

Advanced Topics in Information Retrieval

4. Mining & Organization

Vinay Setty

(vsetty@mpi-inf.mpg.de)

Jannik Strötgen

(jtroetge@mpi-inf.mpg.de)

Mining & Organization

- ▶ Retrieving a list of relevant documents ([10 blue links](#)) insufficient
 - ▶ for vague or exploratory information needs (e.g., “find out about brazil”)
 - ▶ when there are more documents than users can possibly inspect
- ▶ **Organizing and visualizing** collections of documents can help users to **explore and digest** the contained information, e.g.:
 - ▶ **Clustering** groups content-wise similar documents
 - ▶ **Faceted search** provides users with means of exploration

Outline

4.1. Clustering

4.2. Finding similar documents

4.3. Faceted Search

4.4. Tracking Memes

4.1. Clustering

- ▶ **Clustering** groups content-wise similar documents
- ▶ **Clustering** can be used to structure a document collection (e.g., entire corpus or query results)
- ▶ **Clustering methods:** DBScan, *k*-Means, *k*-Medoids, hierarchical agglomerative clustering, nearest neighbor clustering

The screenshot shows the Yippy search engine interface. At the top, there are navigation links for 'web', 'news', 'wikipedia', 'jobs', and 'more'. A search bar contains the text 'Star Wars' and a 'Search' button. To the right of the search bar is a link for 'advanced preferences'. Below the search bar, there are tabs for 'clouds', 'sources', 'sites', and 'time', with a 'remix' button. The main content area displays search results for 'Star Wars'. On the left, there is a sidebar with a list of categories and their counts: All Results (523), Pictures (61), Force Awakens (42), Awards, Winners (7), Episode VII (5), Box office (5), Images, Episodes (2), Scene (2), Video Game (3), Trailer (3), Bedding, Clothes (2), Public Library (2), Classic (2), Mundo (2), Fashion (2), Frugal Living (2), Other Topics (6), Trilog, Original (21), Star Wars, Star Trek (27), Reviews, Game (27), Action Figures (26), Star Wars fan (17), Trailer (15), Tickets (11), Rebel (14), Downloads (13), Blog (13), and Oscar (9). The main results area shows several news articles with titles, dates, and snippets. The first article is 'Cluster Trilog, Original contains 21 documents.' followed by 'The Margin: Original 'Star Wars' trilogy to hit theaters again, briefly' from marketwatch.com. Other articles include 'Lucasfilm and Disney launching The Star Wars Show online' from cbc.ca, 'Star Wars essentials: Charles Ross' hit One-Man Star Wars Trilogy returns to Toronto' from cbc.ca, ''Star Wars' original trilogy returning to theaters for nationwide roadshow' from money.cnn.com, 'The real Star Wars: China's military space play' from cnbc.com, 'Original 'Star Wars' trilogy to return to theaters' from cnbc.com, and 'Star Wars: R2-D2 original builder Tony Dyson dies - BBC News' from BBC.

- ▶ Example of search result clustering: yippy.com
- ▶ Formerly called clusty.com

The screenshot shows the yippy.com search interface. At the top, there are navigation links for 'web news wikipedia jobs more' and a search bar containing 'Star Wars'. Below the search bar, there are tabs for 'clouds', 'sources', 'sites', and 'time', with a 'remix' button. The main content area is divided into two columns. The left column is a sidebar with a list of clusters, each with a count in parentheses. The right column is a list of search results, each with a title, a URL, and a source.

Search results organized as clusters

Search results (523)

- Pictures (61)
- Force Awakens (42)
- Awards, Winners (7)
- Episode VII (5)
- Box office (5)
- Images, Episodes (2)
- Scene (2)
- Video Game (3)
- Trailer (3)
- Bedding, Clothes (2)
- Public Library (2)
- Classic (2)
- Mundo (2)
- Fashion (2)
- Frugal Living (2)
- Other Topics (6)
- **Trilogy, Original (21)**
- Star Wars, Star Trek (27)
- Reviews, Game (27)
- Action Figures (26)
- Star Wars fan (17)
- Trailer (15)
- Tickets (11)
- Rebel (14)
- Downloads (13)
- Blog (13)
- Oscar (9)

Cluster Trilogy, Original contains 21 documents.

- [The Margin: Original 'Star Wars' trilogy to hit theaters again, briefly](#)
www.marketwatch.com/...to-hit-theaters-again-briefly-2016-04-12?siteid=rss&rss=1 - [cache] - Yippy News Archives
- [Lucasfilm and Disney launching The Star Wars Show online](#)
www.cbc.ca/news/arts/lucasfilm-disney-the-star-wars-show-1.3577665?cmp=rss - [cache] - Yippy News Archives
- [Star Wars essentials: Charles Ross' hit One-Man Star Wars Trilogy returns to Toronto](#)
www.cbc.ca/news/arts/charles-ross-one-man-star-wars-1.3551953?cmp=rss - [cache] - Yippy News Archives
- ['Star Wars' original trilogy returning to theaters for nationwide roadshow](#)
money.cnn.com/.../index.html?section=money_latest - [cache] - Yippy News
- [The real Star Wars: China's military space play](#)
Feb 20, 2016 - This year China plans to spend billions on space missions and anti-satellite weapons technology. Is this a threat to US space dominance?
www.cnbc.com/.../chinas-space-missions-in-2016-tied-to-military-ambitions.html - [cache] - Yippy News Archives
- [Original 'Star Wars' trilogy to return to theaters](#)
Apr 13, 2016 - Alamo Drafthouse revealed on Tuesday that it will sponsor a nationwide roadshow of the original 'Star Wars' trilogy in select theaters in August.
www.cnbc.com/.../04/12/star-wars-original-trilogy-to-hit-theaters-in-august.html - [cache] - Yippy News Archives
- [Star Wars: R2-D2 original builder Tony Dyson dies - BBC News](#)
Mar 4, 2016 -

- ▶ Example of search result clustering: yippy.com
- ▶ Formerly called clusty.com

Distance Measures

- For each application, we first need to define what “distance” means

Distance Measures

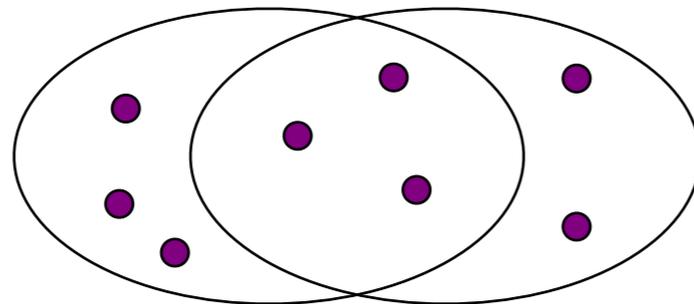
- For each application, we first need to define what “distance” means
 - Eg. : Cosine similarity, Manhattan distance, Jaccard’s distance
- **Is the distance a Metric ?**
 - **Non-negativity:** $d(x,y) \geq 0$
 - **Symmetric:** $d(x,y) = d(y,x)$
 - **Identity:** $d(x,y) = 0$ iff $x = y$
 - **Triangle inequality:** $d(x,y) + d(y,z) \geq d(x,z)$
- **Metric distance leads to better pruning power**

Jaccard Similarity

- The **Jaccard similarity** of two **sets** is the size of their intersection divided by the size of their union:
$$\text{sim}(C_1, C_2) = |C_1 \cap C_2| / |C_1 \cup C_2|$$

Jaccard Similarity

- The **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

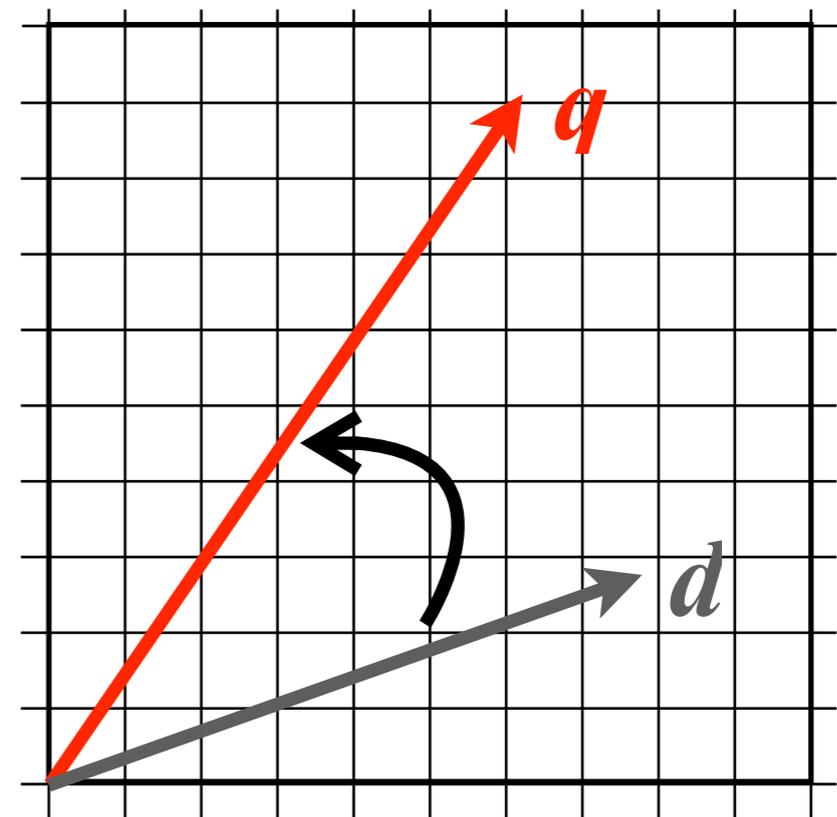
Jaccard similarity = $3/8$

Jaccard distance = $5/8$

Cosine Similarity

- ▶ Vector space model considers queries and documents as vectors in a common high-dimensional vector space
- ▶ Cosine similarity between two vectors q and d is the cosine of the angle between them

$$\begin{aligned} \text{sim}(q, d) &= \frac{q \cdot d}{\|q\| \|d\|} \\ &= \frac{\sum_v q_v d_v}{\sqrt{\sum_v q_v^2} \sqrt{\sum_v d_v^2}} \end{aligned}$$



k-Means

- ▶ **Cosine similarity** $\text{sim}(c,d)$ between document vectors c and d
- ▶ **Clusters** C_i represented by a cluster centroid document vector C_i
- ▶ **k-Means** groups documents into k clusters, maximizing the average similarity between documents and their cluster centroid

$$\frac{1}{|D|} \sum_{d \in D} \max_{c \in C} \text{sim}(c, d)$$

- ▶ Document d is assigned to cluster C having most similar centroid

Documents-to-Centroids

Documents-to-Centroids

- ▶ **k-Means** is typically implemented iteratively with every iteration reading all documents and assigning them to most similar cluster
 - ▶ initialize cluster centroids C_1, \dots, C_k (e.g., as random documents)
 - ▶ **while** not converged (i.e., cluster assignments unchanged)
 - ▶ **for** every document d , determine **most similar** c_i , and **assign** it to C_i
 - ▶ recompute c_i as **mean** of documents assigned to cluster C_i

Documents-to-Centroids

- ▶ **k-Means** is typically implemented iteratively with every iteration reading all documents and assigning them to most similar cluster
 - ▶ initialize cluster centroids C_1, \dots, C_k (e.g., as random documents)
 - ▶ **while** not converged (i.e., cluster assignments unchanged)
 - ▶ **for** every document d , determine **most similar** c_i , and **assign** it to C_i
 - ▶ recompute c_i as **mean** of documents assigned to cluster C_i
- ▶ Problem: Iterations need to **read the entire document collection**, which has cost in $O(nkd)$ with n as number of documents, k as number of clusters and, and d as number of dimensions

Centroids-to-Documents

- ▶ Broder et al. [1] devise an alternative method to implement k-Means, which makes use of **established IR methods**
- ▶ Key Ideas:
 - ▶ build an **inverted index** of the document collection
 - ▶ treat **centroids as queries** and identify the top- ℓ most similar documents in every iteration using **WAND**
 - ▶ documents showing up in **multiple top- ℓ results** are assigned to the most similar centroid
 - ▶ **recompute centroids** based on assigned documents
 - ▶ finally, assign **outliers** to cluster with most similar centroid

Sparsification

- ▶ While documents are typically sparse (i.e., contain only relatively few features with non-zero weight), **cluster centroids are dense**
- ▶ Identification of top- ℓ most similar documents to a cluster centroid can further be sped up by sparsifying, i.e., **considering only the p features having highest weight**

Experiments

- ▶ Datasets: Two datasets each with about 1M documents but different numbers of dimensions: ~26M for (1), ~7M for (2)

System	ℓ	Dataset 1 Similarity	Dataset 1 Time	Dataset 2 Similarity	Dataset 2 Time
k-means	—	0.7804	445.05	0.2856	705.21
wand-k-means	100	0.7810	83.54	0.2858	324.78
wand-k-means	10	0.7811	75.88	0.2856	243.9
wand-k-means	1	0.7813	61.17	0.2709	100.84

System	p	ℓ	Dataset 1 Similarity	Dataset 1 Time	ℓ	Dataset 2 Similarity	Dataset 2 Time
k-means	—	—	0.7804	445.05	—	0.2858	705.21
wand-k-means	—	1	0.7813	61.17	10	0.2856	243.91
wand-k-means	500	1	0.7817	8.83	10	0.2704	4.00
wand-k-means	200	1	0.7814	6.18	10	0.2855	2.97
wand-k-means	100	1	0.7814	4.72	10	0.2853	1.94
wand-k-means	50	1	0.7803	3.90	10	0.2844	1.39

- ▶ Time per iteration reduced from 445 minutes to 3.9 minutes on Dataset 1; 705 minutes to 1.39 minutes on Dataset 2

Outline

4.1. Clustering

4.2. Finding similar documents

4.2. Faceted Search

4.3. Tracking Memes

Similar Items Problem

- ▶ Similar Items
- ▶ Finding similar news stories
- ▶ Finding near duplicate images
- ▶ Plagiarism detection
- ▶ Duplications in Web crawls
- ▶ Find near-neighbors in high-dim. space
- ▶ Nearest neighbors are points that are a small distance apart

Near duplicate news articles

World | Tue Apr 12, 2016 8:23am EDT

Related: WORLD, AFF

Iceland finance minister says won't resign over Panama Papers leaks



Sigurdur Ingi Johannsson (L), minister of fisheries and agriculture of the Progressive Party who was named as new prime minister by two government coalition parties attends press conference together with finance minister Bjarni Benediktsson in Reykjavik, Iceland on April 6,...

Iceland's Finance Minister Bjarni Benediktsson said on Tuesday he would not resign over the Panama Papers leaks, which showed he once had a stake in an offshore investment firm in the Seychelles.



Tuesday, April 12, 2016, 11:36 by Reuters

Iceland finance minister says he won't resign over Panama Papers

Iceland's Finance Minister Bjarni Benediktsson said today that he would not resign over the Panama Papers leaks, which showed he once had a stake in an offshore investment firm in the Seychelles.

Asked by reporters in London whether he would quit, Benediktsson answered: "No".

The statement came a week after Sigmundur David Gunnlaugsson [resigned as prime minister](#) over the leaked documents which showed his wife owned an offshore company that held debt from failed Icelandic banks.

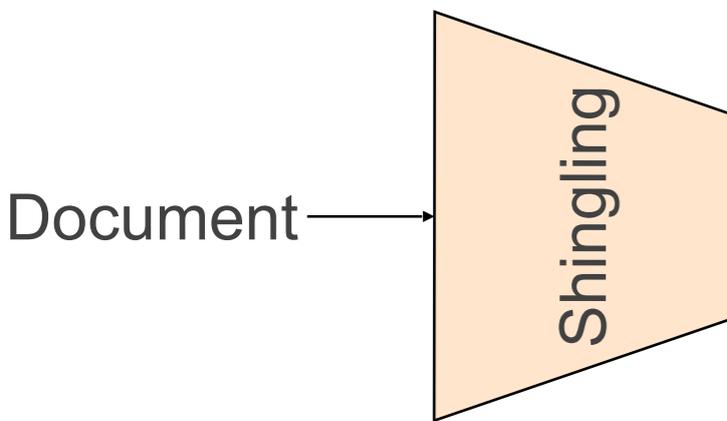
The [International Consortium of Investigative Journalists](#), which saw the leaked papers from Panamanian lawyers Mossack Fonseca, said they showed Benediktsson and two Icelandic businessmen had power of attorney over a shell company called Falson & Co. created in 2005 in the Seychelles.



