

CS60021: Scalable Data Mining

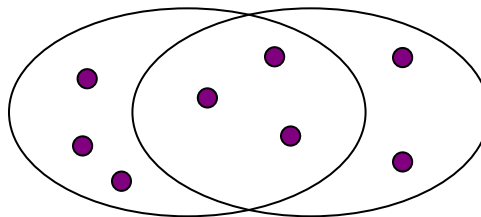
Similarity Search and Hashing

Sourangshu Bhattacharya

Finding Similar Items

Distance Measures

- **Goal: Find near-neighbors in high-dim. space**
 - We formally define “near neighbors” as points that are a “small distance” apart
- For each application, we first need to define what “**distance**” means
- **Today: Jaccard distance/similarity**
 - The **Jaccard similarity** of two **sets** is the size of their intersection divided by the size of their union:
$$\text{sim}(\mathbf{C}_1, \mathbf{C}_2) = |\mathbf{C}_1 \cap \mathbf{C}_2| / |\mathbf{C}_1 \cup \mathbf{C}_2|$$
 - **Jaccard distance:** $d(\mathbf{C}_1, \mathbf{C}_2) = 1 - |\mathbf{C}_1 \cap \mathbf{C}_2| / |\mathbf{C}_1 \cup \mathbf{C}_2|$



