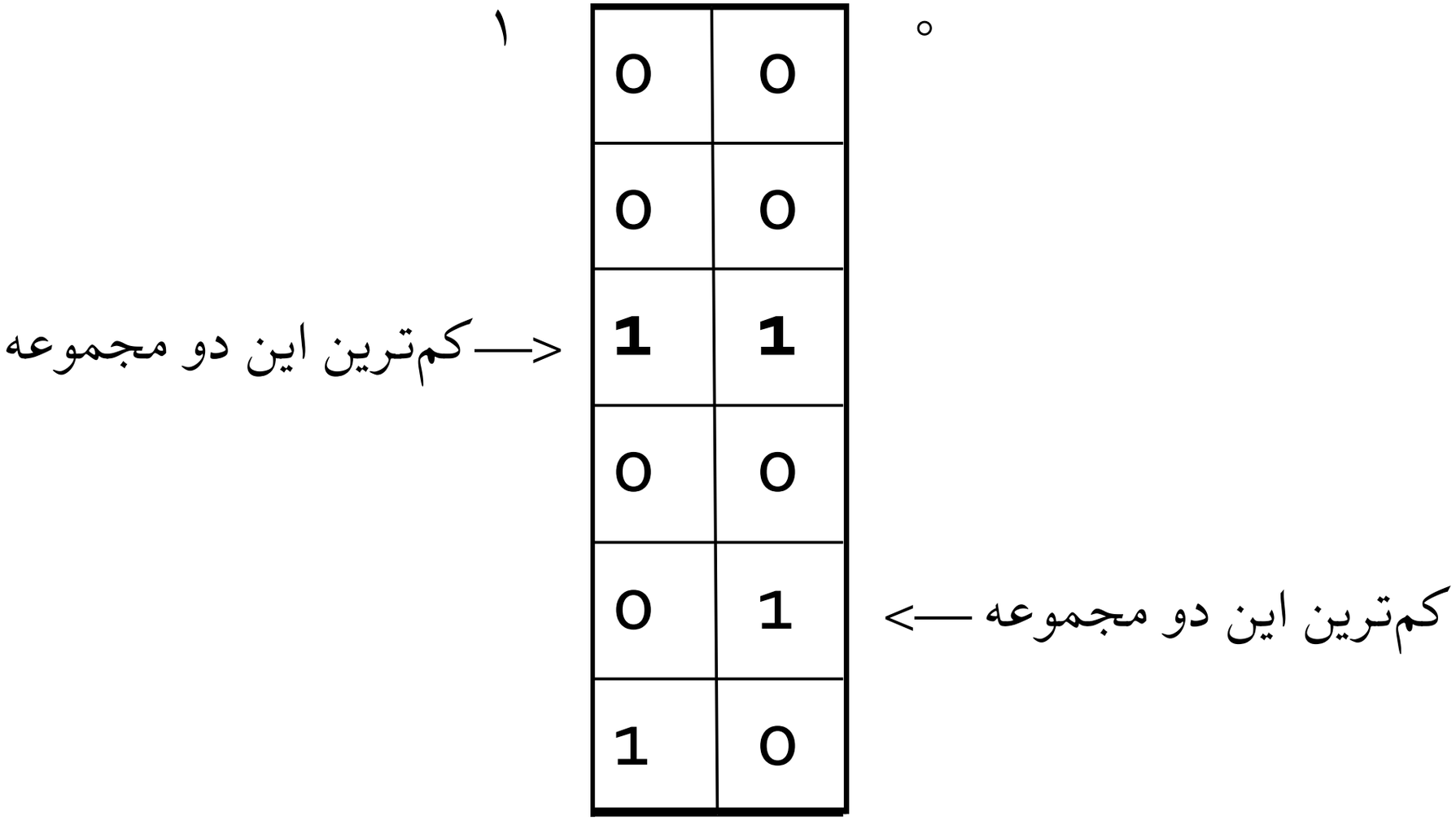
$h_2(3)=1$ *(permutation 2, column 3)*
4th element of the permutation
(row 1) is the first to map to a 1

The Min-Hash Property

- **Choose a random permutation π**
- **Claim:** $\Pr[h_\pi(C_1) = h_\pi(C_2)] = sim(C_1, C_2)$
- **Why?**



$= |C_1 \cap C_2| / |C_1 \cup C_2|$



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	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