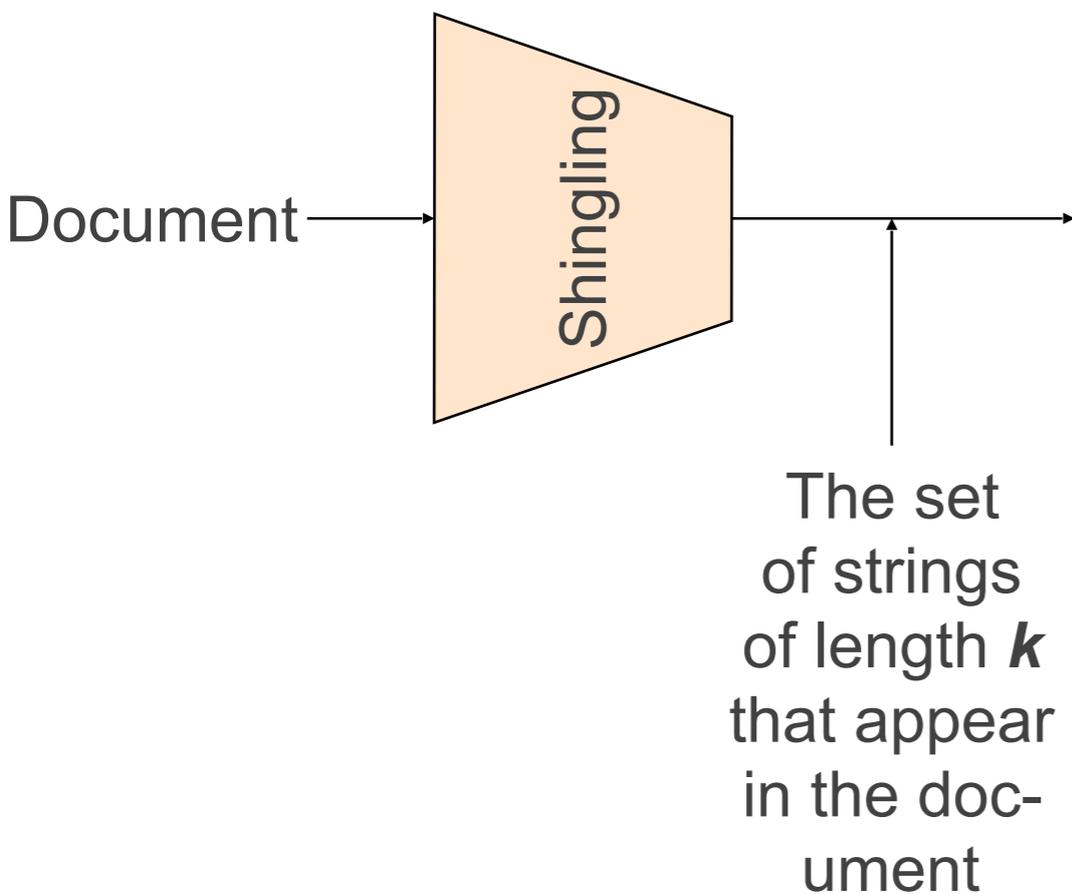
Near duplicate images



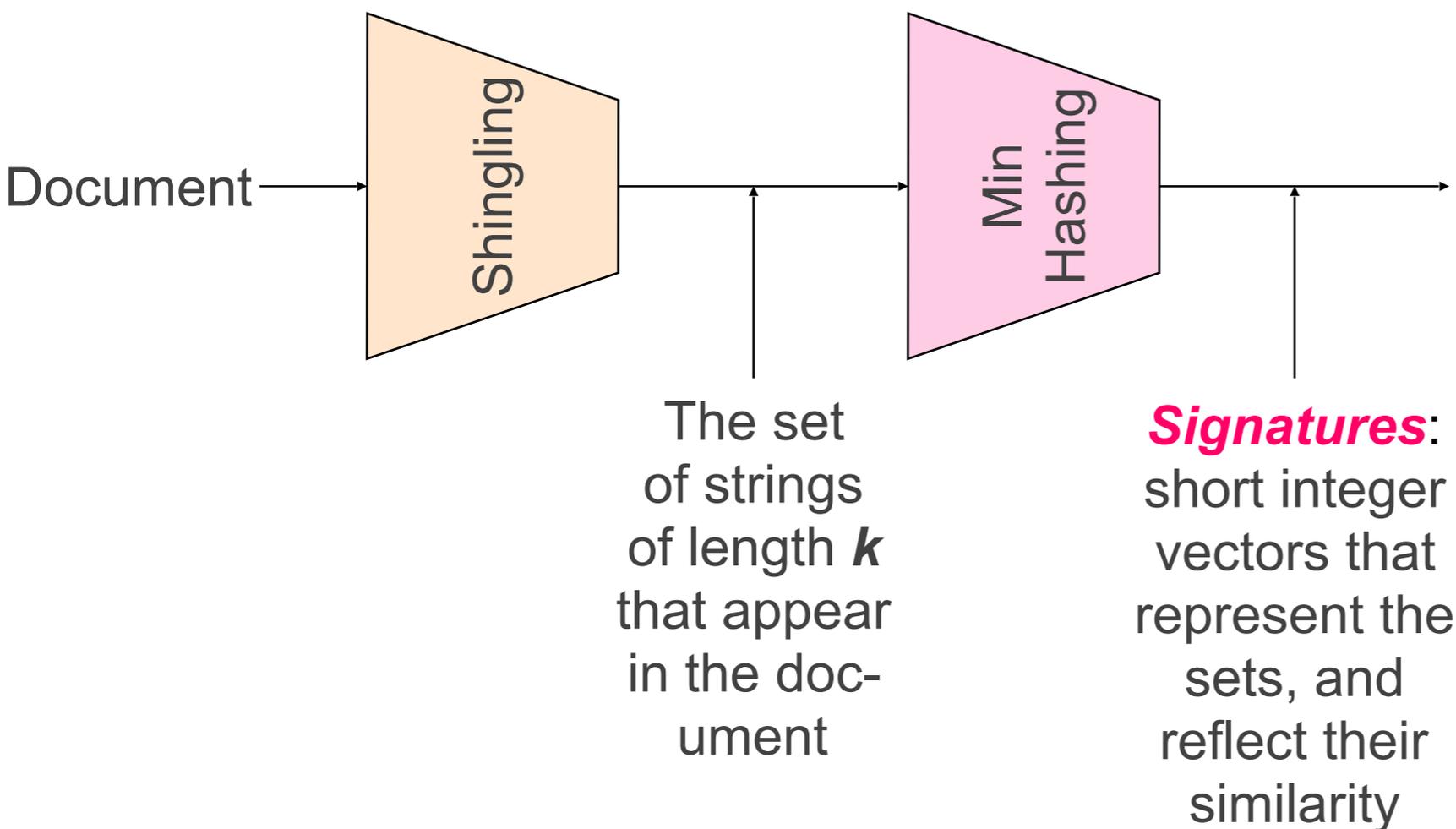
The Big Picture



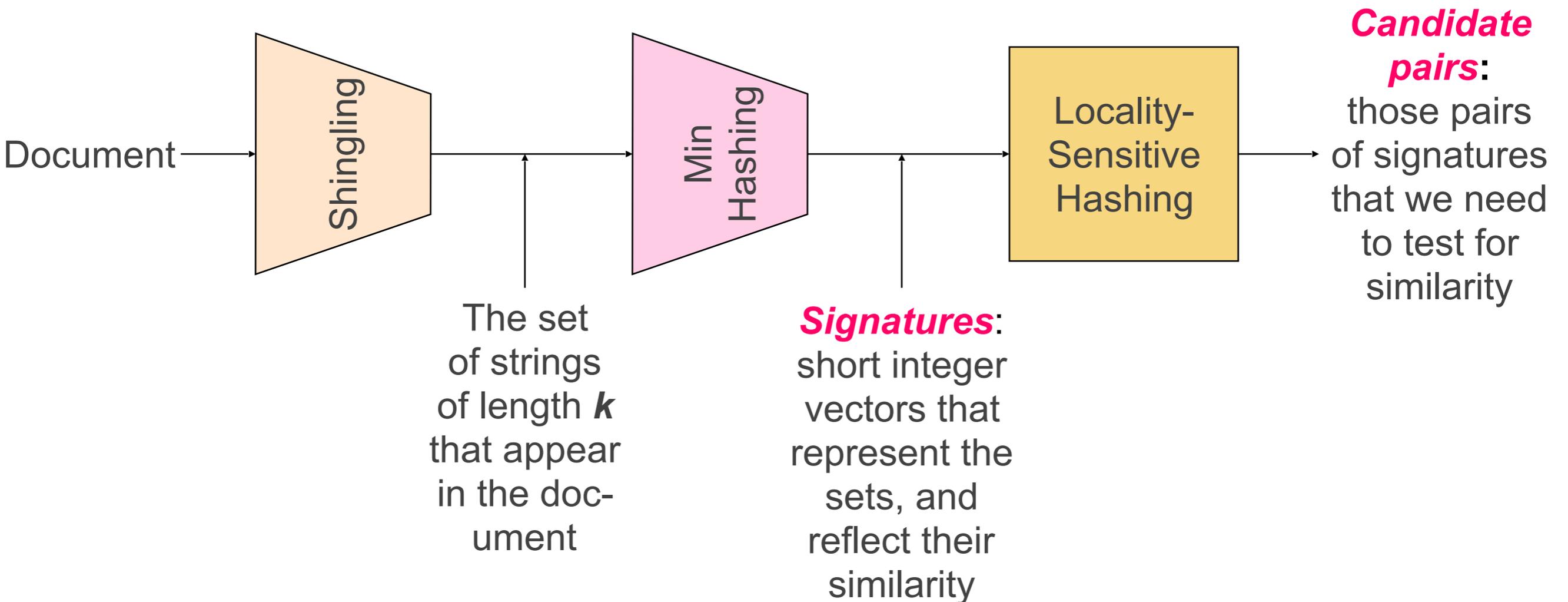
The Big Picture



The Big Picture



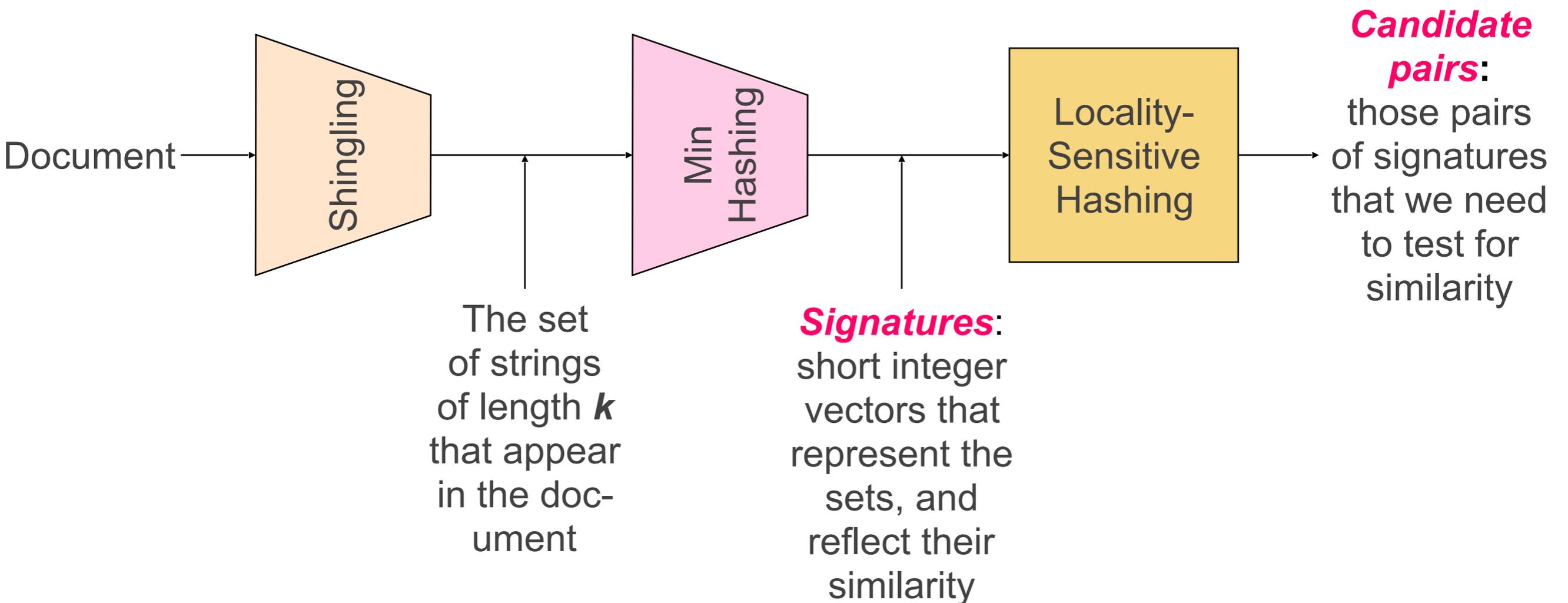
The Big Picture



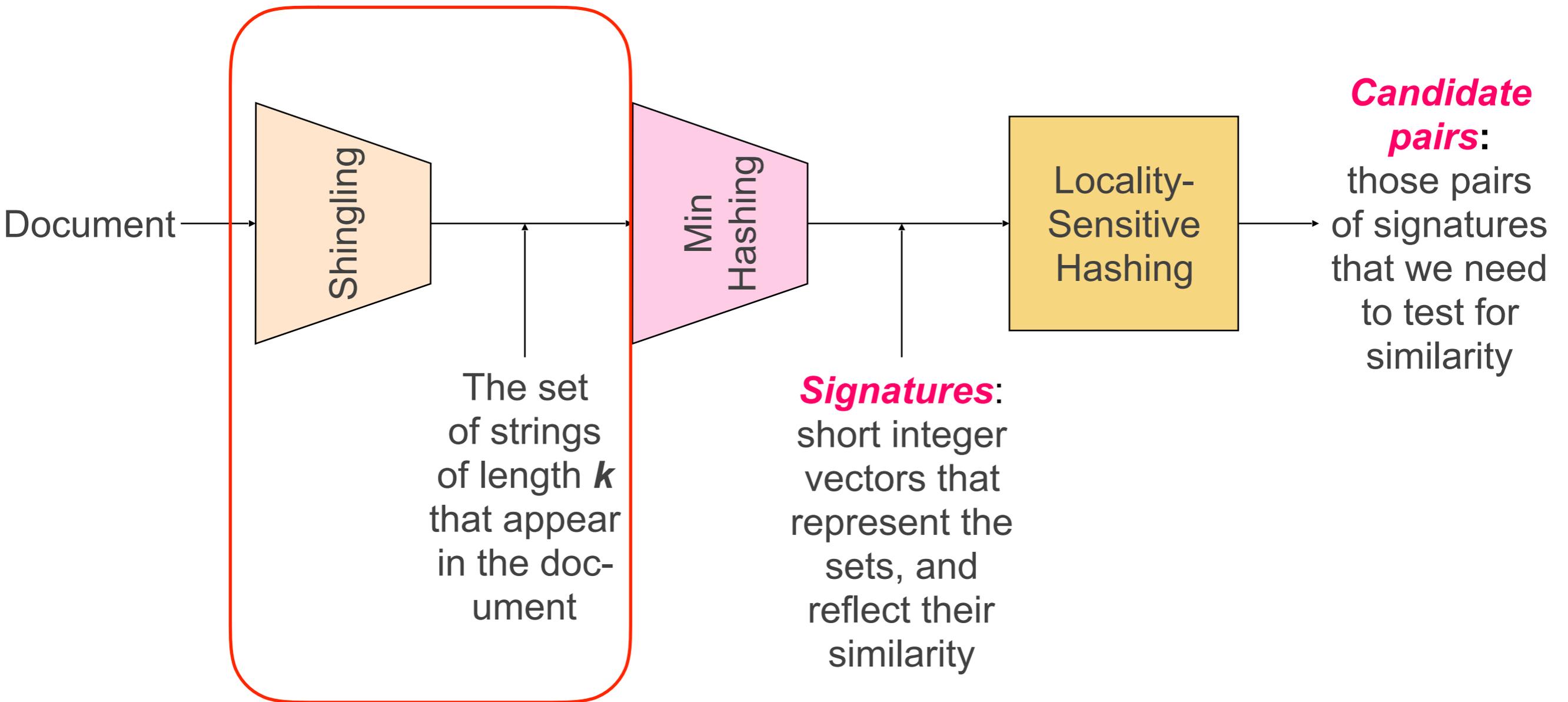
Three Essential Steps for Similar Docs

- 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 Big Picture



Documents as High-Dim. Data

Documents as High-Dim. Data

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

Documents as High-Dim. Data

- ▶ **Step 1: *Shingling*: Convert documents to sets**
- ▶ **Simple approaches:**
 - ▶ Document = set of words appearing in document
 - ▶ Document = set of “important” words
 - ▶ Don’t work well for this application. **Why?**

Documents as High-Dim. Data

- ▶ **Step 1: *Shingling*: Convert documents to sets**
- ▶ **Simple approaches:**
 - ▶ Document = set of words appearing in document
 - ▶ Document = set of “important” words
 - ▶ Don’t work well for this application. **Why?**
- ▶ **Need to account for ordering of words!**

Documents as High-Dim. Data

- ▶ **Step 1: *Shingling*: Convert documents to sets**
- ▶ **Simple approaches:**
 - ▶ Document = set of words appearing in document
 - ▶ Document = set of “important” words
 - ▶ Don’t work well for this application. **Why?**
- ▶ **Need to account for ordering of words!**
- ▶ A different way: **Shingles!**

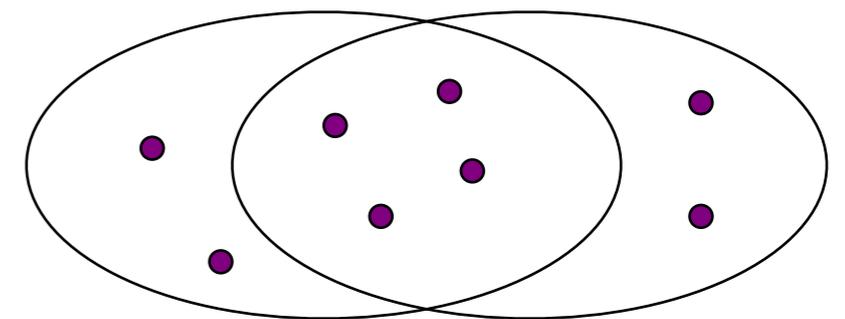
Define: Shingles

- ▶ A **k -shingle** (or **k -gram**) for a document is a sequence of k tokens that appears in the doc
 - ▶ Tokens can be **characters**, **words** or something else, depending on the application
 - ▶ Assume tokens = characters for examples
- ▶ **Example:** $k=2$; document $D_1 = \text{abcab}$
Set of 2-shingles: $S(D_1) = \{\text{ab}, \text{bc}, \text{ca}\}$
 - ▶ **Option:** Shingles as a bag (multiset), count ab twice:
 $S'(D_1) = \{\text{ab}, \text{bc}, \text{ca}, \text{ab}\}$

Similarity Metric for Shingles

- ▶ **Document D_1 is a set of its k -shingles $C_1=S(D_1)$**
- ▶ Equivalently, each document is a 0/1 vector in the space of k -shingles
 - ▶ Each unique shingle is a dimension
 - ▶ Vectors are very sparse
- ▶ A natural similarity measure is the **Jaccard similarity:**