3 in intersection

8 in union

Jaccard similarity = 3/8

Jaccard distance = 5/8

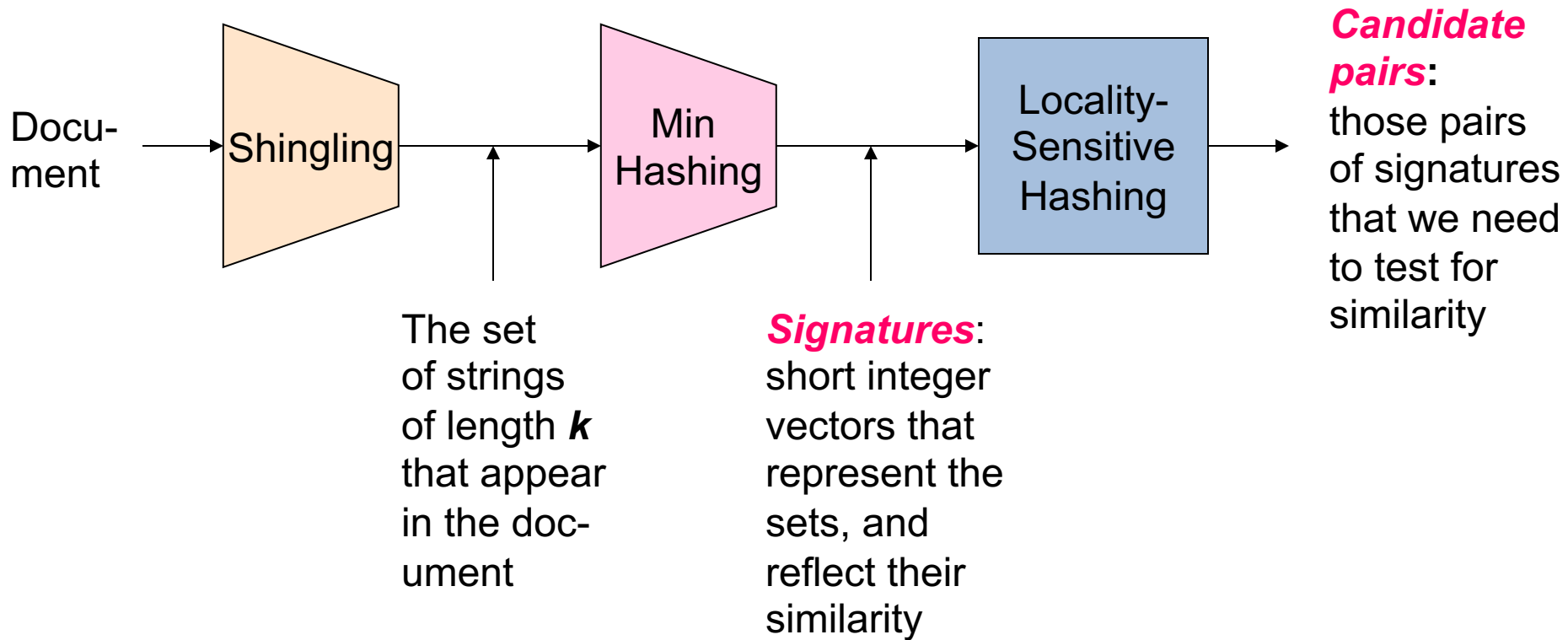
Task: Finding Similar Documents

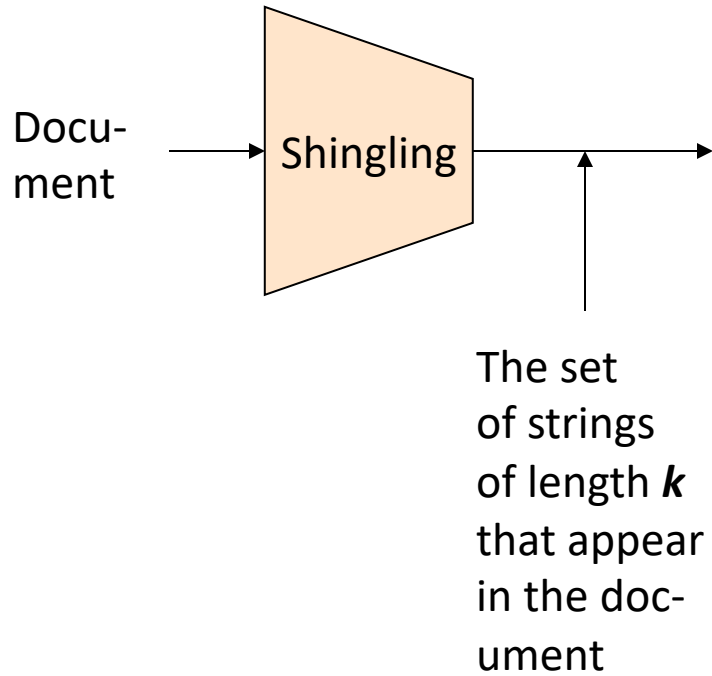
- **Goal:** Given a large number (N in the millions or billions) of documents, find “near duplicate” pairs
- **Applications:**
 - Mirror websites, or approximate mirrors
 - Don’t want to show both in search results
 - Similar news articles at many news sites
 - Cluster articles by “same story”
- **Problems:**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

3 Essential Steps for Similar Docs

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

The Big Picture





Shingling

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

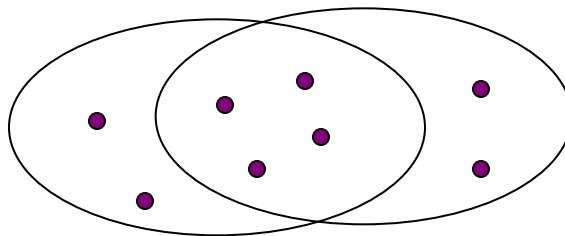
Define: Shingles

- 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 $D_1 = \text{ab cab}$
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

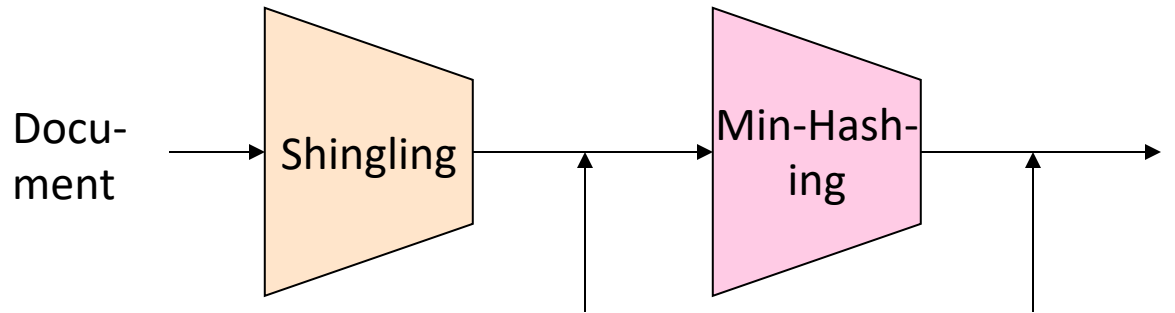
- **Document D_1 is a set of its k -shingles $C_1=S(D_1)$**
- Equivalently, each document is a 0/1 vector in the space of k -shingles
 - Each unique shingle is a dimension
 - Vectors are very sparse
- **A natural similarity measure is the Jaccard similarity:**

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



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



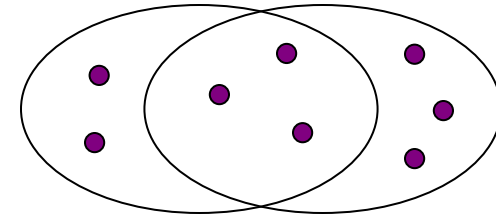
The set
of strings
of length k
that appear
in the doc-
ument