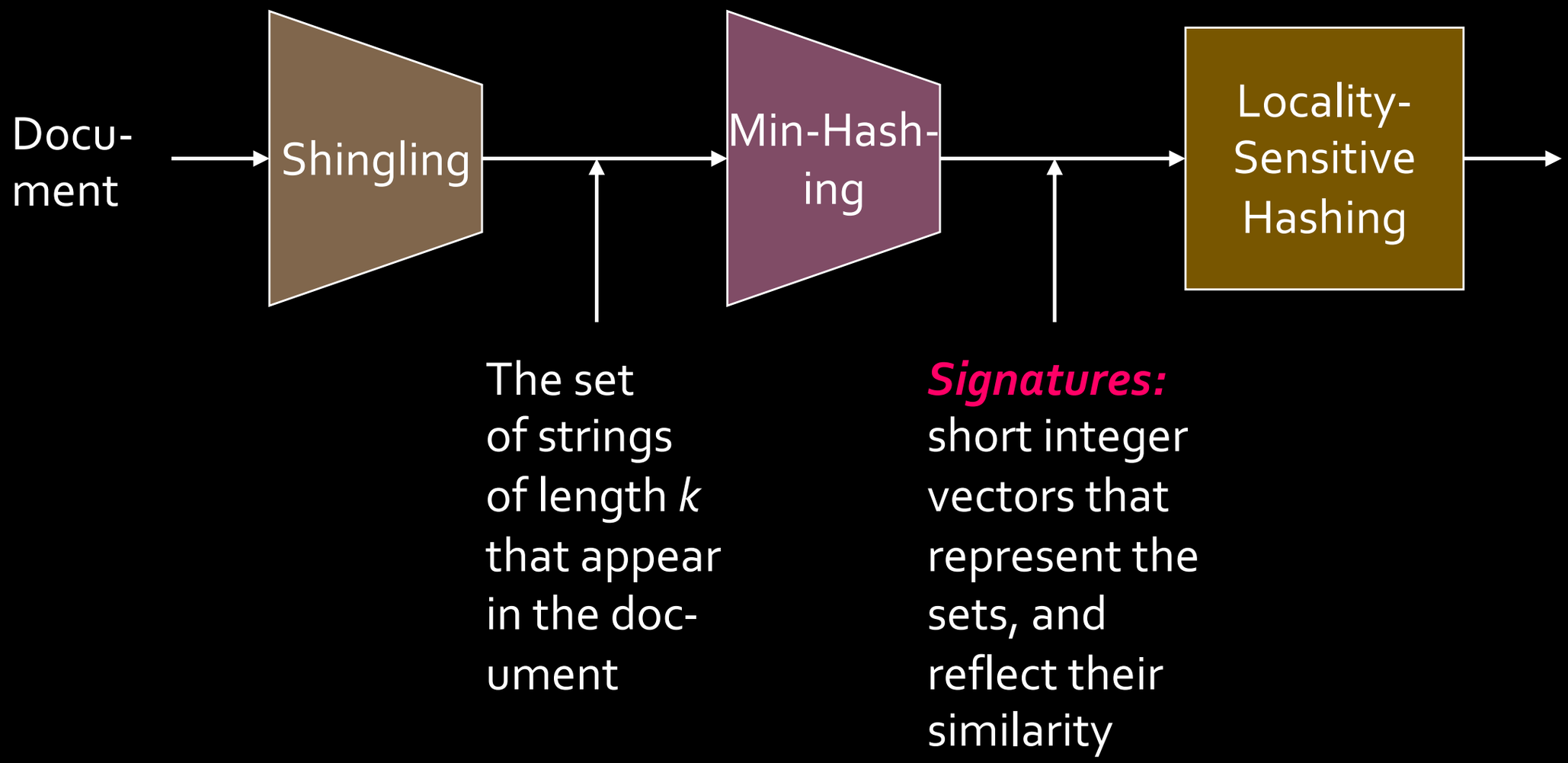
Similarities:

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

Implementation Trick

- **Permuting rows even once is prohibitive**
- **Row hashing!**
  - Pick **K = 100** hash functions h_i
  - Ordering under h_i gives a random permutation π of rows!
- **One-pass implementation**
  - For each column ***c*** and hash-func. h_i keep a "slot" $M(i, c)$ for the min-hash value of column ***c*** and hash-func ***i***
  - Initialize all $M(i, c) = \infty$
  - **Scan rows looking for 1s**
    - Suppose row ***j*** has 1 in column ***c***
    - Then for each h_i :
      - If $h_i(j) < M(i, c)$, then $M(i, c) \leftarrow h_i(j)$

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 … random integers
p … prime number (p > N)



Locality Sensitive Hashing

Step 3: ***Locality Sensitive Hashing:***

Focus on pairs of signatures likely to be from similar documents

LSH: Overview

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

- **Goal:** Find documents with Jaccard similarity at least ***s*** (for some similarity threshold, e.g., ***s***=0.8)

- LSH – **General idea:** Use a hash function 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**

LSH: Overview

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

- **Goal:** Find documents with Jaccard similarity at least ***s*** (for some similarity threshold, e.g., ***s***=0.8)

- **LSH – General idea:** Use a hash function 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**