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



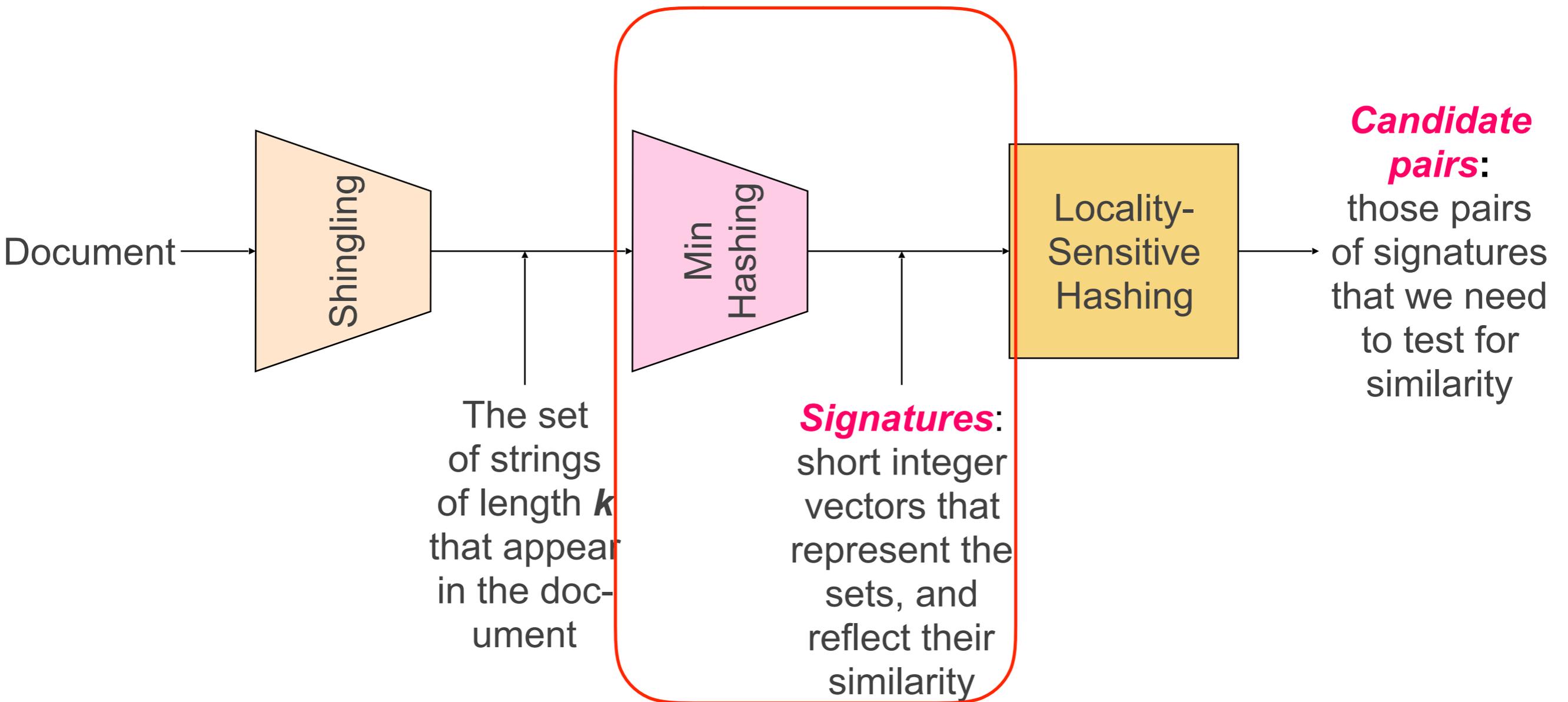
Working Assumption

- ▶ 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 Minhash/LSH

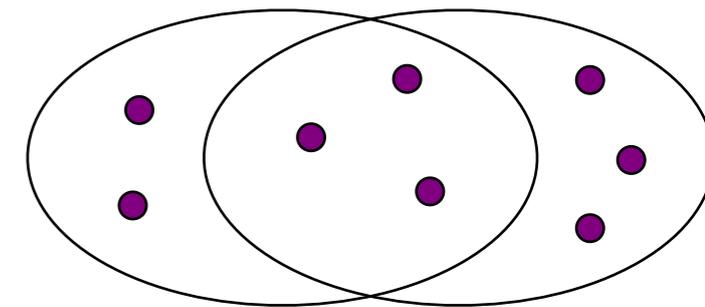
- Suppose we need to find near-duplicate documents among $N = 1$ million documents
- Naïvely, we would have to compute **pairwise Jaccard similarities** for **every pair of docs**
 - ▶ ■ $N(N - 1)/2 \approx 5 \cdot 10^{11}$ comparisons
 - At 10^5 secs/day and 10^6 comparisons/sec, it would take **5 days**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

The Big Picture



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



From Sets to 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!**

From Sets to 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:** $\text{sim}(C_1, C_2) = ?$
 - ▶ Size of intersection = 3; size of union = 6, Jaccard similarity (not distance) = $3/6$
 - ▶ $d(C_1, C_2) = 1 - (\text{Jaccard similarity}) = 3/6$

Documents (N)

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

Shingles (D)

Hashing Columns (Signatures)

- ▶ **Key idea:** “hash” each column \mathbf{C} to a small *signature* $h(\mathbf{C})$, such that:
 - ▶ (1) $h(\mathbf{C})$ is small enough that the signature fits in RAM
 - ▶ (2) $sim(\mathbf{C}_1, \mathbf{C}_2)$ is the same as the “similarity” of signatures $h(\mathbf{C}_1)$ and $h(\mathbf{C}_2)$

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) $sim(C_1, C_2)$ is the same as the “similarity” of signatures $h(C_1)$ and $h(C_2)$

- ▶ **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. $h(C_1) \neq h(C_2)$

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

Min-Hashing

- ▶ **Goal: Find a hash function $h(\cdot)$ such that:**
 - ▶ if $\text{sim}(C_1, C_2)$ is high, then with high prob. $h(C_1) = h(C_2)$
 - ▶ if $\text{sim}(C_1, C_2)$ 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

Example

Input matrix (Shingles x Documents)
Permutation π

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

Example

Input matrix (Shingles x Documents)

Permutation π

2
3
7
6
1
5
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

Example

Input matrix (Shingles x Documents)

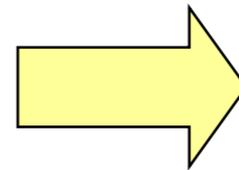
Permutation π

2
3
7
6
1
5
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
---	---	---	---



Example

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

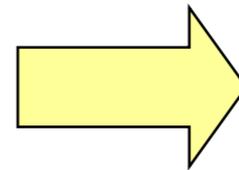
Input matrix (Shingles x Documents)
Permutation π

2
3
7
6
1
5
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
---	---	---	---



Example

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

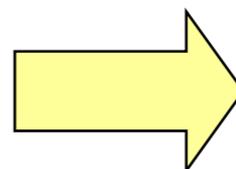
Input matrix (Shingles x Documents)
Permutation π

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

Signature matrix M

2	1	2	1
2	1	4	1



Example

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

Input matrix (Shingles x Documents)
Permutation π

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

Signature matrix M

2	1	2	1
2	1	4	1

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

Example

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

Input matrix (Shingles x Documents)
Permutation π

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

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

Example

Note: Another (equivalent) way is to store row indexes:

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

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

Input matrix (Shingles x Documents)
Permutation π

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

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

Four Types of Rows

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

	<u>C_1</u>	<u>C_2</u>
A	1	1
B	1	0
C	0	1
D	0	0

- ▶ $a = \#$ rows of type A, etc.
- ▶ Note: $\text{sim}(C_1, C_2) = a/(a + b + c)$
- ▶ Then: $\Pr[h(C_1) = h(C_2)] = \text{Sim}(C_1, C_2)$
 - ▶ 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

Similarity for Signatures

Similarity for Signatures

- ▶ We know: $\Pr[h_{\pi}(C_1) = h_{\pi}(C_2)] = \text{sim}(C_1, C_2)$

Similarity for Signatures

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

Similarity for Signatures

- ▶ We know: $\Pr[h_{\pi}(C_1) = h_{\pi}(C_2)] = \text{sim}(C_1, C_2)$
- ▶ Now generalize to multiple hash functions - why?
 - ▶ Permuting rows is expensive for large number of rows

Similarity for Signatures

- ▶ We know: $\Pr[h_{\pi}(C_1) = h_{\pi}(C_2)] = \text{sim}(C_1, C_2)$
- ▶ Now generalize to multiple hash functions - why?
 - ▶ Permuting rows is expensive for large number of rows
 - ▶ Instead we want to simulate the effect of a random permutation using hash functions

Similarity for Signatures

- ▶ We know: $\Pr[h_{\pi}(C_1) = h_{\pi}(C_2)] = \text{sim}(C_1, C_2)$
- ▶ Now generalize to multiple hash functions - why?
 - ▶ Permuting rows is expensive for large number of rows
 - ▶ Instead we want to simulate the effect of a random permutation using hash functions
- ▶ The *similarity of two signatures* is the fraction of the hash functions in which they agree

Similarity for Signatures

- ▶ We know: $\Pr[h_{\pi}(C_1) = h_{\pi}(C_2)] = \text{sim}(C_1, C_2)$
- ▶ Now generalize to multiple hash functions - why?
 - ▶ Permuting rows is expensive for large number of rows
 - ▶ Instead we want to simulate the effect of a random permutation using hash functions
- ▶ The *similarity of two signatures* is the fraction of the hash functions in which they agree

Similarity for Signatures

- ▶ We know: $\Pr[h_{\pi}(C_1) = h_{\pi}(C_2)] = \text{sim}(C_1, C_2)$
- ▶ Now generalize to multiple hash functions - why?
 - ▶ Permuting rows is expensive for large number of rows
 - ▶ Instead we want to simulate the effect of a random permutation using 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

Min-Hashing Example

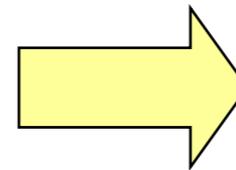
Input matrix (Shingles x Documents)

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

Permutation π

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

Min-Hash Signatures

- **Pick $K=100$ random permutations of the rows**
- Think of $\mathit{sig}(\mathbf{C})$ as a column vector
- $\mathit{sig}(\mathbf{C})[i]$ = according to the i -th permutation, the index of the first row that has a 1 in column C
$$\mathit{sig}(\mathbf{C})[i] = \min (\pi_i(\mathbf{C}))$$
- **Note:** The sketch (signature) of document C is small **~ 100 bytes!**
- **We achieved our goal! We “compressed” long bit vectors into short signatures**

Min-Hash Signatures Example

<i>Row</i>	S_1	S_2	S_3	S_4	$x + 1 \pmod{5}$	$3x + 1 \pmod{5}$
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	∞	∞	∞	∞

Init

Min-Hash Signatures Example

<i>Row</i>	S_1	S_2	S_3	S_4	$x + 1 \pmod{5}$	$3x + 1 \pmod{5}$
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	∞	∞	∞	∞

Init

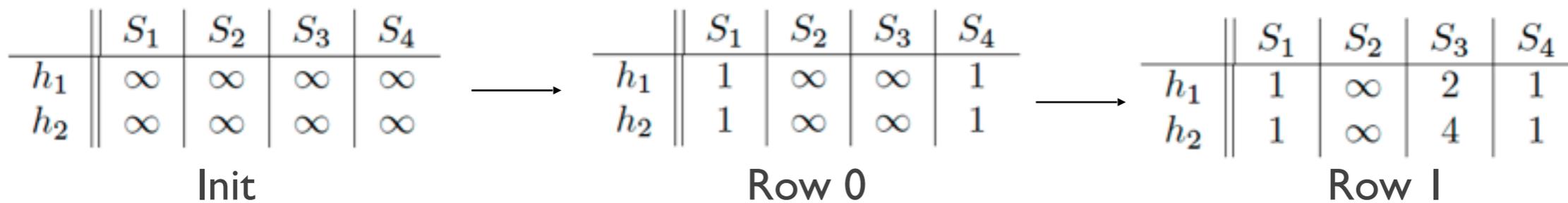
→

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

Row 0

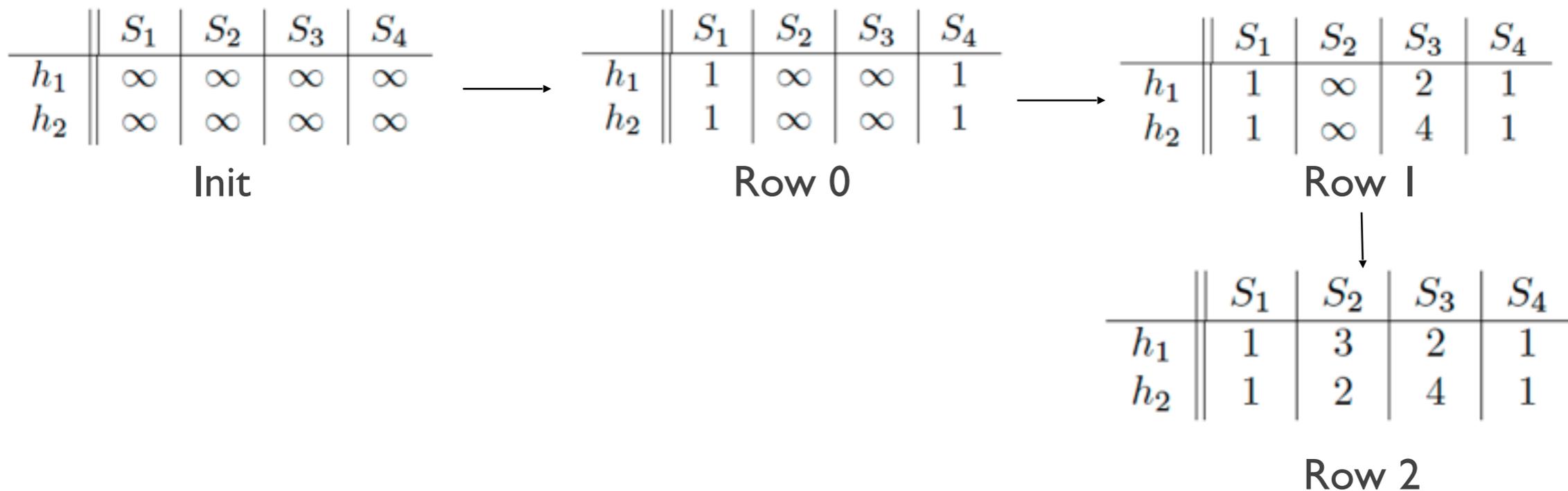
Min-Hash Signatures Example

<i>Row</i>	S_1	S_2	S_3	S_4	$x + 1 \pmod{5}$	$3x + 1 \pmod{5}$
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



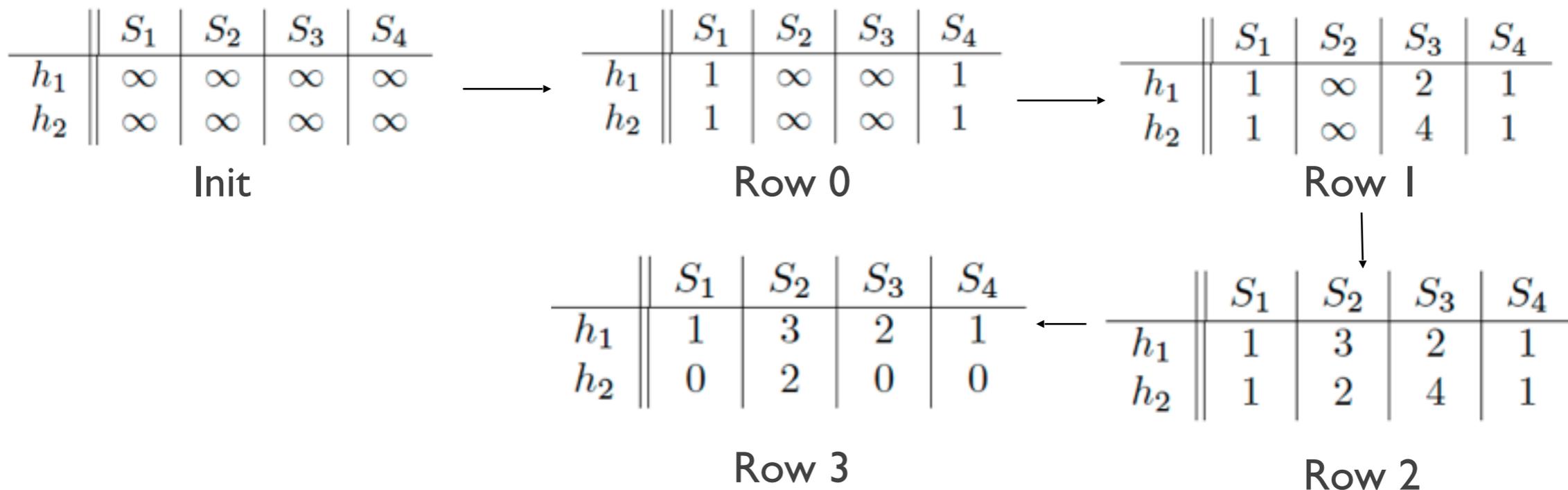
Min-Hash Signatures Example

<i>Row</i>	S_1	S_2	S_3	S_4	$x + 1 \pmod{5}$	$3x + 1 \pmod{5}$
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



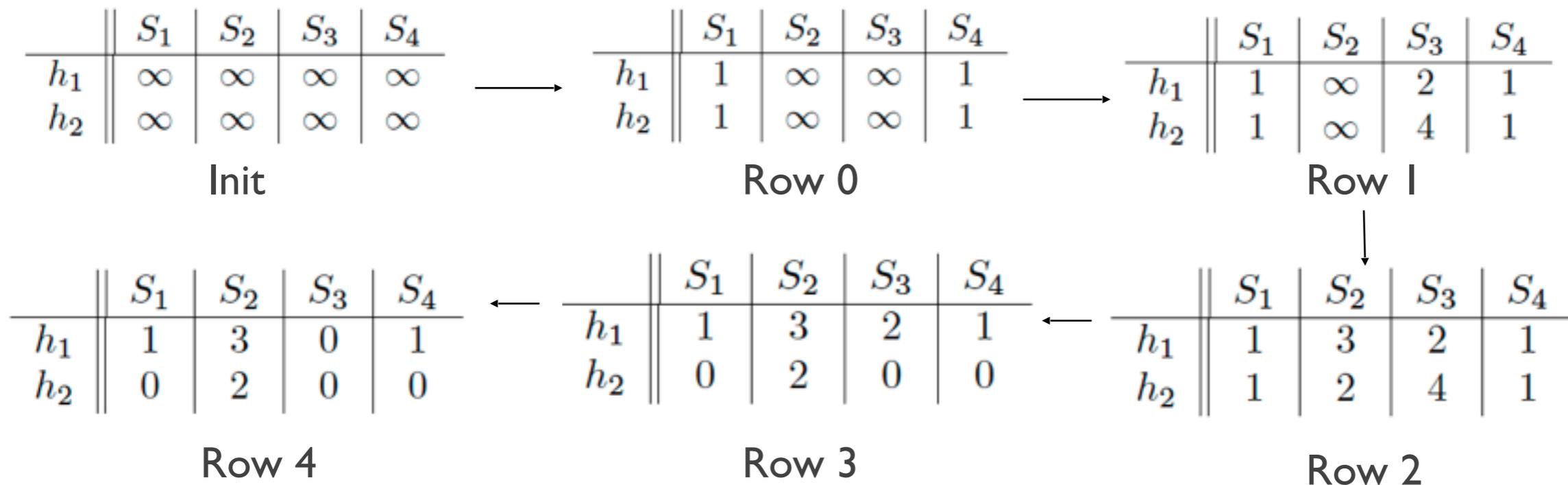
Min-Hash Signatures Example

Row	S_1	S_2	S_3	S_4	$x + 1 \pmod{5}$	$3x + 1 \pmod{5}$
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