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

MinHashing

Step 2: **Minhashing:** Convert large sets to short signatures, while preserving similarity

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!**
- **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			
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

Outline: Finding Similar Columns

- **So far:**
 - Documents \rightarrow Sets of shingles
 - Represent sets as boolean vectors in a matrix
- **Next goal: Find similar columns while computing small signatures**
 - **Similarity of columns == similarity of signatures**

Outline: Finding Similar Columns

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

Hashing Columns (Signatures)

- **Key idea:** “hash” each column C to a small **signature** $h(C)$, such that:
 - (1) $h(C)$ is small enough that the signature fits in RAM
 - (2) $\text{sim}(C_1, C_2)$ 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 $\text{sim}(C_1, C_2)$ 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. $h(C_1) \neq h(C_2)$
- **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**

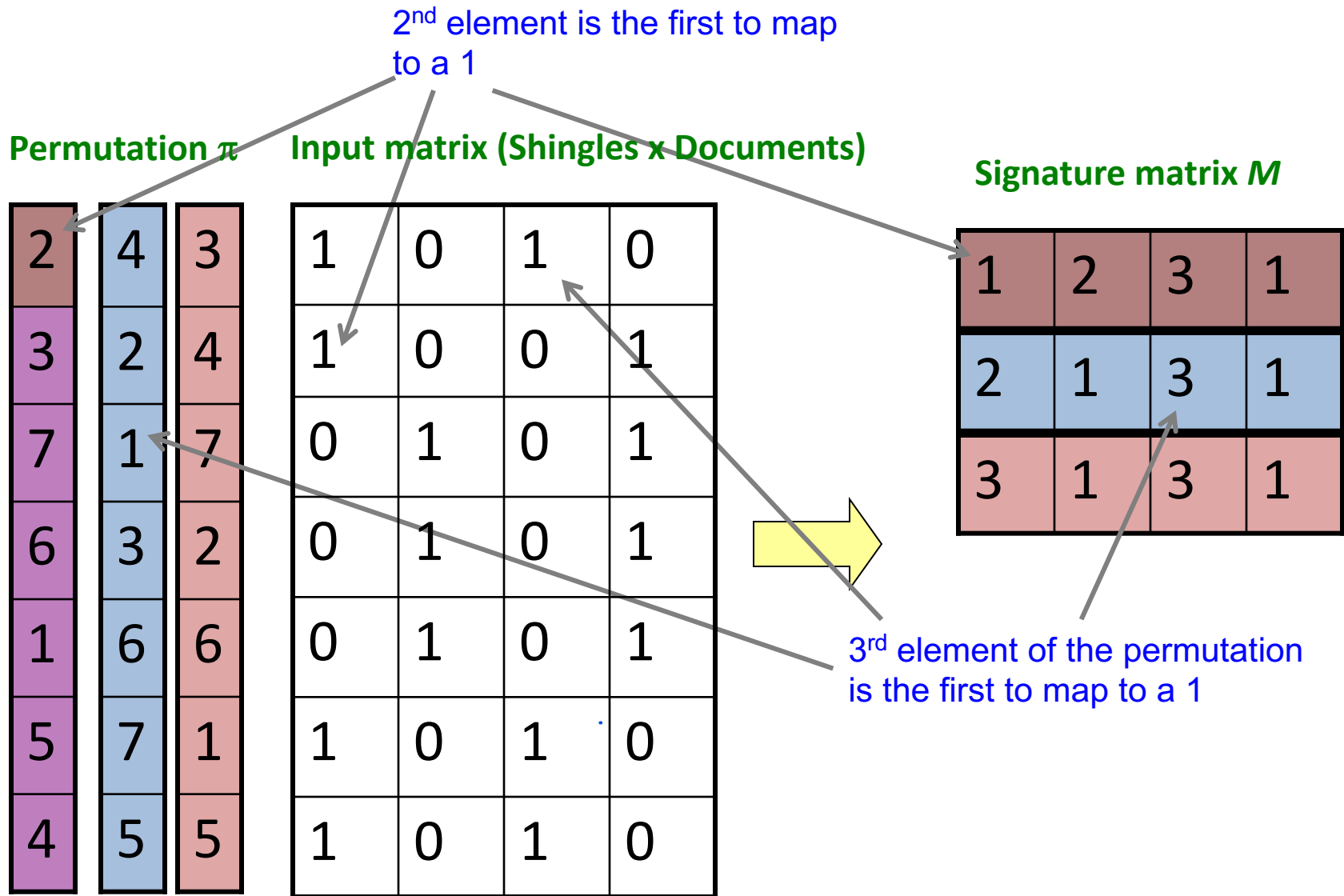
Min-Hashing

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

$$h_{\pi}(\mathbf{C}) = \min_{\pi} \pi(\mathbf{C})$$

- Use several (e.g., 100) independent hash functions (that is, permutations) to create a signature of a column

Min-Hashing Example



The Min-Hash Property

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

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

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

Four Types of Rows

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

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

– a = # rows of type A, etc.