LSH: Overview

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

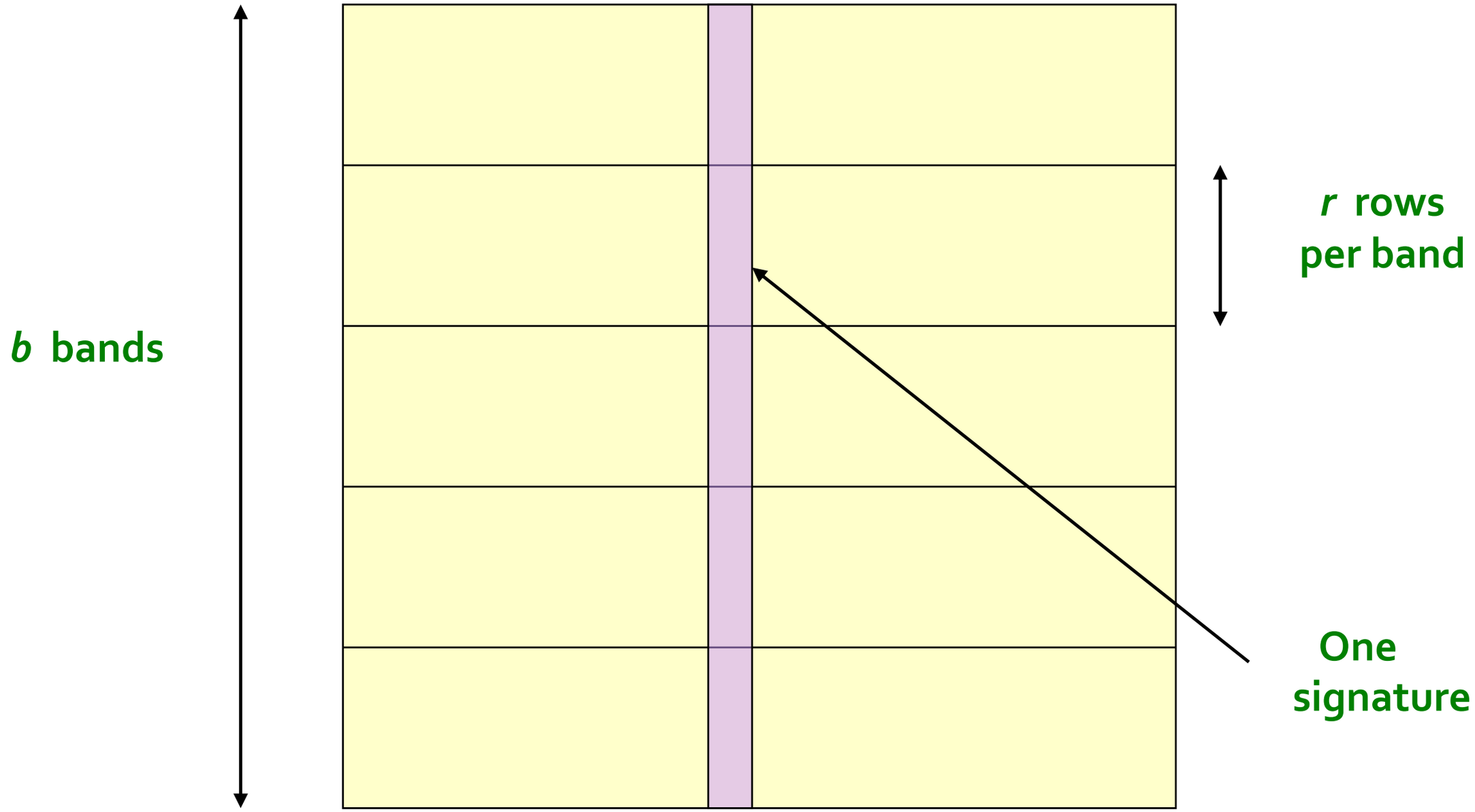
- **Goal:** Find documents with Jaccard similarity at least ***s*** (for some similarity threshold, e.g., ***s***=0.8)

- **LSH – General idea:** Use a hash function that tells whether ***x*** and ***y*** is a ***candidate pair:*** a pair of elements whose similarity must be evaluated