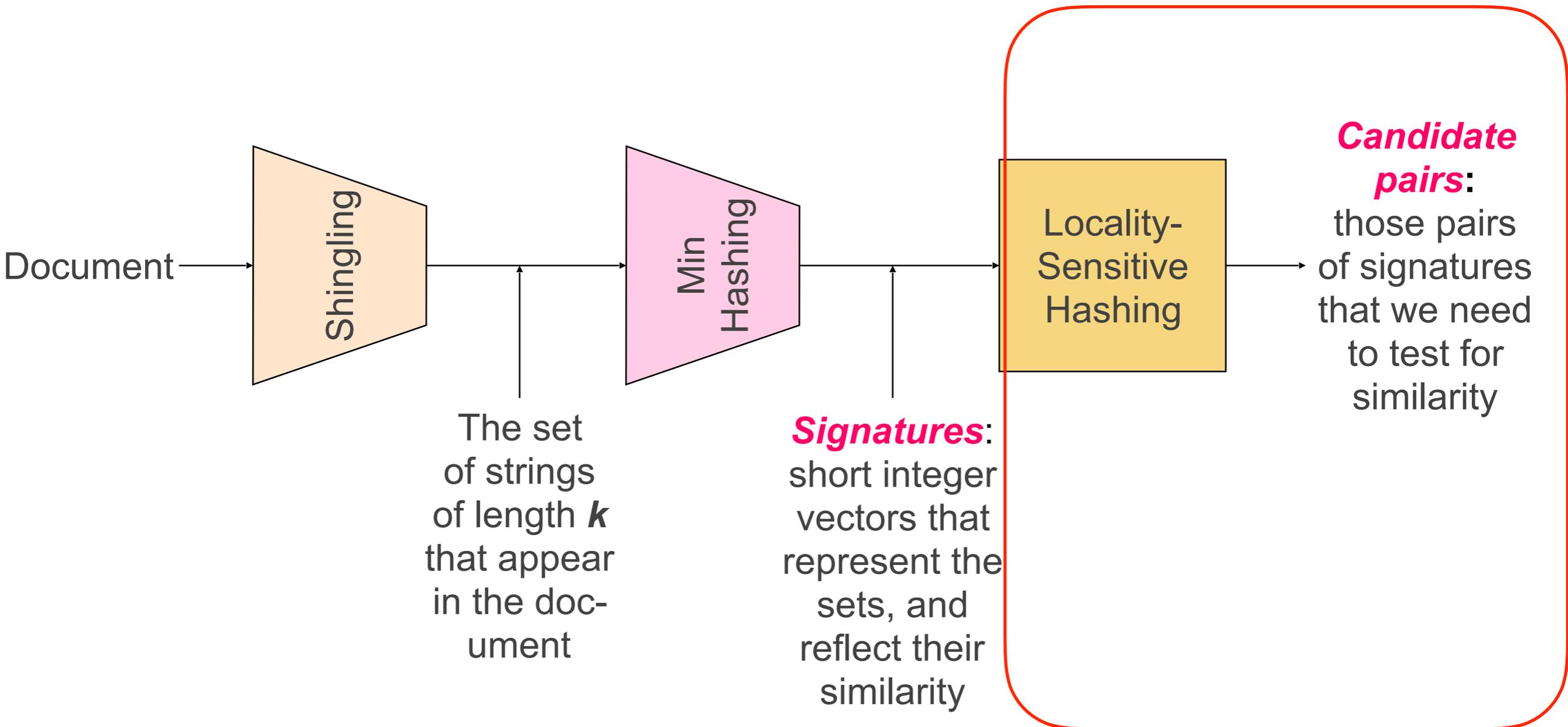


Min-Hash Signatures Example

Row	S_1	S_2	S_3	S_4	$x + 1 \pmod{5}$	$3x + 1 \pmod{5}$
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



The Big Picture



LSH: First Cut

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 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**

Candidates from Min-Hash

- ▶ Pick a similarity threshold s ($0 < s < 1$)

2	1	4	1
1	2	1	2
2	1	2	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 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

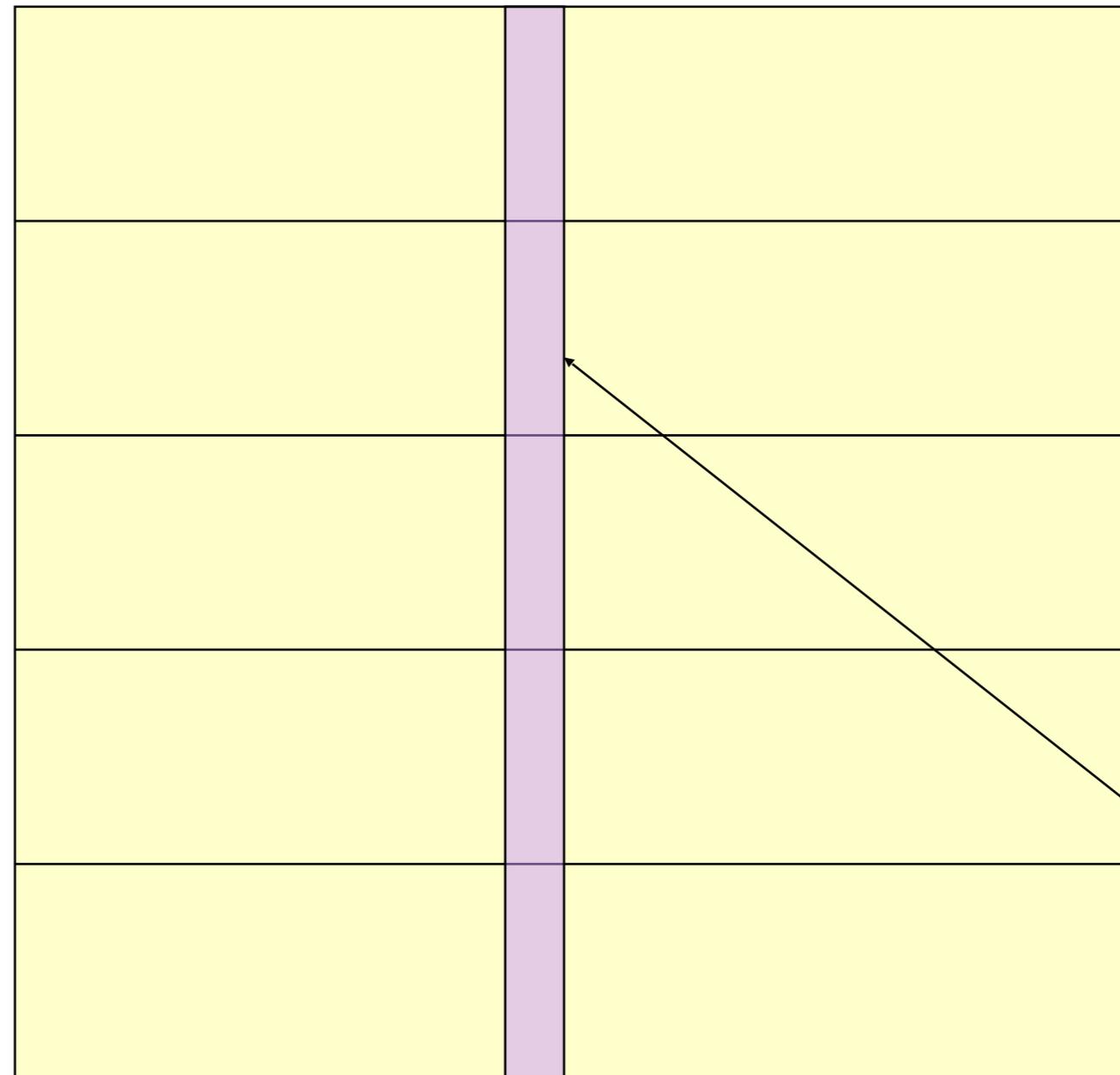
2	1	4	1
1	2	1	2
2	1	2	1

- ▶ 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 M into b Bands

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

b bands

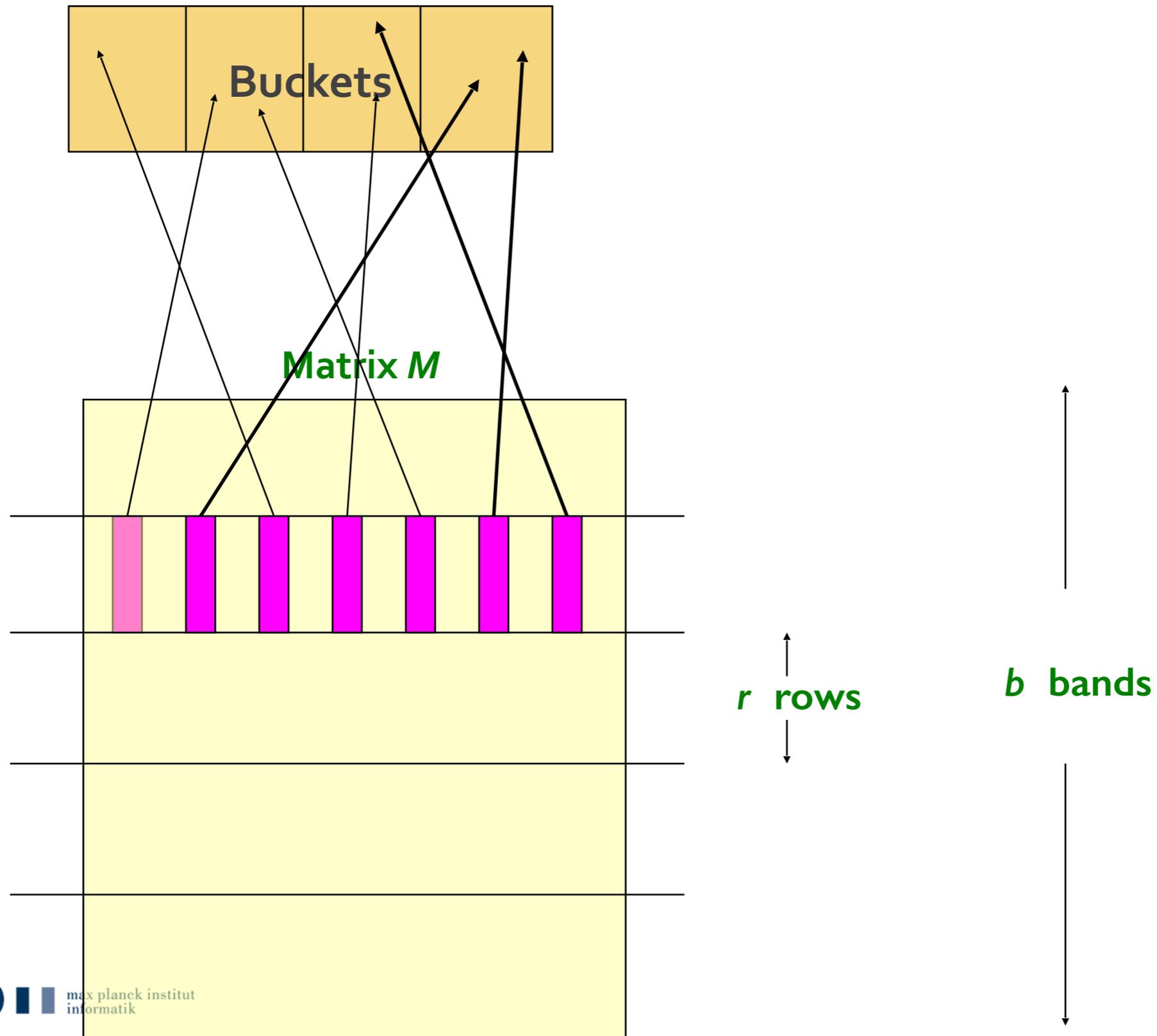


r rows
per band

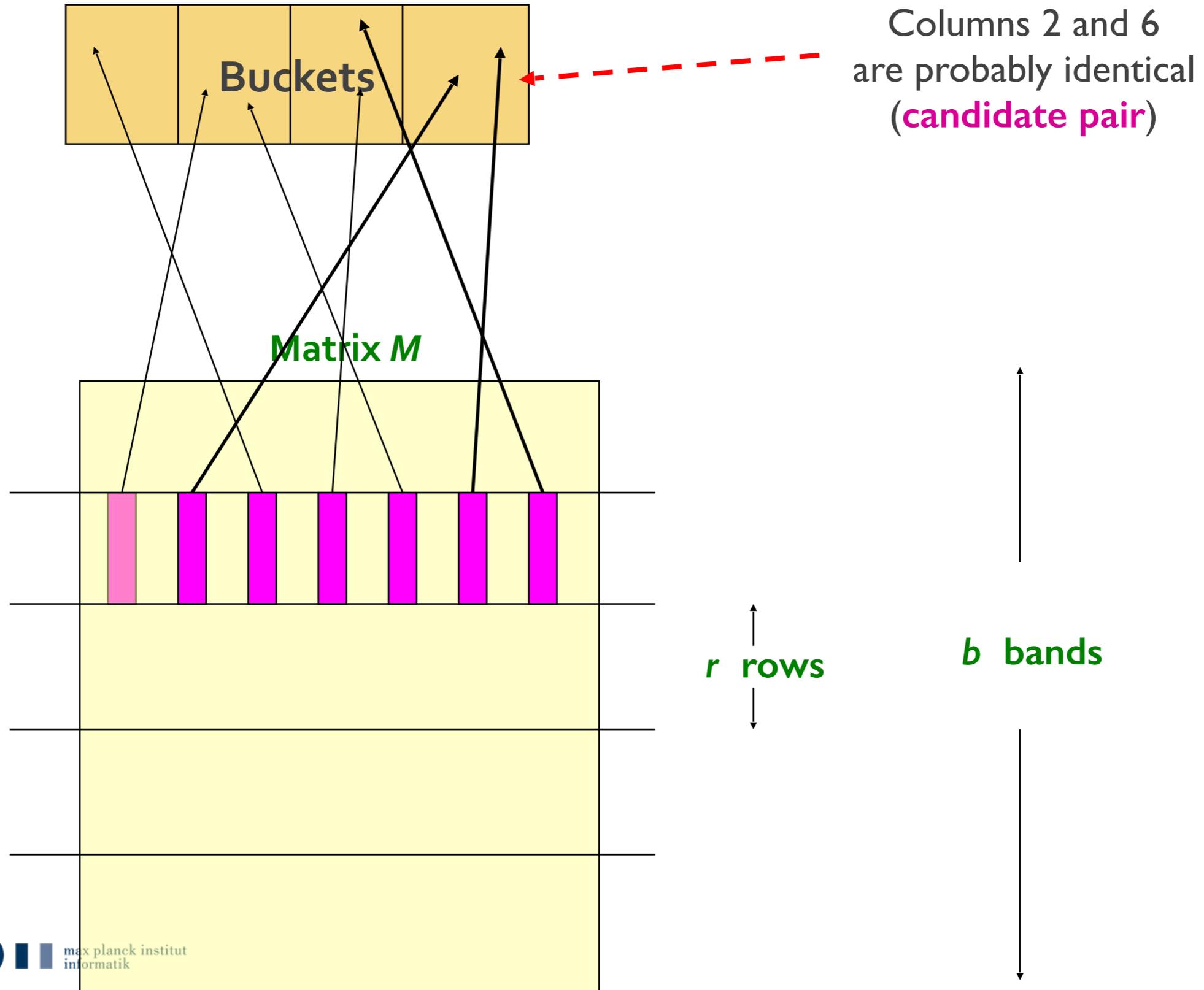
One
signature

Signature matrix M

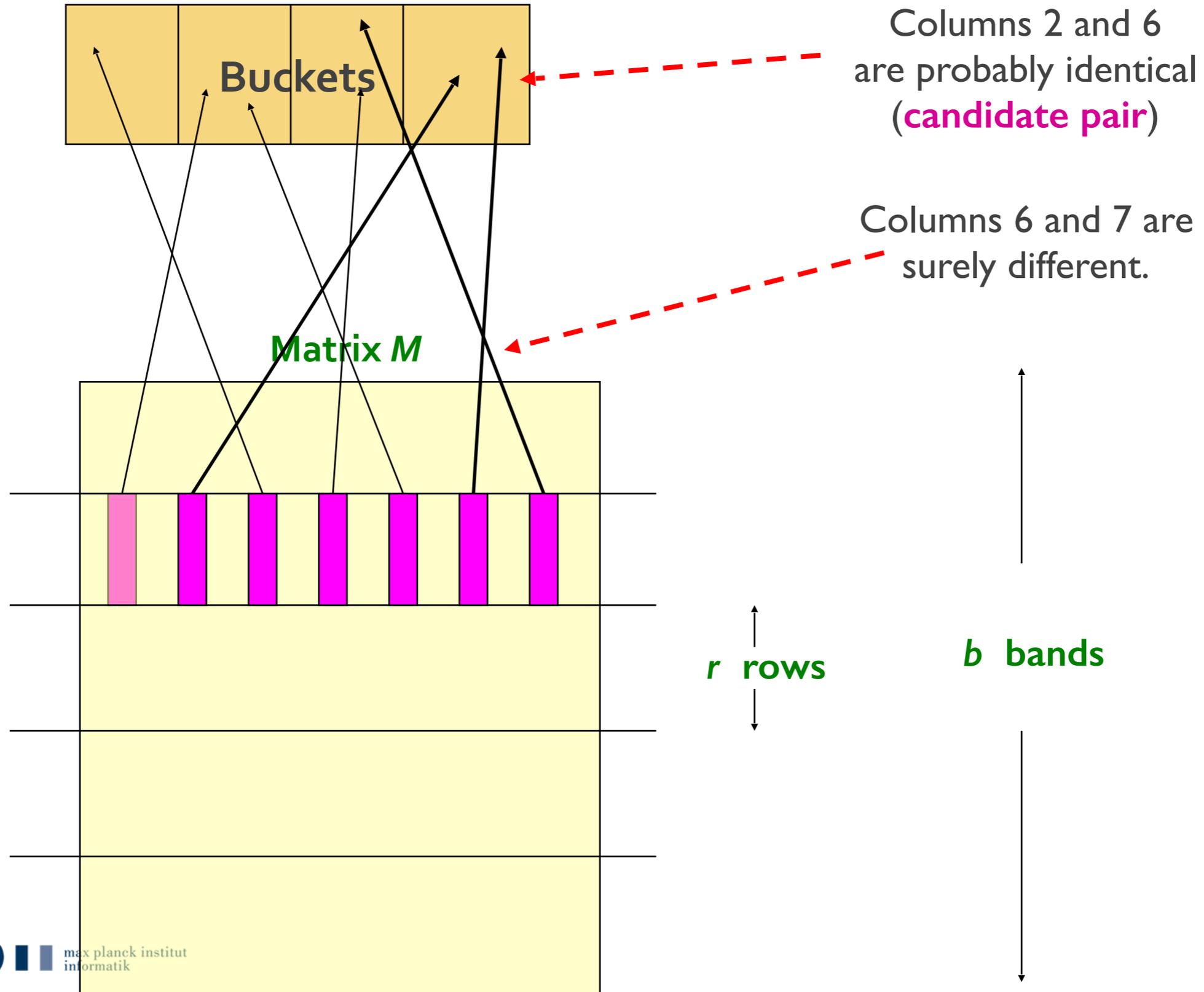
Hashing Bands



Hashing Bands



Hashing Bands



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
 - ▶ Make k as large as possible
- ▶ **Candidate** column pairs are those that hash to the same bucket for $\geq l$ band
- ▶ Tune b and r to catch most similar pairs, but few non-similar pairs