- Note:** $\text{sim}(C_1, C_2) = a/(a + b + c)$
- 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

- We know: $\Pr[h_{\pi}(C_1) = h_{\pi}(C_2)] = \text{sim}(C_1, C_2)$
- Now generalize to multiple hash functions
- The *similarity of two signatures* is the fraction of the hash functions in which they agree
- **Note:** Because of the Min-Hash property, the similarity of columns is the same as the expected similarity of their signatures

Min-Hashing Example

Permutation π

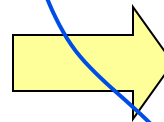
2	4	3
3	2	4
7	1	7
6	3	2
1	6	6
5	7	1
4	5	5

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

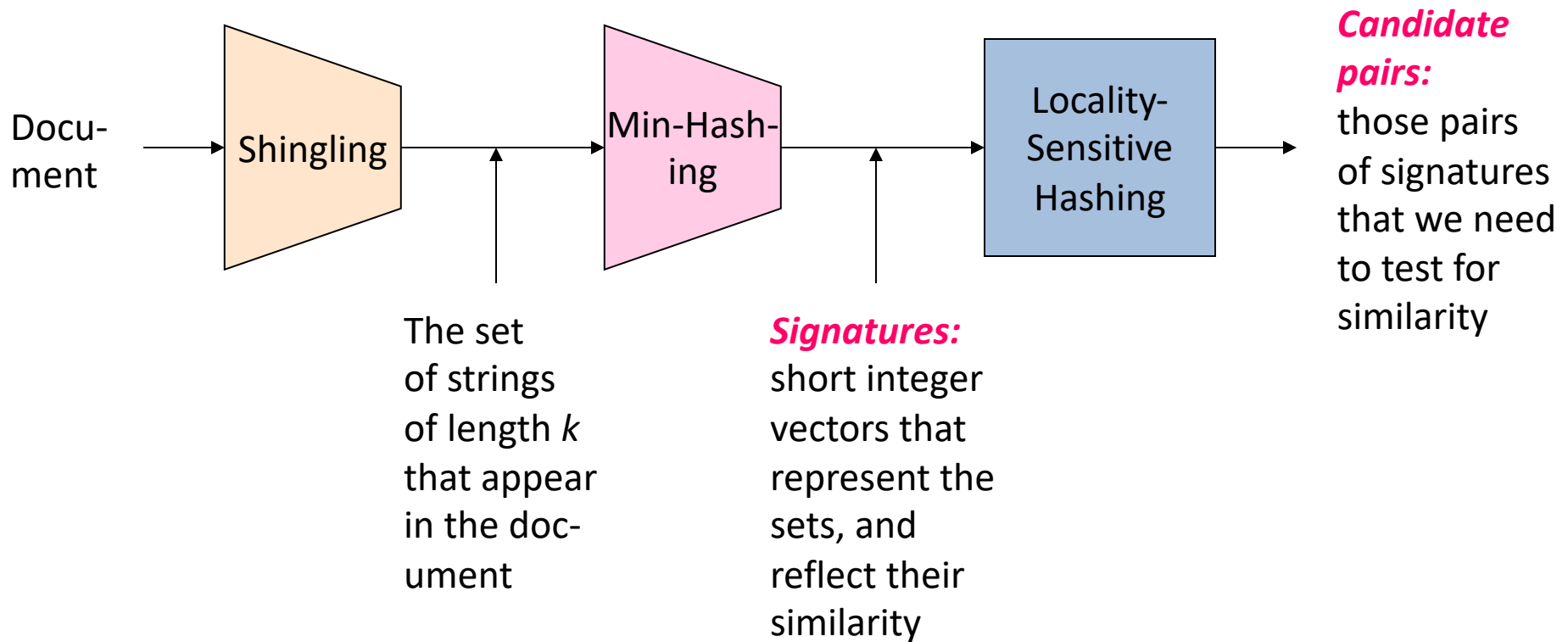
Signature matrix M

1	2	3	1
2	1	3	1
3	1	3	1



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



Locality Sensitive Hashing

Step 3: *Locality-Sensitive Hashing:*

Focus on pairs of signatures likely to be from similar documents