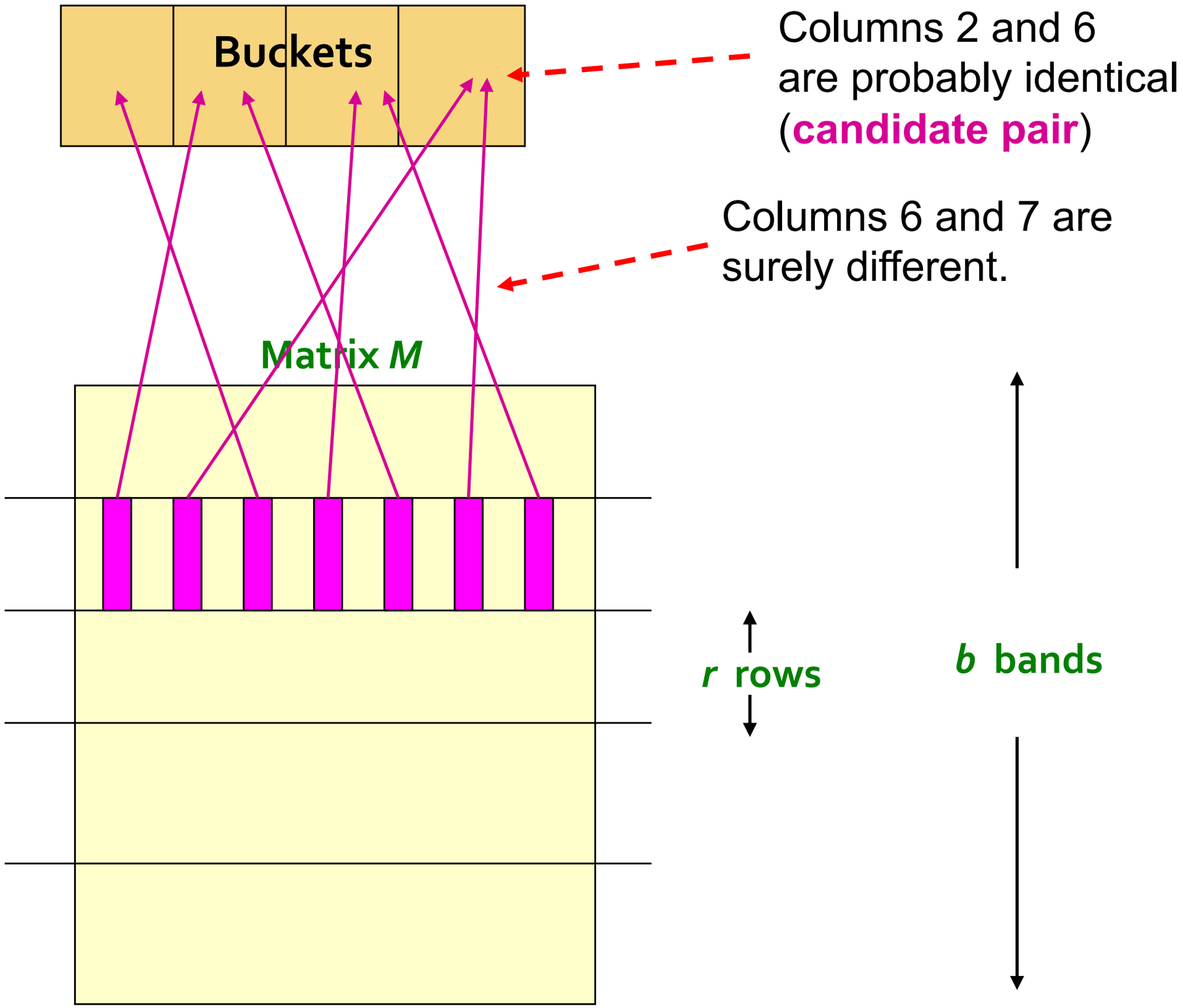
- **For Min-Hash matrices:**
  - Hash columns of signature matrix ***M*** to many buckets
  - Each pair of documents that hashes into the same bucket is a **candidate pair**

Partition *M* into *b* Bands

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



Hashing Bands

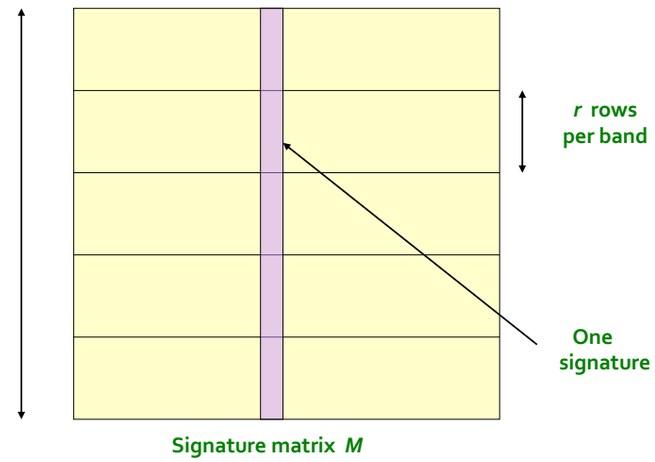


Example of Bands

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

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

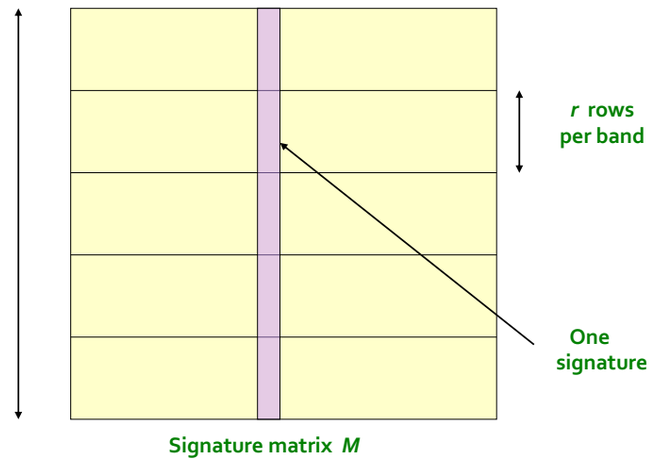


Example of Bands

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

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 tha are at least ***s*** = *0.8* similar
- Choose ***b*** = 20 bands of ***r*** = 5 integers/band



C_1, C_2 are 80% 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:** 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 1/3000th 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 80% Similar

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

- **Find pairs of ≥ *s*=0.8 similarity, set b=20, r=5**
- **Assume:** sim(C_1, C_2) = 0.8
  - Since sim(C_1, C_2) ≥ **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 1/3000th 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 80% Similar

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

- **Find pairs of ≥ *s*=0.8 similarity, set b=20, r=5**
- **Assume:** sim(C_1, C_2) = 0.8
  - Since sim(C_1, C_2) ≥ **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 1/3000th 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