Simplifying Assumption

- ▶ There are **enough buckets** that columns are unlikely to hash to the same bucket unless they are **identical** in a particular band
- ▶ Hereafter, we assume that “**same bucket**” means “**identical in that band**”
- ▶ Assumption needed only to simplify analysis, not for correctness of algorithm

b bands, r rows/band

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

Example of Bands

Assume the following case:

- ▶ Suppose 100,000 columns of M (100k docs)
- ▶ Signatures of 100 integers (rows)
- ▶ Therefore, signatures take 40Mb
- ▶ Choose $b = 20$ bands of $r = 5$ integers/band
- ▶ **Goal: Find pairs of documents that are at least $s = 0.8$ similar**

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

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:** $\text{sim}(C_1, C_2) = 0.8$
 - ▶ Since $\text{sim}(C_1, C_2) \geq s$, 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)

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:** $\text{sim}(C_1, C_2) = 0.8$
 - ▶ Since $\text{sim}(C_1, C_2) \geq s$, 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:** $(0.8)^5 = 0.328$

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:** $\text{sim}(C_1, C_2) = 0.8$
 - ▶ Since $\text{sim}(C_1, C_2) \geq s$, 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 30% Similar

▶ Find pairs of $\geq s=0.8$ similarity, set $b=20, r=5$

▶ **Assume:** $\text{sim}(C_1, C_2) = 0.3$

- ▶ Since $\text{sim}(C_1, C_2) < s$ we want C_1, C_2 to hash to **No common buckets** (all bands should be different)

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

C_1, C_2 are 30% Similar

▶ Find pairs of $\geq s=0.8$ similarity, set $b=20, r=5$

▶ **Assume:** $\text{sim}(C_1, C_2) = 0.3$

▶ Since $\text{sim}(C_1, C_2) < s$ we want C_1, C_2 to hash to **No common buckets** (all bands should be different)

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

▶ **Probability C_1, C_2 identical in one particular band: $(0.3)^5 = 0.00243$**

C_1, C_2 are 30% Similar

- ▶ Find pairs of $\geq s=0.8$ similarity, set $b=20, r=5$

- ▶ **Assume:** $\text{sim}(C_1, C_2) = 0.3$

- ▶ Since $\text{sim}(C_1, C_2) < s$ we want C_1, C_2 to hash to **No common buckets** (all bands should be different)

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

- ▶ **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 Involves a Tradeoff

► Pick:

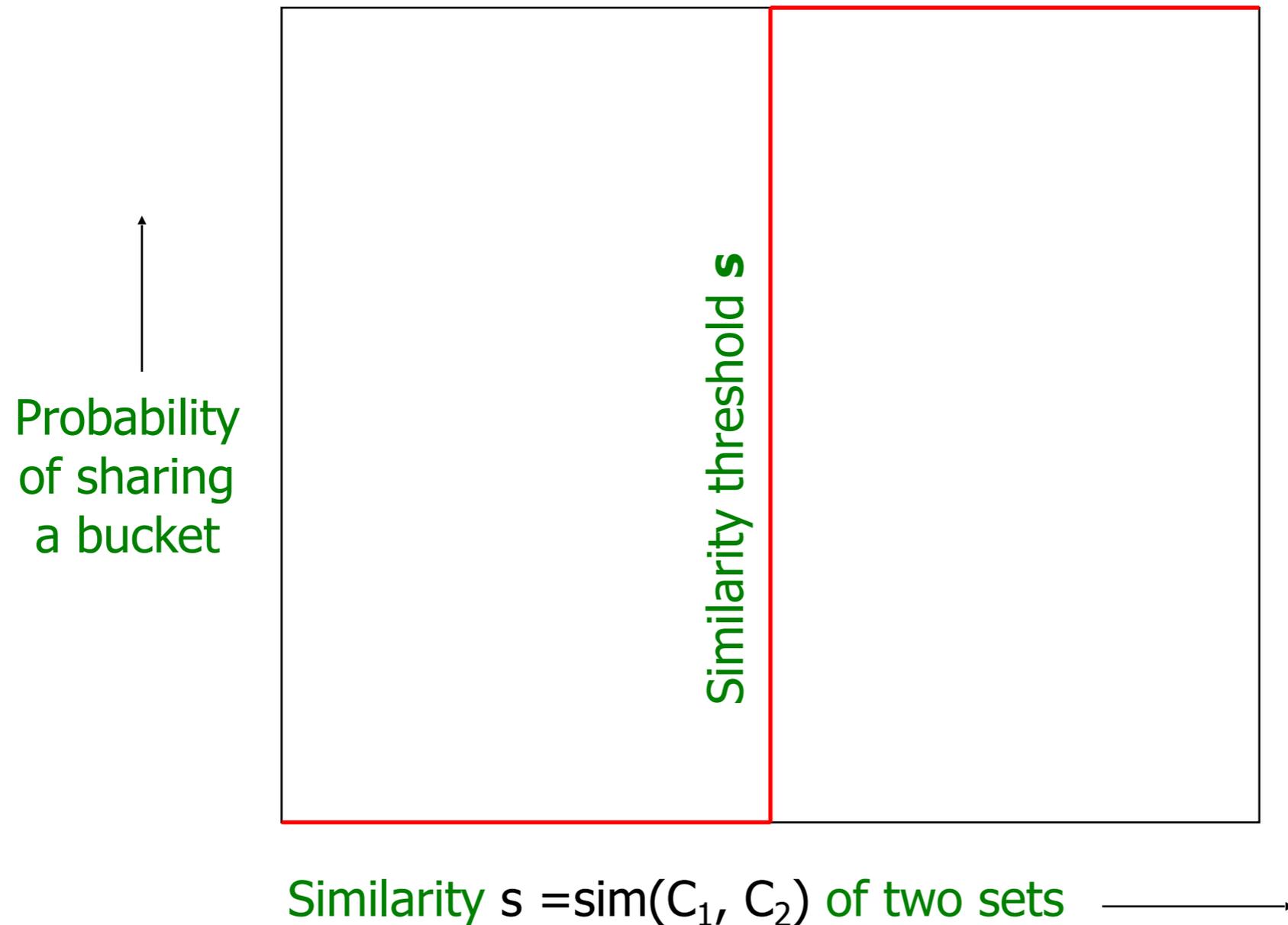
- The number of Min-Hashes (rows of M)
- The number of bands b , and
- The number of rows r per band

to balance false positives/negatives

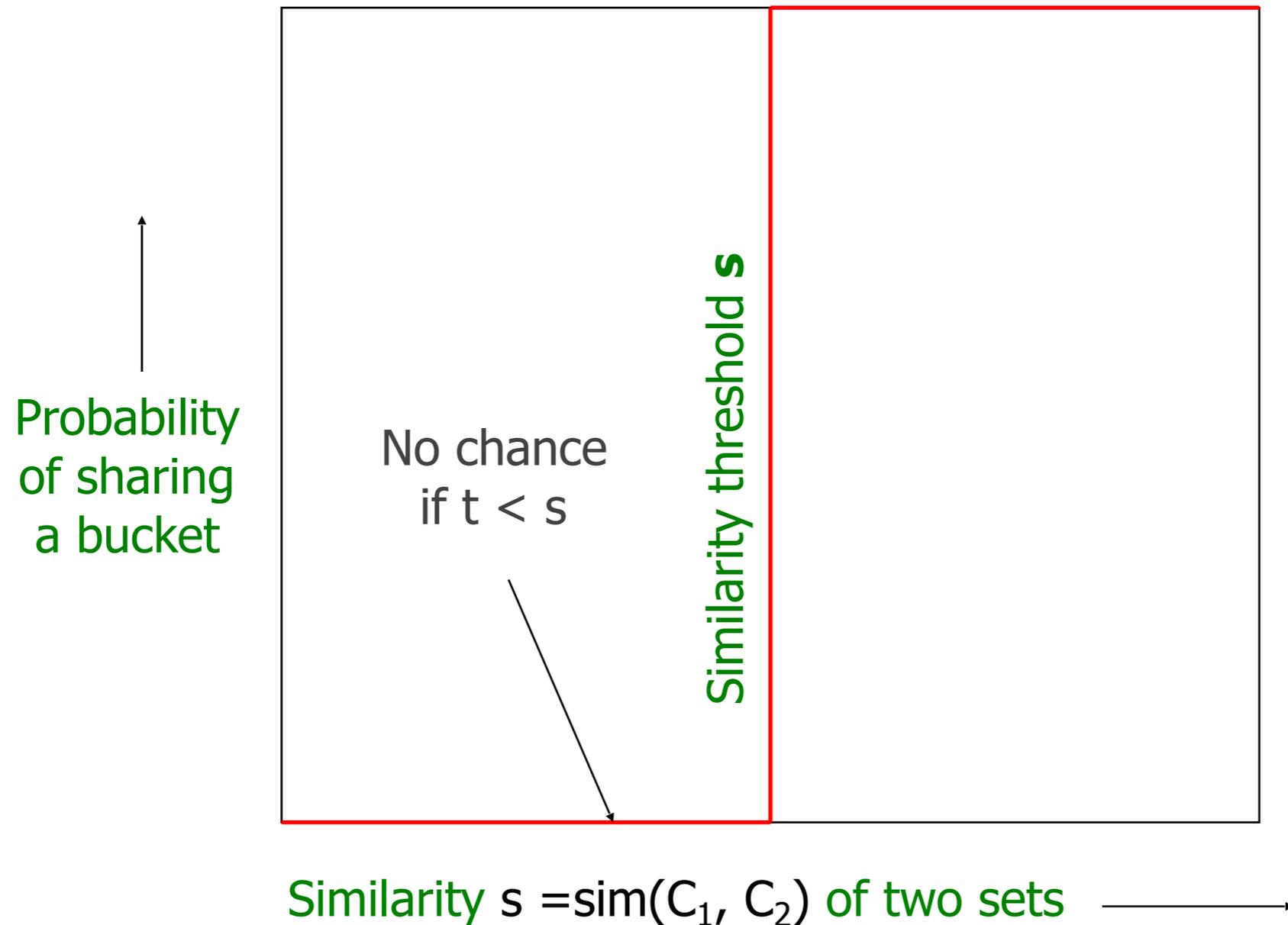
- **Example:** If we had only 15 bands of 5 rows, the number of false positives would go down, but the number of false negatives would go up

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

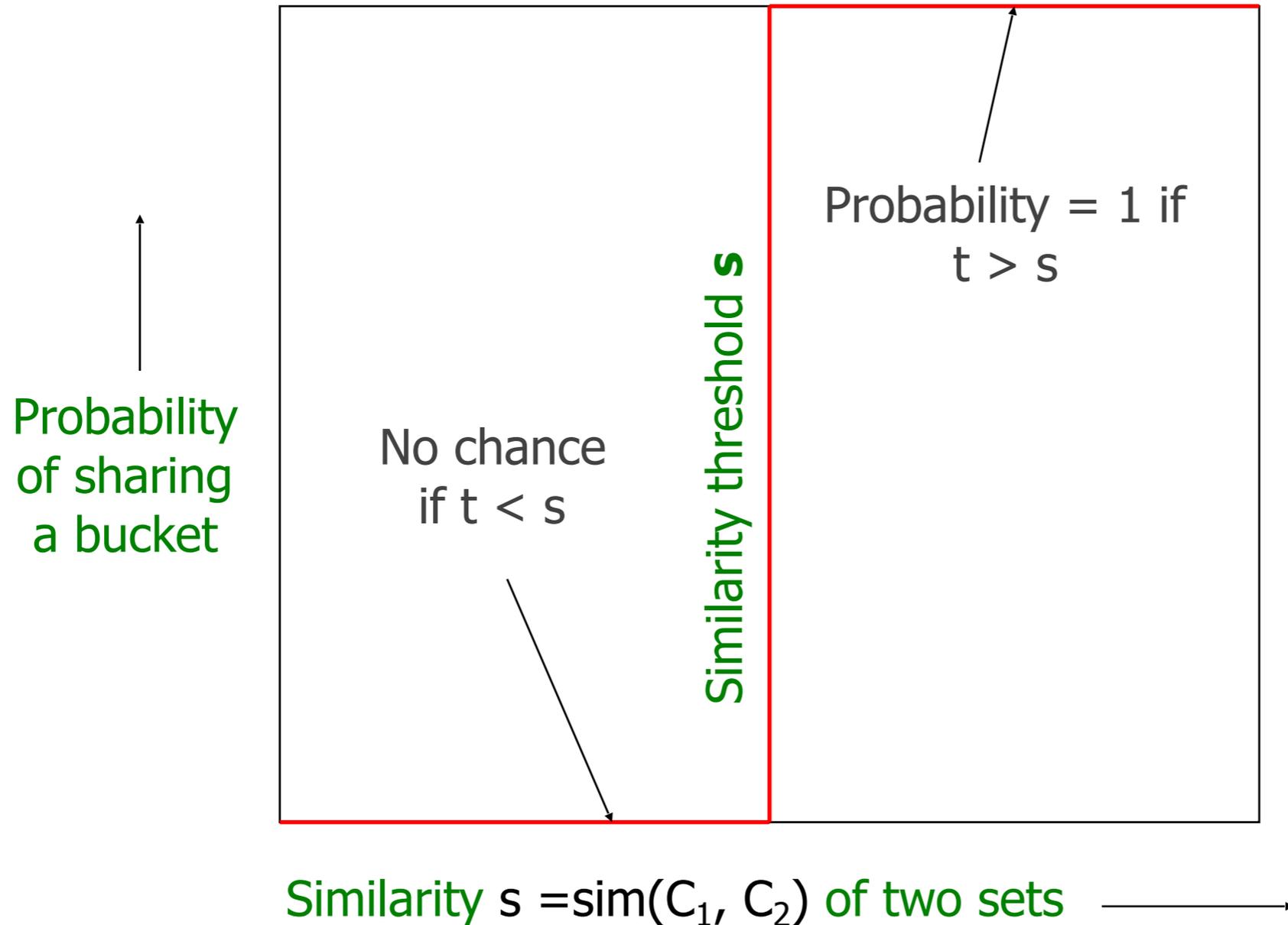
Analysis of LSH – What We Want



Analysis of LSH – What We Want



Analysis of LSH – What We Want



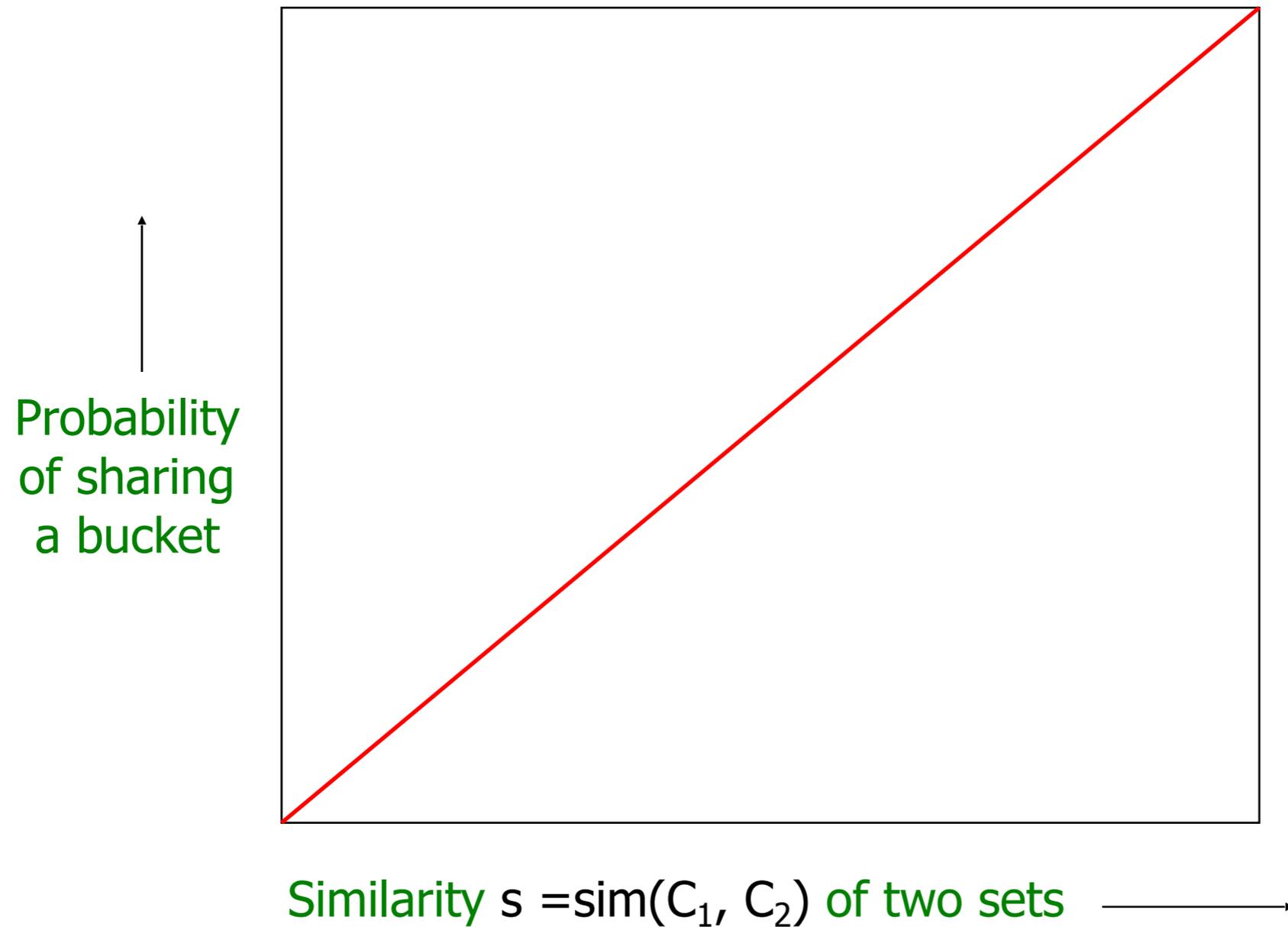
What One Band of One Row Gives You

↑
Probability
of sharing
a bucket

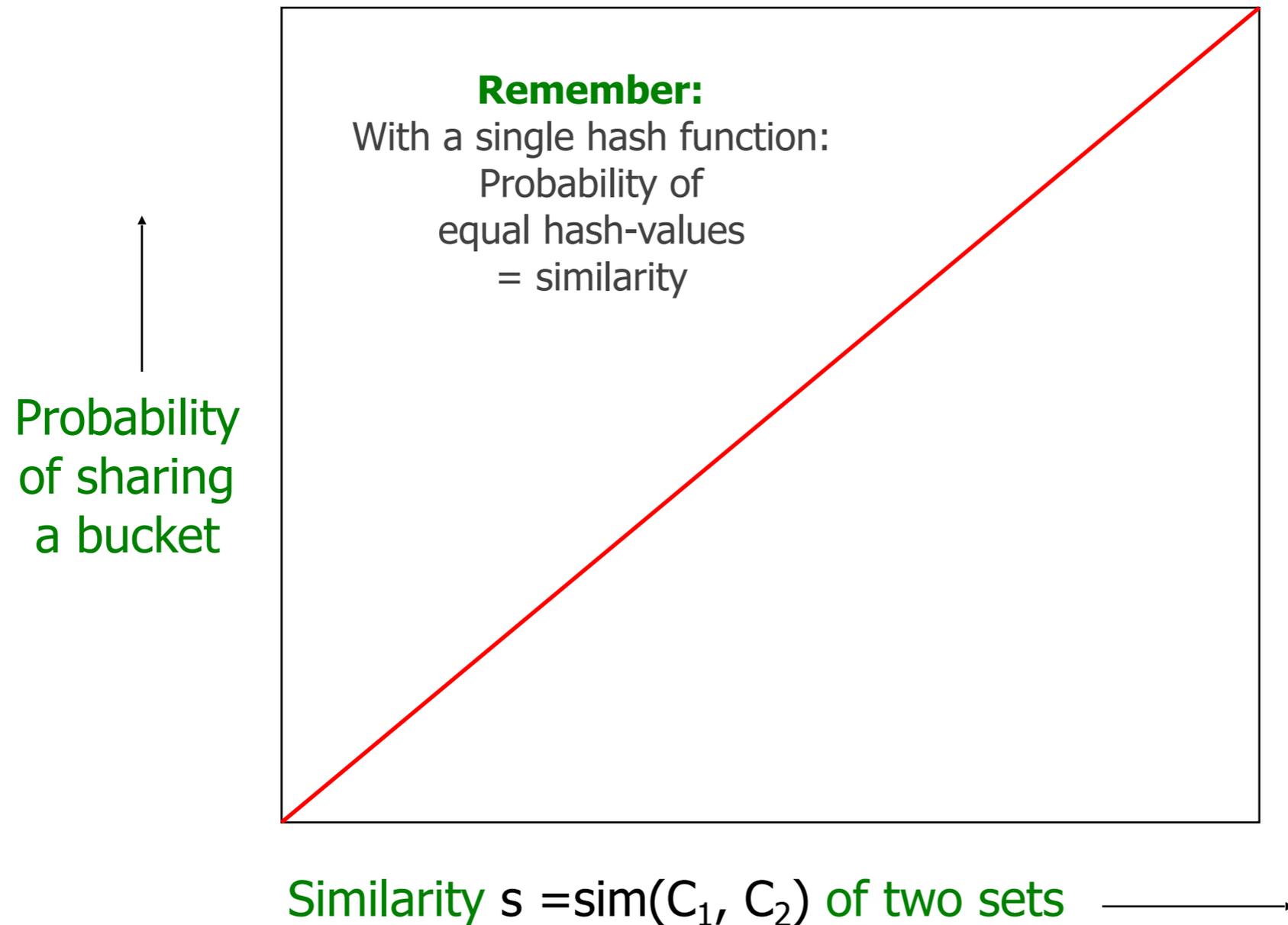


Similarity $s = \text{sim}(C_1, C_2)$ of two sets →

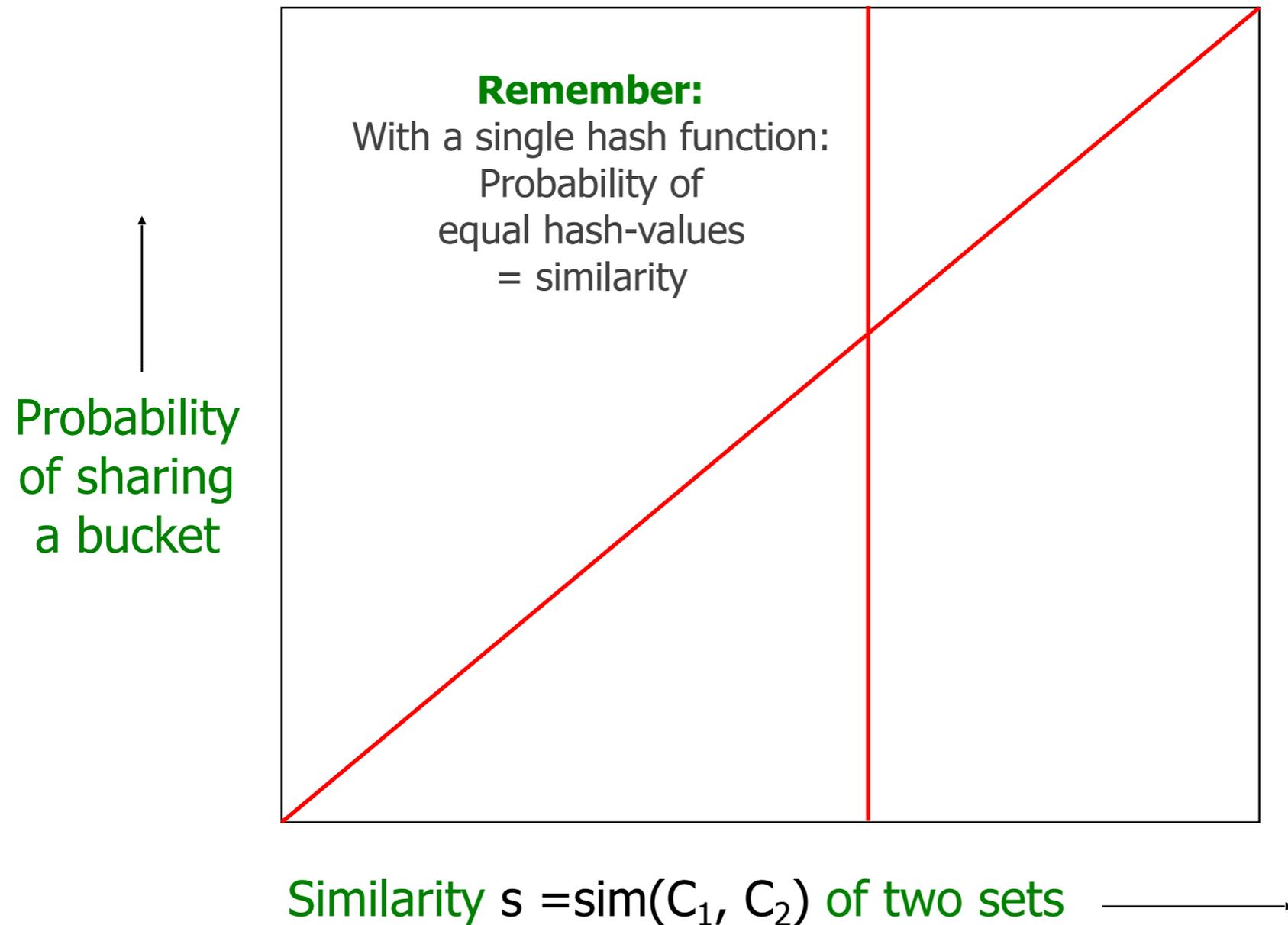
What One Band of One Row Gives You



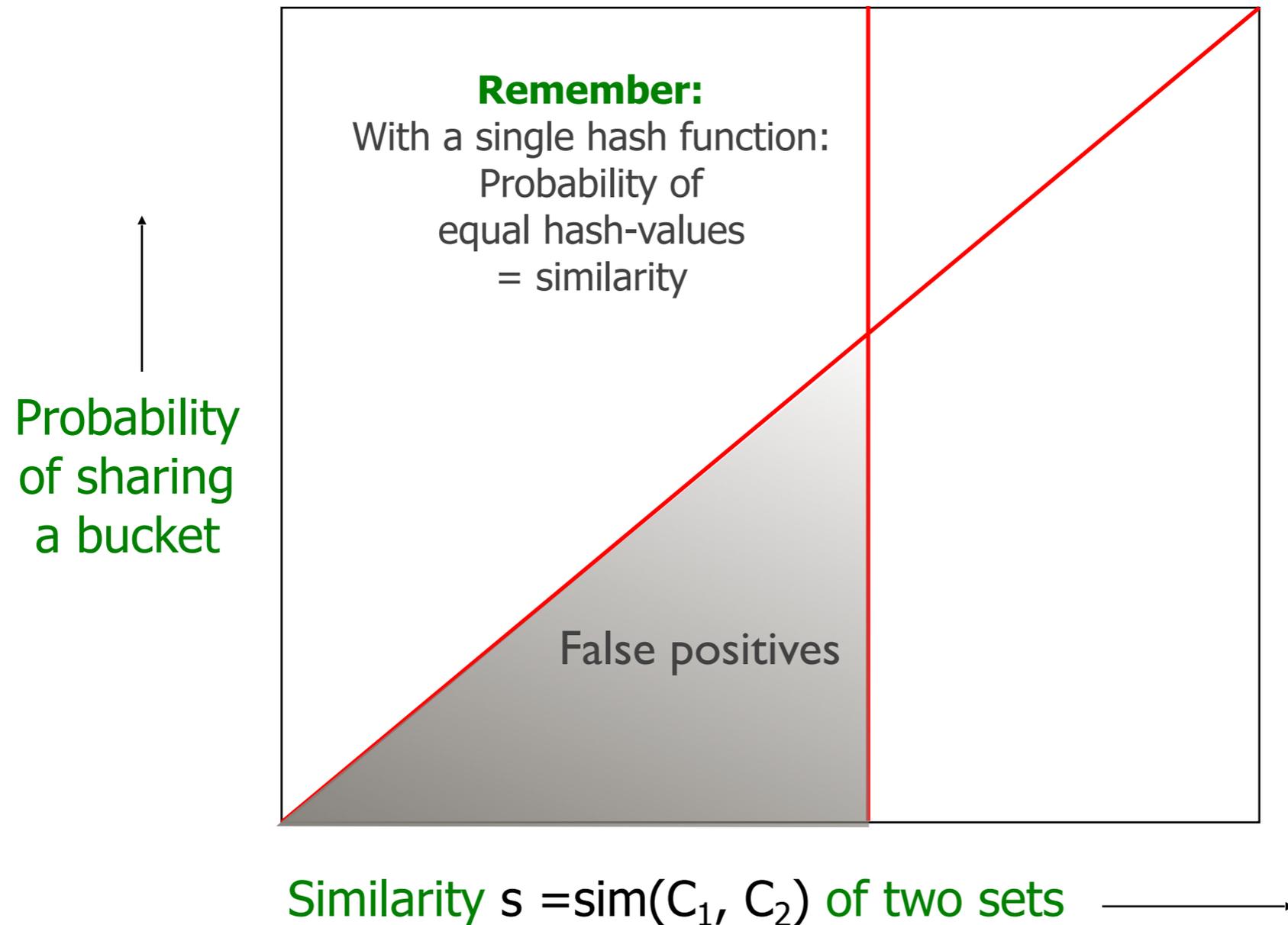
What One Band of One Row Gives You



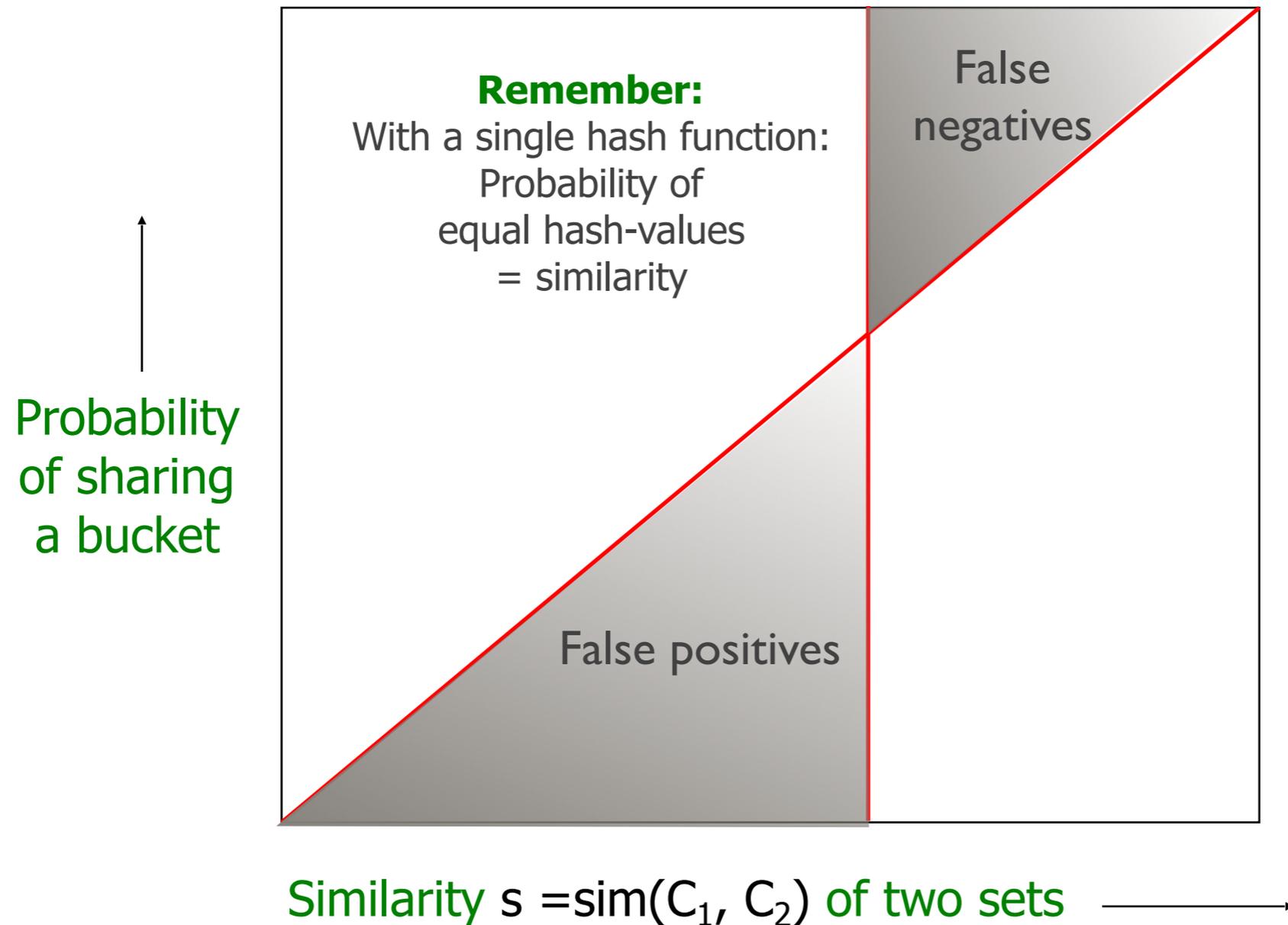
What One Band of One Row Gives You



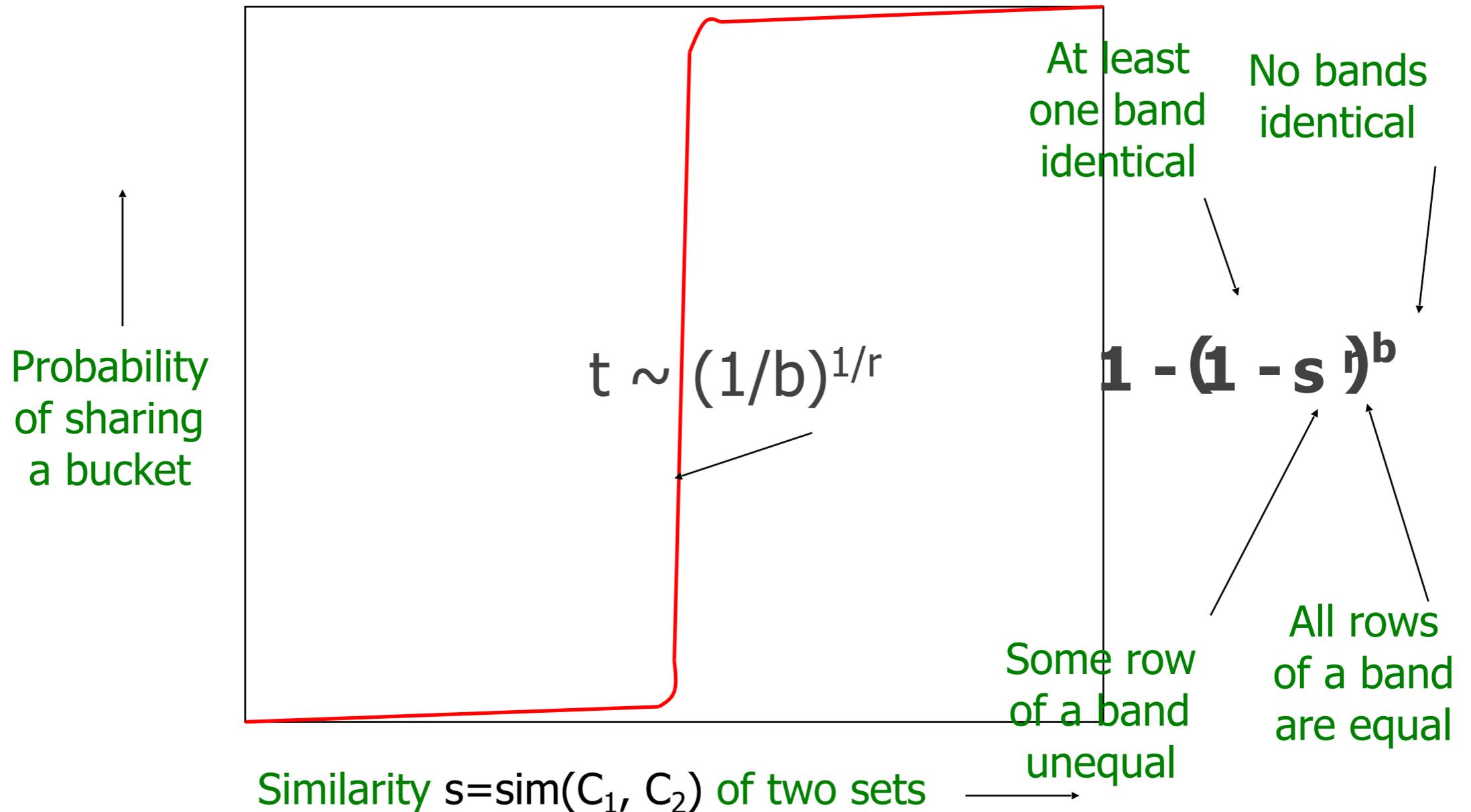
What One Band of One Row Gives You



What One Band of One Row Gives You



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

LSH Summary

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

Outline

4.1. Clustering

4.2. Finding similar documents

4.3. Faceted Search

4.3. Tracking Memes

4.2. Faceted Search

4.2. Faceted Search

dblp.uni-trier.de

CompleteSearch DBLP

a DBLP mirror with extended search capabilities maintained by [Hannah Bast, University of Freiburg](#) (formerly [MPII Saarbrücken](#))

zoomed in on 276 documents ... NEW: get these search results as [XML](#), [JSON](#), [JSONP](#)

[Feedback](#) [Help](#)

efficient query processing

NOTE: The DBLP search has moved
CompleteSearch DBLP has moved to the new domain www.dblp.org. Please update any links or bookmarks you might have set accordingly. The pages under www.informatik.uni-trier.de will eventually be moved there, too. There will be a separate notification about that.