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

- **Find pairs of ≥ *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**

C_1, C_2 are 30% Similar

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

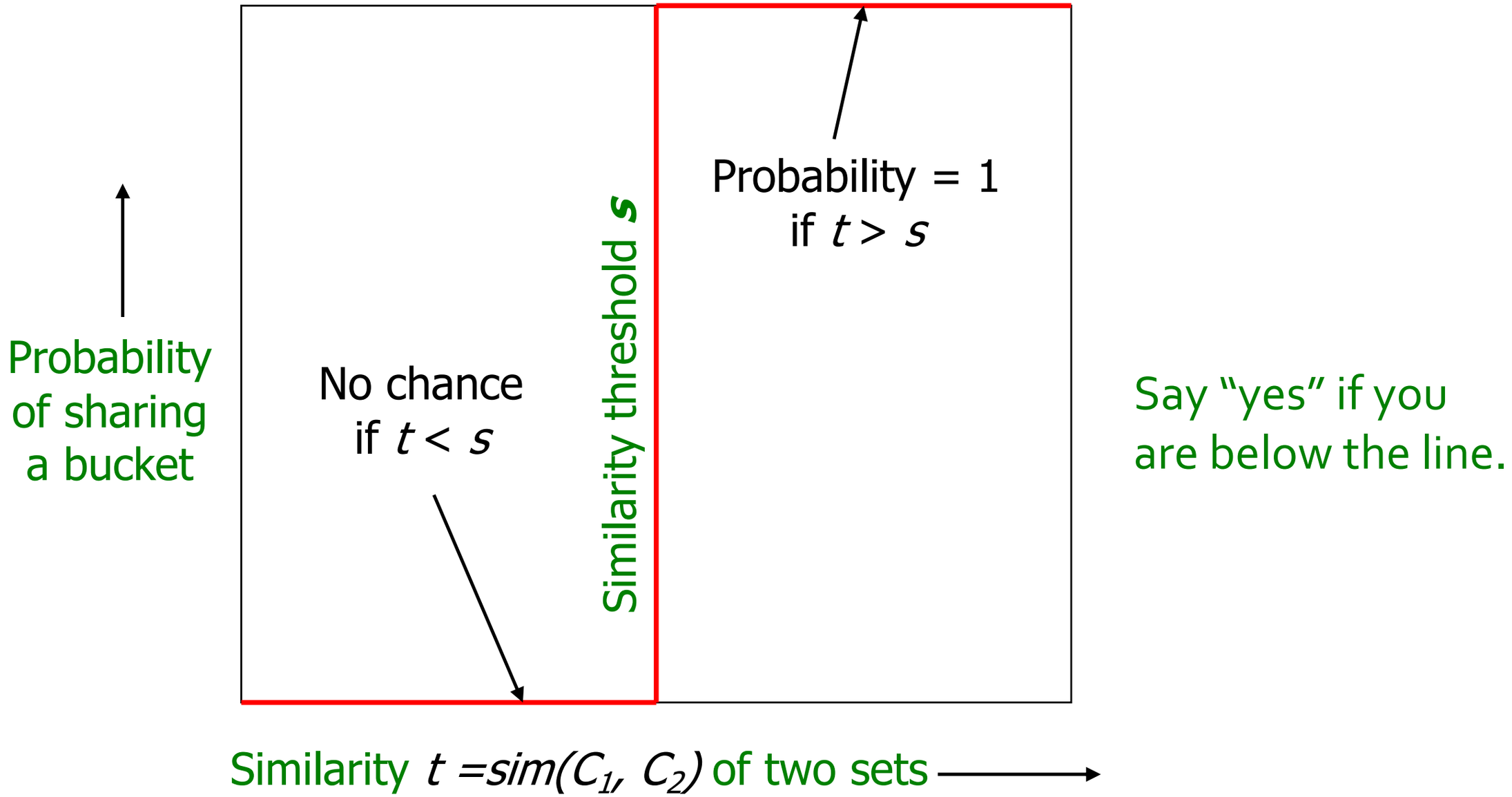
- **Find pairs of ≥ *s*=0.8 similarity, set b=20, r=5**
- **Assume:** sim(C_1, C_2) = 0.3
  - Since sim(C_1, C_2) < **s** we want C_1, C_2 to hash to **NO common buckets** (all bands should be different)
- **Probability C_1, C_2 identical in one particular band:** $(0.3)^5 = 0.00243$
- Probability C_1, 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**

C_1, C_2 are 30% Similar

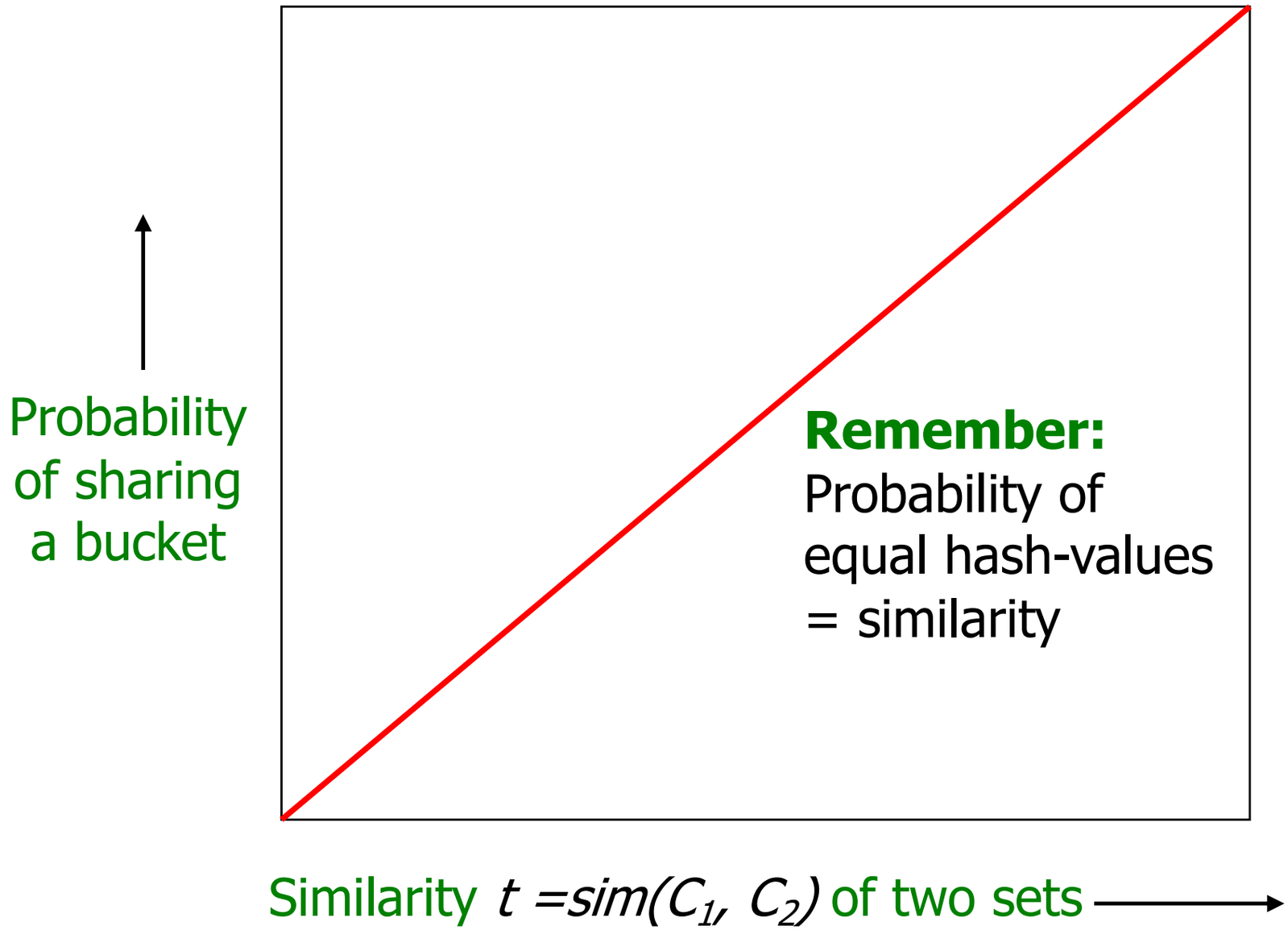
2	1	4	1
1	2	1	2
2	1	2	1

- **Find pairs of ≥ *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**

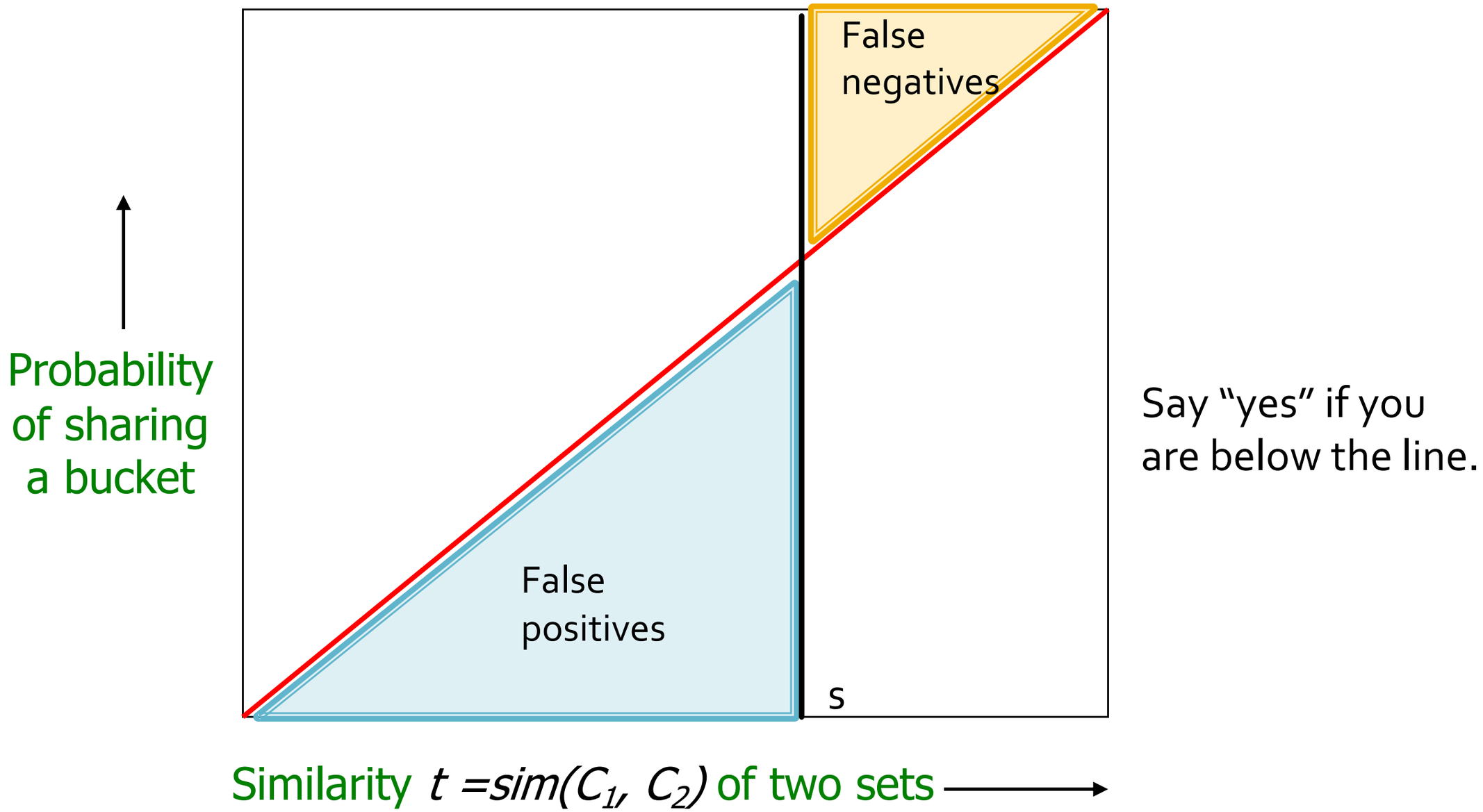
Analysis of LSH – What We Want



What 1 Band of 1 Row Gives You



What 1 Band of 1 Row Gives You



b bands, *r* rows/band

- Say 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 =

$$1 - (1 - t^r)^b$$