2015	
276	EE Yu Li, Man Lung Yiu: Route-Saver: Leveraging Route APIs for Accurate and Efficient Query Processing at Location-Based Services. <i>IEEE Trans. Knowl. Data Eng. (TKDE)</i> 27(1):235-249 (2015)
2014	
275	EE Haozhou Wang, Kai Zheng, Han Su, Jiping Wang, Shazia Wasim Sadiq, Xiaofang Zhou: Efficient Aggregate Farthest Neighbour Query Processing on Road Networks. <i>ADC</i> 2014:13-25
274	EE He Li, Jaesoo Yoo: An efficient scheme for continuous skyline query processing over dynamic data set. <i>BigComp</i> 2014:54-59
273	EE Heejung Yang, Chin-Wan Chung: Efficient Iceberg Query Processing in Sensor Networks. <i>Comput. J. (CJ)</i> 57(12):1834-1851 (2014)
272	EE Jiajia Li, Botao Wang, Guoren Wang, Xin Bi: Efficient Processing of Probabilistic Group Nearest Neighbor Query on Uncertain Data. <i>DASFAA</i> 2014:436-450
271	EE Nikolaos Nodarakis, Evaggelia Pitoura, Spyros Sioutas, Athanasios K. Tsakalidis, Dimitrios Tsoumakos, Giannis Tzimas: Efficient Multidimensional AkNN Query Processing in the Cloud. <i>DEXA</i> 2014:477-491
270	EE Yunjun Gao, Qing Liu, Baihua Zheng, Gang Chen: On efficient reverse skyline query processing. <i>Expert Syst. Appl. (ESWA)</i> 41(7):3237-3249 (2014)
269	EE Kisung Kim, Bongki Moon, Hyoung-Joo Kim: RG-index: An RDF graph index for efficient SPARQL query processing. <i>Expert Syst. Appl. (ESWA)</i> 41(10):4596-4607 (2014)
268	EE Alfredo Cuzzocrea, José Cecilio, Pedro Furtado: An Effective and Efficient Middleware for Supporting Distributed Query Processing in Large-Scale Cyber-Physical Systems. <i>IDCS</i> 2014:124-135
267	EE Khaled Mohammed Al-Naami, Sadi Evren Seker, Latifur Khan: GISQF: An Efficient Spatial Query Processing System. <i>IEEE CLOUD</i> 2014:681-688
266	EE Jia Liu, Bin Xiao, Kai Bu, Lijun Chen: Efficient distributed query processing in large RFID-enabled supply chains. <i>INFOCOM</i> 2014:163-171
265	EE Yuan-Ko Huang, Lien-Fa Lin: Efficient processing of continuous min-max distance bounded query with updates in road networks. <i>Inf. Sci. (ISCI)</i> 278:187-205 (2014)
264	EE Qiming Fang, Guangwen Yang: Efficient Top-k Query Processing Algorithms in Highly Distributed Environments. <i>JCP</i> 9(9):2000-2006 (2014)
263	EE Jiping Wang, Kai Zheng, Hoyoung Jeung, Haozhou Wang, Bolong Zheng, Xiaofang Zhou: Cost-Efficient Spatial Network Partitioning for Distance-Based Query Processing. <i>MDM</i> 2014:13-22
262	EE Sanjay Chatterji, G. S. Sreedhara, Maunendra Sankar Desarkar: An Efficient Tool for Syntactic Processing of English Query Text. <i>MIKE</i> 2014:278-287
261	EE Merih Seran Uysal, Christian Beecks, Thomas Seidl: On Efficient Query Processing with the Earth Mover's Distance. <i>PIKM@CIKM</i> 2014:25-32
260	EE Fabian Nagel, Gavin M. Bierman, Stratis D. Viglas: Code Generation for Efficient Query Processing in Managed Runtimes. <i>PVLDB</i> 7(12):1095-1106 (2014)
259	EE Tomas Karnagel, Matthias Hille, Mario Ludwig, Dirk Habich, Wolfgang Lehner, Max Heimel, Volker Markl: Demonstrating efficient query processing in heterogeneous environments. <i>SIGMOD</i> 2014:693-696
258	EE Junfeng Zhou, Zhifeng Bao, Wei Wang, Jinjia Zhao, Xiaofeng Meng: Efficient query processing for XML keyword queries based on the IDList index. <i>VLDB J. (VLDB)</i> 23(1):25-50 (2014)
2013	
257	EE Jianbin Qin, Wei Wang, Chuan Xiao, Yifei Lu, Xuemin Lin, Haixun Wang: Asymmetric signature schemes for efficient exact edit similarity query processing. <i>ACM Trans. Database Syst. (TODS)</i> 38(3):16 (2013)

[more]

Refine by AUTHOR

[Hans-Peter Kriegel](#) (11)
[Guoren Wang](#) (8)
[Qing Li](#) (7)
[Yunjun Gao](#) (7)
[top 4] [top 50] [top 250]

Refine by VENUE

[IEEE Trans. Knowl. Data Eng. \(TKDE\)](#) (15)
[ICDE](#) (12)
[CIKM](#) (10)
[DASFAA](#) (10)
[top 4] [top 50] [all 155]

Refine by YEAR

2015 (1)
2014 (18)
2013 (21)
2012 (19)
[top 4] [all 30]

Refine by TYPE

[Conference](#) (181)
[Journal](#) (90)
[Book](#) (3)
[CoRR](#) (2)
[top 4]

4.2. Faceted Search

The screenshot shows an Amazon search results page for 'digital camera'. The search bar at the top contains 'digital camera' and the results are sorted by 'Relevance'. The left sidebar contains several faceted search filters:

- Show results for:** Any Category, Electronics, Camera & Photo, Digital Cameras (with sub-categories like Point & Shoot Digital Cameras, DSLR Camera Bundles, etc.), and See more.
- Refine by:** International Shipping (Ship to Germany), Eligible for Free Shipping (Free Shipping by Amazon), Camera Expert Reviews (DPReview Tested), Digital Camera Megapixels (ranging from 7.9 MP to 36 MP & Up), Optical Zoom (ranging from 2.9x to 13x & Up), Point & Shoot Digital Camera Video Resolution (1080 P, 720 P, 480 P), Point & Shoot Digital Camera Color (various color swatches), and Avg. Customer Review (5 stars & Up, 4 stars & Up, 3 stars & Up).

The main content area displays a grid of product listings:

- Nikon Coolpix L330 - 20.2 MP Digital Camera with 26x zoom 35mm NIKKOR VR lens and FULL HD 720p (Black)**: \$149.99 (was \$249.99), Prime, Get it by Monday, Jan 19. #1 Best Seller in Compact System Cameras.
- Sony W800/B 20.1 MP Digital Camera (Black)**: \$89.00. FREE Shipping.
- Nikon COOLPIX L830 16 MP CMOS Digital Camera with 34x Zoom NIKKOR Lens and Full 1080p HD Video (Black)**: \$195.99 (was \$299.95), Prime, Get it by Monday, Jan 19.
- Nikon Coolpix S3600 Digital Camera (Silver) with 8GB Card + Case + Accessory Kit**: \$54.95 used & new (1 offer).
- Sony W800/S 20 MP Digital Camera (Silver)**: \$89.00. FREE Shipping.
- Sony DSCW830/B 20.1 MP Digital Camera with 2.7-Inch LCD (Black)**: \$74.99 used & new (23 offers). #1 Best Seller in Digital Point & Shoot Cameras.
- Canon PowerShot SX520 16Digital Camera with 42x Optical Image Stabilized Zoom with 3-Inch LCD (Black)**: \$199.00 (was \$329.00), Prime.
- Nikon COOLPIX L830 16 MP CMOS Digital Camera with 34x Zoom NIKKOR Lens and Full 1080p HD Video (Red)**: \$195.00 (was \$299.96), Prime, Get it by Monday, Jan 19.
- Canon EOS Rebel T5 18MP EF-S Digital SLR Camera USA warranty with canon EF-S 18-55mm f/3.5-5.6 IS [Image Stabilizer...]**: \$599.95 (was \$899.95), Prime.

4.2. Faceted Search

Flamenco

Refine your search further within these categories: These terms define your current search. Click the to remove a term.

Media (group results)
costume (3), drawing (2), lithograph (1), woodcut (6), woven object (2)

Location: all > Asia
Afghanistan (1), China (4), China or Tibet? (3), India (2), Japan (13), Russia (1), Turkey (3), Turkmenistan (1)

Date (group results)
17th century (3), 18th century (3), 19th century (10), 20th century (3), date ranges spanning multiple centuries (7), date unknown (2)

Themes (group results)
music, writing, and sport (5), nautical (1), religion (2)

Objects (group results)
clothing (5), food (1), furnishings (4), timepieces (1)

Nature (group results)
bodies of water (3), fish (1), flowers (2), geological formations (1), heavens (3), invertebrates and arthropods (1), mammals (2), plant material (3), trees (1)

Places and Spaces (group results)
bridges (1), buildings (1), dwellings (1)

Location: Asia [start a new search](#)

Shapes, Colors, and Materials: fabrics

Search all items within current results

28 items (grouped by location) [view ungrouped items](#)

Afghanistan 1


Girl's Ceremonia...
no artist
20th century

China 4


4 boats on lake...
Anonymous
post World War II

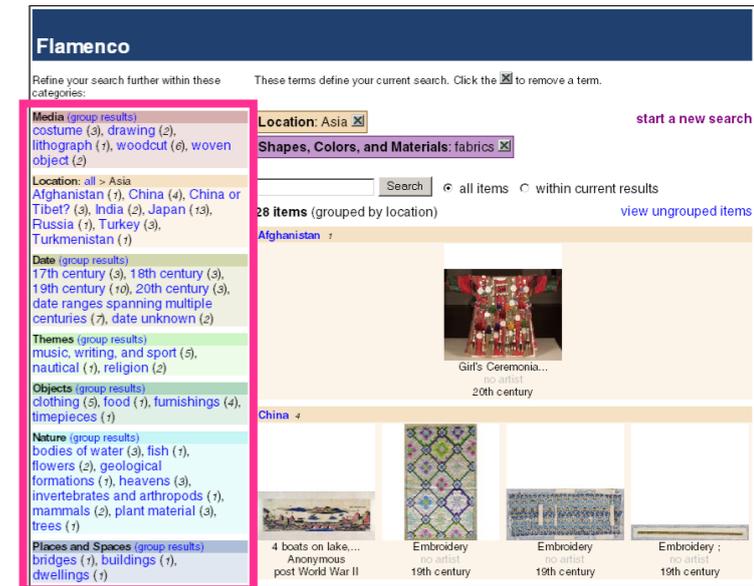

Embroidery
no artist
19th century


Embroidery
no artist
19th century


Embroidery ;
no artist
19th century

Faceted Search

- ▶ **Faceted search [3,7]** supports the user in exploring/navigating a collection of documents (e.g., query results)
- ▶ **Facets** are orthogonal sets of categories that can be flat or hierarchical, e.g.:
 - ▶ topic: arts & photography, biographies & memoirs, etc.
 - ▶ origin: Europe > France > Provence, Asia > China > Beijing, etc.
 - ▶ price: 1–10\$, 11–50\$, 51–100\$, etc.
- ▶ Facets are **manually curated** or **automatically derived** from meta-data



Automatic Facet Generation

- ▶ Need to manually curate facets prevents their application for **large-scale** document collections with **sparse meta-data**
- ▶ Dou et al. [3] investigate how facets can be **automatically mined** in a **query-dependent manner** from pseudo-relevant documents
- ▶ Observation: **Categories** (e.g., brands, price ranges, colors, sizes, etc.) are typically **represented as lists** in web pages
- ▶ Idea: Extract lists from web pages, rank and cluster them, and use the **consolidated lists as facets**

List Extraction

- ▶ Lists are extracted from web pages using several patterns
 - ▶ **enumerations** of items in text (e.g., we serve *beef, lamb, and chicken*) via: `item {, item} * (and|or) {other} item`
 - ▶ **HTML form elements** (`<SELECT>`) and **lists** (``) ignoring instructions such as “select” or “chose”
 - ▶ as rows and columns of **HTML tables** (`<TABLE>`) ignoring header and footer rows
- ▶ Items in extracted lists are **post-processed**, removing non-alphanumeric characters (e.g., brackets), converting them to lower case, and removing items longer than 20 terms

List extraction examples

SELECT:

```
<select name="ProductFinder2" id="ProductFinder2" >
<option value="WatchBrands.htm"> Watch Brands</option>
<option value="Brands-Accutron.htm"> Accutron</option>
<option value="Brands-Bulova.htm"> Bulova</option>
<option value="Brands-Caravelle.htm"> Caravelle</option>
<option value="Brands-Seiko.htm"> Seiko</option></select>
```

UL:

```
<ul><li><a href="/rst.asp?q=dive"> Dive</a></li>
<li><a href="/rst.asp?q=titanium"> Titanium</a></li>
<li><a href="/rst.asp?q=automatic"> Automatic</a></li>
<li><a href="/rst.asp?q=quartz"> Quartz</a></li>
<li><a href="/rst.asp?q=gold"> Gold</a></li></ul>
```

TABLE:

```
<table width="100%">
<tr><td width="10%"></td><td> White</td></tr>
<tr><td></td><td height="20"> Red</td></tr>
<tr><td></td><td height="20"> Black</td></tr>
<tr><td></td><td height="20"> Pink</td></tr>
<tr><td height="4" colspan="2"></td></tr></table>
```

List Weighting

- ▶ Some of the extracted lists are **spurious** (e.g., from HTML tables)
- ▶ Intuition: Good lists consist of items that are **informative** to the query, i.e., are **mentioned in many** pseudo-relevant documents
- ▶ **Lists weighted** taking into account a document matching weight S_{DOC} and their average inverse document frequency S_{IDF}

$$S_l = S_{DOC} \cdot S_{IDF}$$

- ▶ **Document matching weight S_{DOC}**

$$S_{DOC} = \sum_{d \in R} (s_d^m \cdot s_d^r)$$

with s_d^m as fraction of list items mentioned in document d
and s_d^r as importance of document d (estimated as $\text{rank}(d)^{-1/2}$)

List Weighting

- ▶ Average inverse document S_{IDF} is defined as

$$S_{IDF} = \frac{1}{|l|} \sum_{i \in l} idf(i)$$

- ▶ Problem: Individual lists (extracted from a single document) may still contain noise, be incomplete, or overlap with other lists
- ▶ Idea: Cluster lists containing similar items to consolidate them and form dimensions that can be used as facets

List Clustering

- ▶ Distance between two lists is defined as

$$d(l_1, l_2) = 1 - \frac{|l_1 \cap l_2|}{\min\{|l_1|, |l_2|\}}$$

- ▶ Complete-linkage distance between two clusters

$$d(c_1, c_2) = \max_{l_1 \in c_1, l_2 \in c_2} d(l_1, l_2)$$

- ▶ Greedy clustering algorithm

- ▶ pick most important not-yet-clustered list
- ▶ add nearest lists while cluster diameter is smaller than Dia_{max}
- ▶ save cluster if total weight is larger than W_{min}

Dimension and Item Ranking

- ▶ Problem: In which order to present dimensions and items therein?

- ▶ Importance of a dimension (cluster) is defined as

$$S_c = \sum_{s \in Sites(c)} \max_{l \in c, l \in s} S_l$$

favoring dimensions grouping lists with high weight

- ▶ Importance of an item within a dimension defined as

$$S_{i|c} = \sum_{s \in Sites(c)} \frac{1}{\sqrt{AvgRank(c, i, s)}}$$

favoring items which are often ranked high within containing lists

Facet Generation Example

Search Results

1. [Watches from Overstock.com](#)
Watches by Seiko, Omega, Movado, and more brand names. \$2.95 shipping and product reviews on men's watches and women's watches.
www.overstock.com/watches/31/store.html
2. [Shop Watches, 50+ brands, Invicta, Soiko, Citizen, Movado, Tag Heuer](#)
Omega Watches, Tag Heuer Watches, Breitling Watches, Invicta Watches, TechnoMarie Watches, Seiko Watches. Men's watches & women's watches all authentic and new..
www.worldofwatches.com
3. [Watches, Men's Watches, Ladies' Watches, Dive Watches, Aviation ...](#)
Online watches dealer with discounted prices, free shipping and more than 10,000 watches in stock. Authorized dealer of more than 50 brands of watches including Citizen watches ...
www.ewatches.com
4. [Amazon.com: Watches, Men's Watches, Women's Watches, Kids' Watches](#)
Online Shopping for Watches: Men's Watches, Women's Watches, Kids' Watches, Luxury Watches, Sport Watches, Fashion Watches, Watch Accessories, and more.
www.amazon.com/Watches-Mens-Womens-Kids-Accessories/b?node=377110011

List Extraction

d ₁	men's watches, women's watches, luxury watches, ... automatic, bracelet, ceramic, ... breitling, cartier, omega, tag heuer ...
d ₂	accessories watches, activa watches, ... 299.99, 349.99, 423.99,
...	...

Dimension and Item Ranking

1.	cartier, breitling, omega, citizen, tag heuer, bulova, casio, rolex, ...
2.	men's, women's, kids, unisex, ...
3.	analog, digital, chronograph, analog digital, ...
4.	dress, casual, sport, fashion, luxury, ...
5.	black, blue, white, green, red, brown, ...
...	...

List Clustering

C ₁	breitling, cartier, omega, tag heuer ...
C ₂	men's, women's, luxury, ... men's, women's, unisex men's, women's, kids, accessories ...
C ₃	automatic, bracelet, ceramic,
...	...

List Weighting

breitling, cartier, omega, tag heuer	93.7
...	...
automatic, bracelet, ceramic, ...	53.1
...	...
men's watches, women's watches, ...	47.9
...	...
accessories watches, activa watches,	42.8
...	...
299.99, 349.99, 423.99, ...	2.7

Anecdotal Results

► Dimensions mined from top-100 of commercial search engine

query: **watches**

1. cartier, breitling, omega, citizen, tag heuer, bulova, casio, rolex, audemars piguet, seiko, accutron, movado, fossil, gucci, ...
2. men's, women's, kids, unisex
3. analog, digital, chronograph, analog digital, quartz, mechanical, manual, automatic, electric, dive, ...
4. dress, casual, sport, fashion, luxury, bling, pocket, ...
5. black, blue, white, green, red, brown, pink, orange, yellow, ...

query: **lost**

1. season 1, season 6, season 2, season 3, season 4, season 5
2. matthew fox, naveen andrews, evangeline lilly, josh holloway, jorge garcia, daniel dae kim, michael emerson, terry o'quinn, ...
3. jack, kate, locke, sawyer, claire, sayid, hurley, desmond, boone, charlie, ben, juliet, sun, jin, ana, lucia ...
4. what they died for, across the sea, what kate does, the candidate, the last recruit, everybody loves hugo, the end, ...

query: **lost season 5**

1. because you left, the lie, follow the leader, jughead, 316, dead is dead, some like it hoth, whatever happened happened, the little prince, this place is death, the variable, ...
2. jack, kate, hurley, sawyer, sayid, ben, juliet, locke, miles, desmond, charlotte, various, sun, none, richard, daniel
3. matthew fox, naveen andrews, evangeline lilly, jorge garcia, henry ian cusick, josh holloway, michael emerson, ...
4. season 1, season 3, season 2, season 6, season 4

query: **flowers**

1. birthday, anniversary, thanksgiving, get well, congratulations, christmas, thank you, new baby, sympathy, fall
2. roses, best sellers, plants, carnations, lilies, sunflowers, tulips, gerberas, orchids, iris
3. blue, orange, pink, red, purple, white, green, yellow

query: **what is the fastest animals in the world**

1. cheetah, pronghorn antelope, lion, thomson's gazelle, wildebeest, cape hunting dog, elk, coyote, quarter horse
2. birds, fish, mammals, animals, reptiles
3. science, technology, entertainment, nature, sports, lifestyle, travele, gaming, world business

query: **the presidents of the united states**

1. john adams, thomas jefferson, george washington, john tyler, james madison, abraham lincoln, john quincy adams, william henry harrison, martin van buren, james monroe, ...
2. the presidents of the united states of america, the presidents of the united states ii, love everybody, pure frosting, these are the good times people, freaked out and small, ...
3. kitty, lump, peaches, dune buggy, feather pluckn, back porch, kick out the jams, stranger, boll weevil, ca plane pour moi, ...
4. federalist, democratic-republican, whig, democratic, republican, no party, national union, ...

query: **visit beijing**

1. tiananmen square, forbidden city, summer palace, temple of heaven, great wall, beihai park, hutong
2. attractions, shopping, dining, nightlife, tours, travel tip, transportation, facts

query: **cikm**

1. databases, information retrieval, knowledge management, industry research track
2. submission, important dates, topics, overview, scope, committee, organization, programme, registration, cfp, publication, programme committee, organisers, ...
3. acl, kdd, chi, sigir, www, icml, focs, ijcai, osdi, sigmod, sosp, stoc, uist, vldb, wsdm, ...

Outline

4.1. Clustering

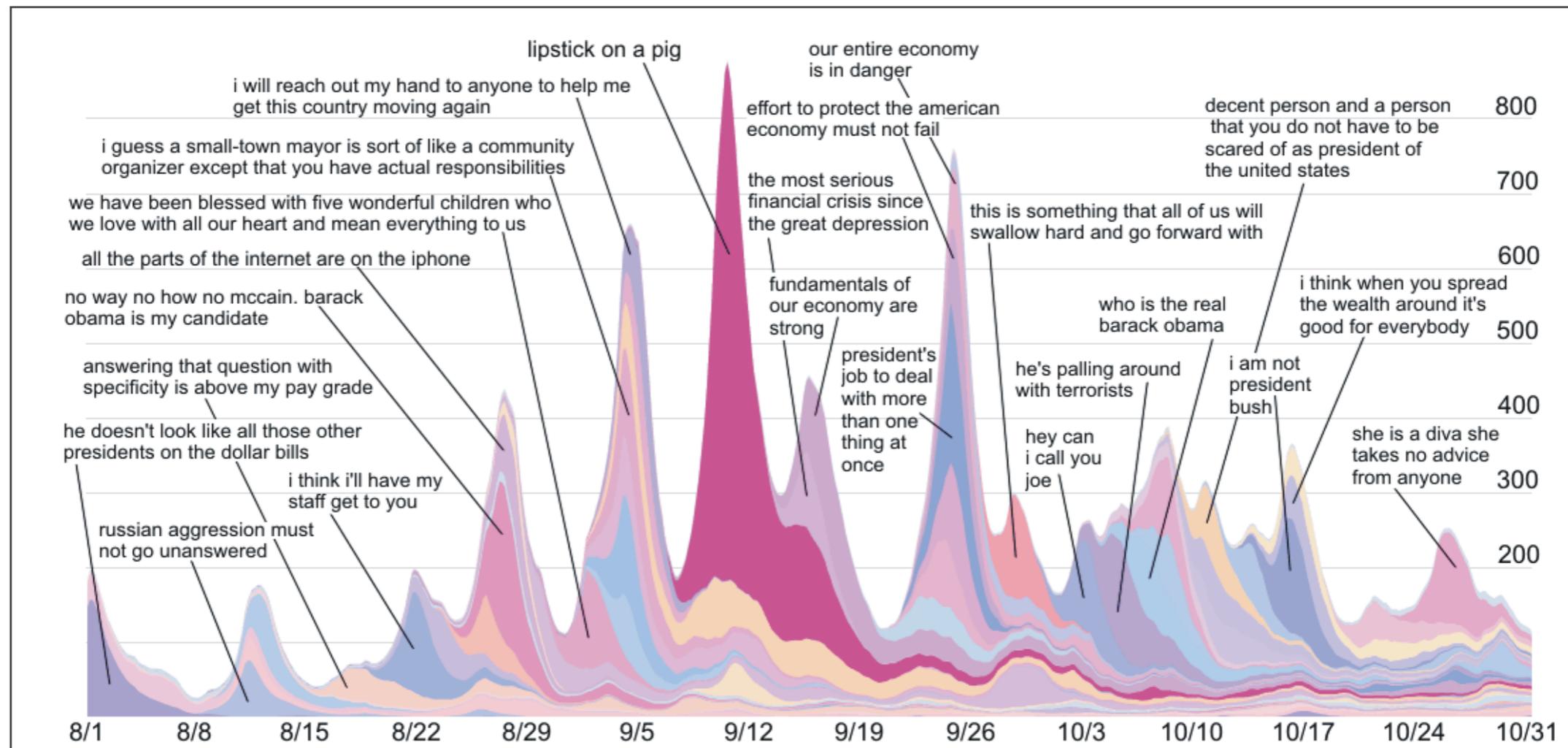
4.2. Finding similar documents

4.2. Faceted Search

4.3. Tracking Memes

4.3. Tracking Memes

- ▶ Leskovec et al. [5] track memes (e.g., “lipstick on a pig”) and visualize their volume in traditional news and blogs

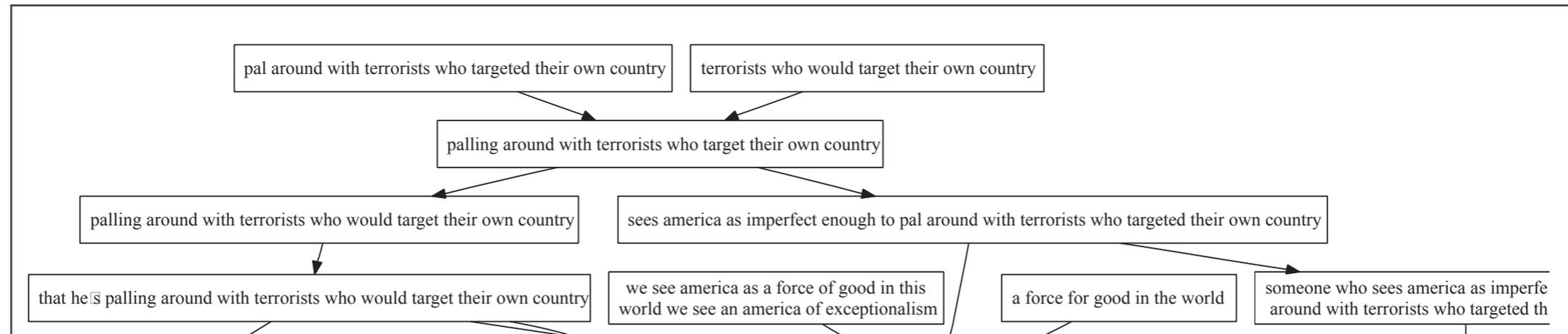


- ▶ Demo: <http://www.memetracker.org>

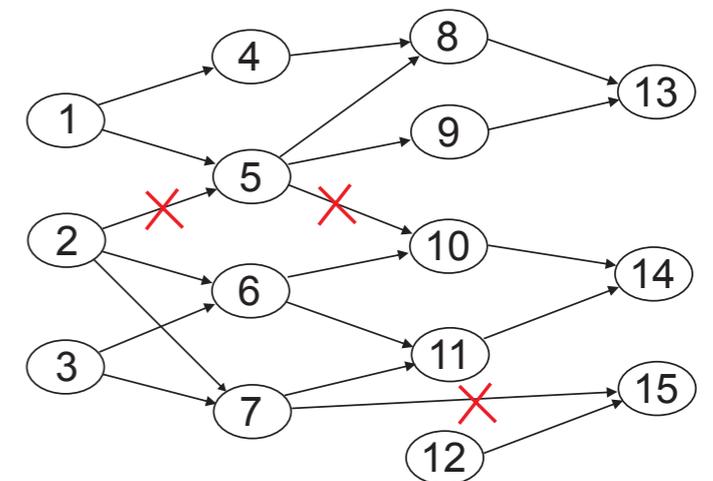
Phrase Graph Construction

- ▶ Problem: Memes are often modified as they spread, so that first **all mentions of the same meme** need to be identified
- ▶ Construction of a phrase graph $G(V, E)$:
 - ▶ vertices V correspond to mentions of a meme that are reasonably long and occur often enough
 - ▶ edge (u, v) exists if meme mentions u and v
 - ▶ u is **strictly shorter** than v
 - ▶ either: have **small directed token-level edit distance** (i.e., u can be transformed into v by adding at most ϵ tokens)
 - ▶ or: have a **common word sequence** of length at least k
 - ▶ **edge weights** based on **edit distance** between u and v and how often v occurs in the document collection

Phrase Graph Partitioning

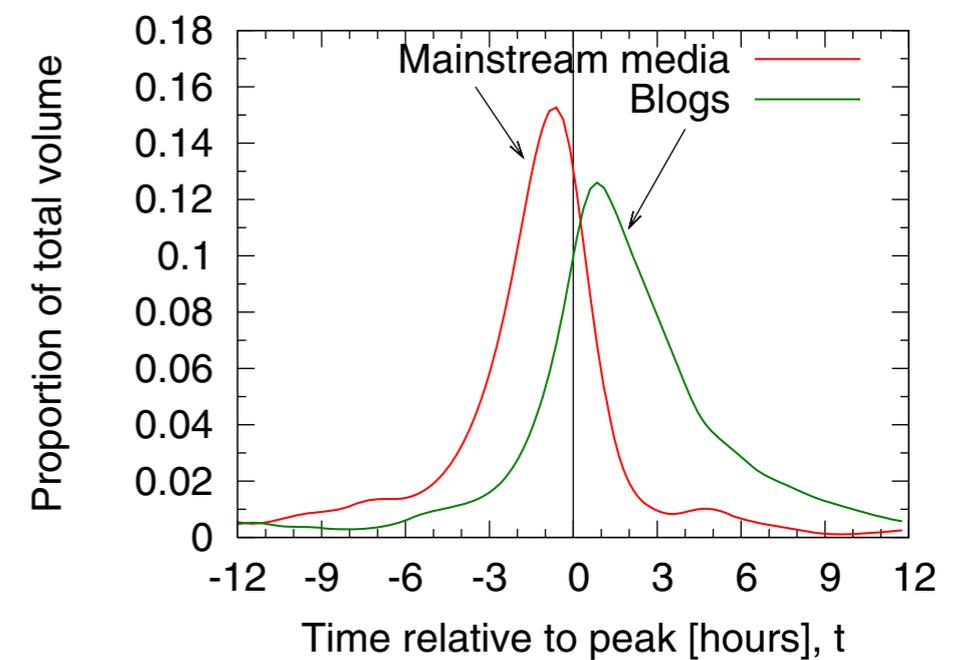
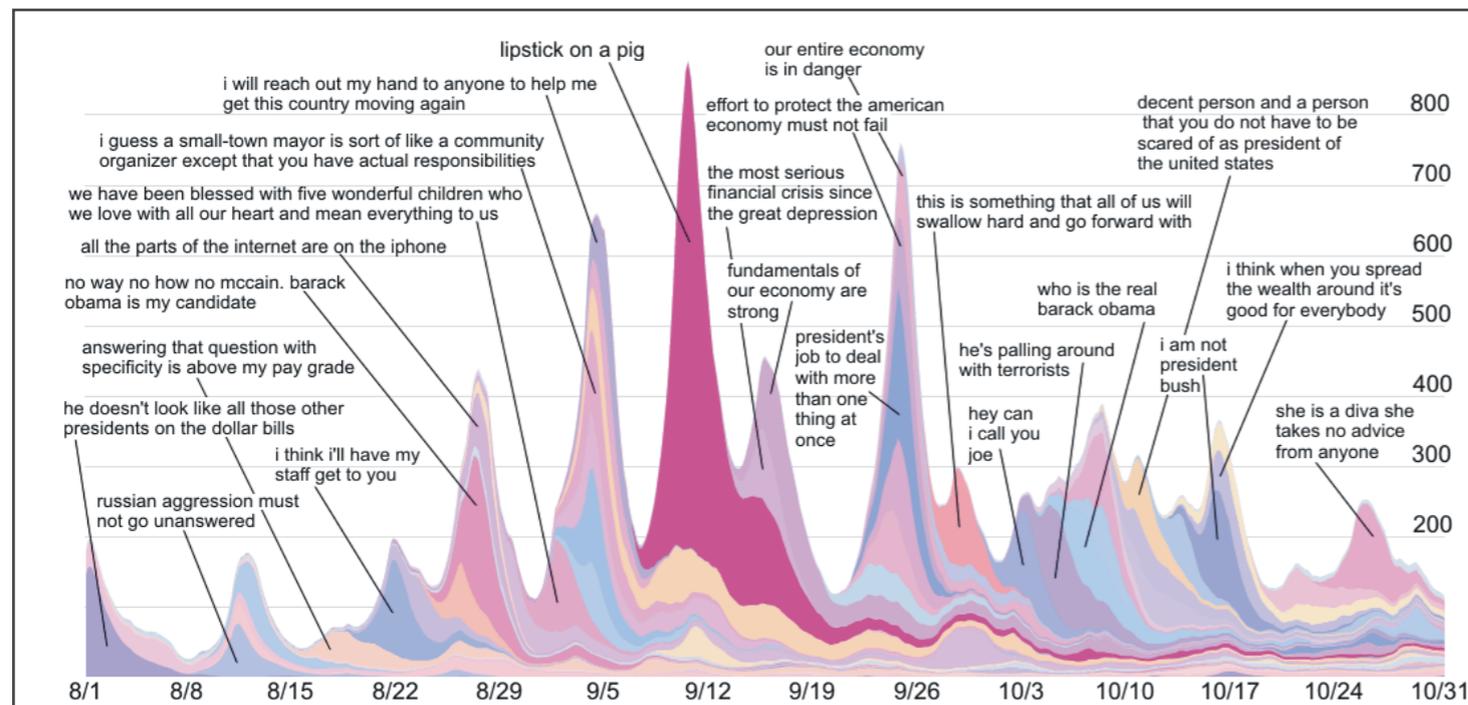


- ▶ Phrase graph is an **directed acyclic graph (DAG)** by construction
- ▶ Partition $G(V, E)$ by deleting a set of edges having minimum total weight, so that each resulting component is **single-rooted**
- ▶ Phrase graph partitioning is *NP*-hard, hence addressed by **greedy heuristic algorithm**



Applications

- ▶ Clustering of meme mentions allows for insightful analyses, e.g.:
 - ▶ volume of meme per time interval
 - ▶ peak time of meme in traditional news and social media
 - ▶ time lag between peak times in traditional news and social media



Summary

- ▶ **Clustering** groups similar documents; *k*-Means can be implemented efficiently by leveraging established IR methods
- ▶ **Minhashing with LSH** provides an efficient way to find similar documents
- ▶ **Faceted search** uses orthogonal sets of categories to allow users to explore/navigate a set of documents (e.g., query results)
- ▶ **Mememes can be tracked** and allow for insightful analyses of media attention and time lag between traditional media and blogs

References

- [1] **A. Broder, L. Garcia-Pueyo, V. Josifovski, S. Vassilvitskii, S. Venkatesan:** *Scalable k-Means by Ranked Retrieval*, WSDM 2014
- [3] **Z. Dou, S. Hu, Y. Luo, R. Song, J.-R. Wen:** *Finding Dimensions for Queries*, CIKM 2011
- [4] **M. Hearst:** *Clustering Versus Faceted Categories for Information Exploration*, CACM 49(4), 2006
- [5] **J. Leskovec, L. Backstrom, J. Kleinberg:** *Meme-tracking and the Dynamics of the News Cycle*, KDD 2009
- [6] **R. Swan and J. Allan:** *Automatic Generation of Timelines*, SIGIR 2000
- [7] **K.-P. Yee, K. Swearingen, K. Li, M. Hearst:** *Faceted Metadata for Image Search and Browsing*, CHI 2003

For LSH refer to the Mining of Massive Datasets Chapter 3 <http://infolab.stanford.edu/~ullman/mmds/book.pdf>

LSH related slides were borrowed from <http://i.stanford.edu/~ullman/cs246slides/LSH-I.pdf>

Some slides were borrowed from Prof. Klaus Berberich as well