b bands, *r* rows/band

- Say 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 =

$$1 - (1 - t^r)^b$$

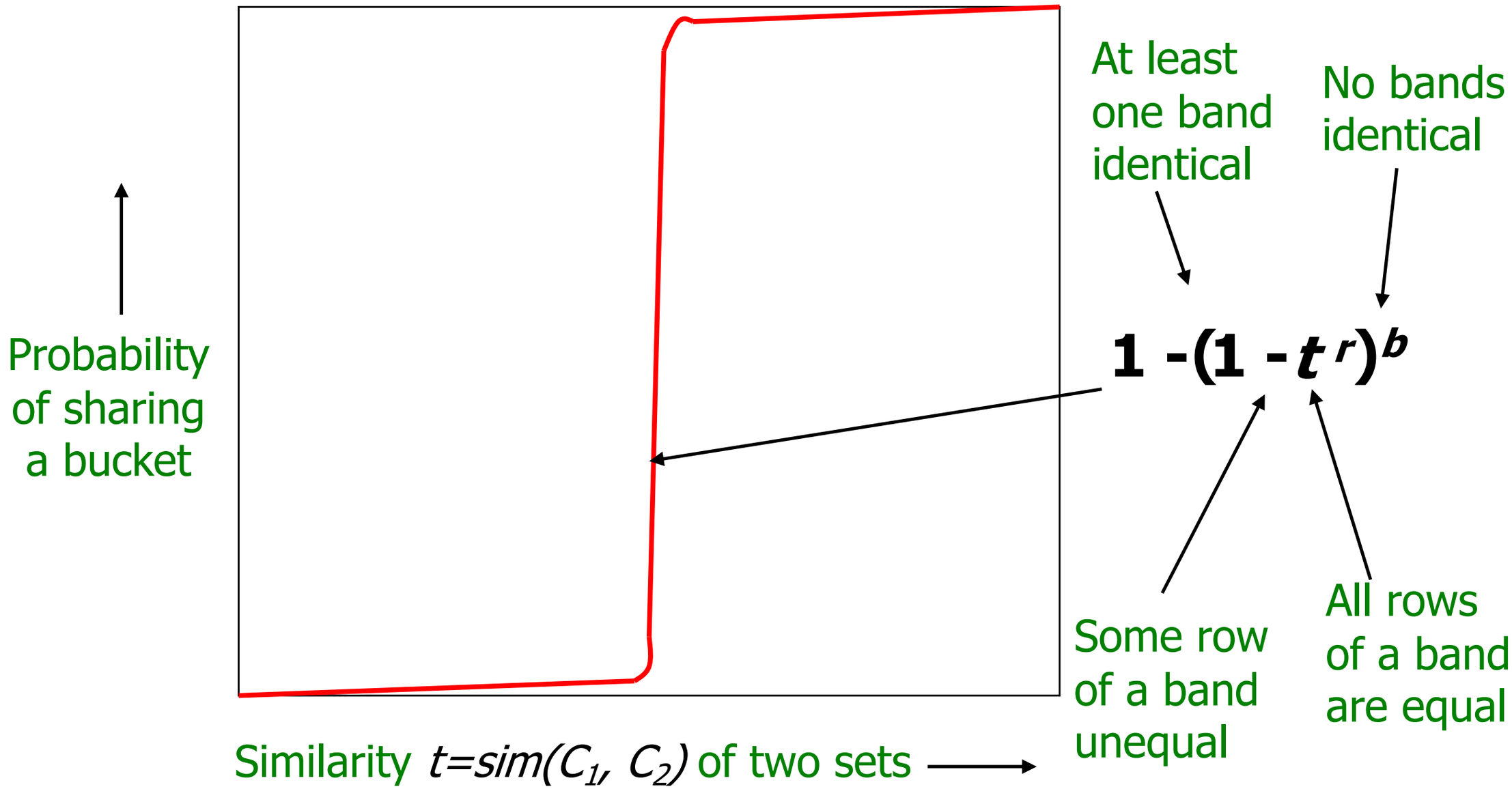
b bands, *r* rows/band

- Say 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 =

$$1 - (1 - t^r)^b$$

What *b* Bands of *r* Rows Gives You



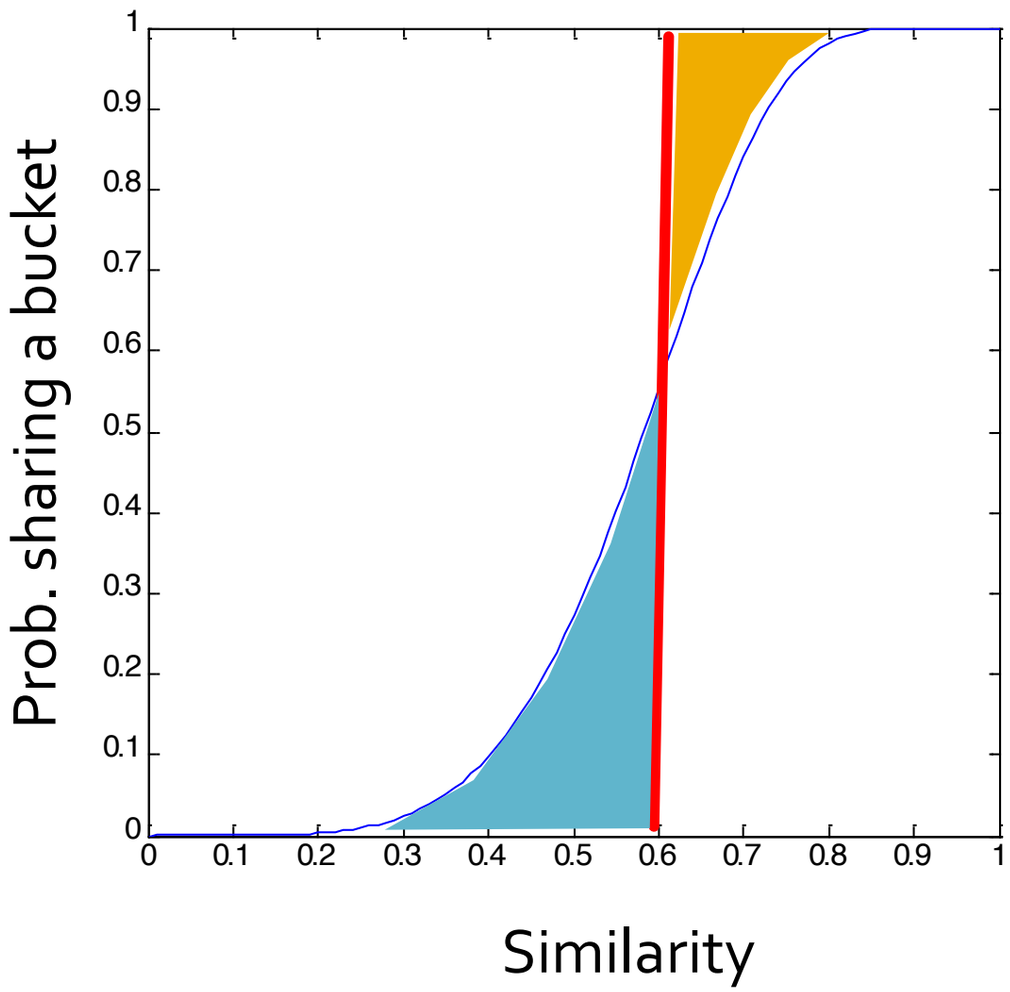
Example: $b = 20$; $r = 5$

- **Similarity threshold s**
- **Prob. that at least 1 band is identical:**

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

Picking *r* and *b*: The S-curve

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



Yellow area: False Negative rate
Blue area: False Positive rate

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 $\mathbf{Pr}[h_\pi(C_1) = h_\pi(C_2)] = \mathbf{sim}(C_1, C_2)$
  - 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**