**Lecture 2 – Finding Similar Items**

Find near-neighbors in high-dimensional space is a case where a problem is expressed in finding similar sets. Examples are;

* Pages with similar words :- Duplicate detection(Mirror Pages, Plagiarism)
* Customers who purchased similar products :- Online Purchases(Amazon)

Problems:

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

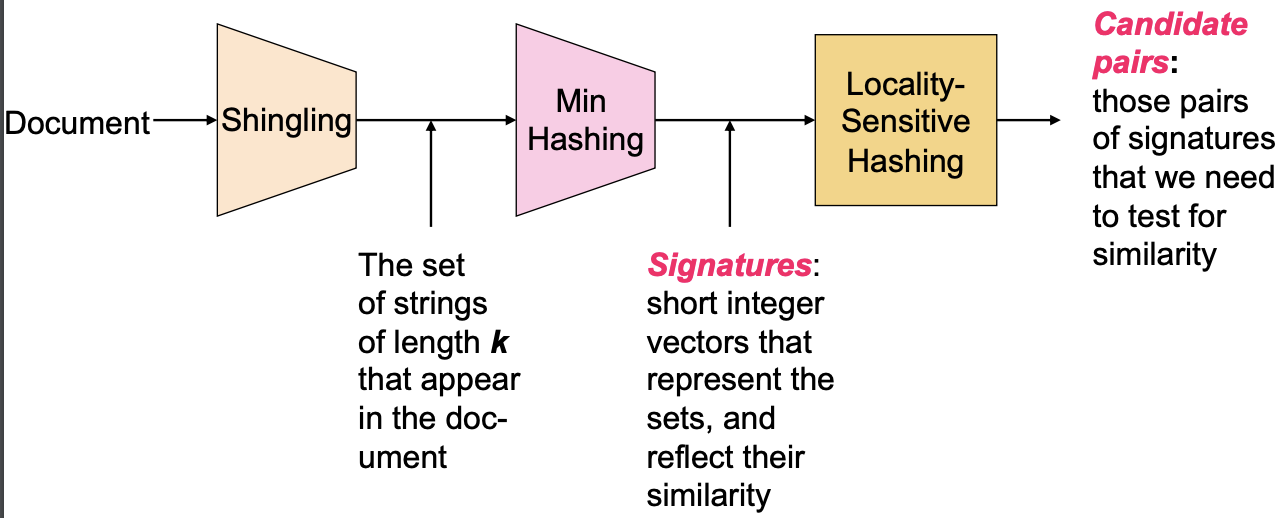
**Jaccard Similarity –** The Jaccard similarity of two sets is the size of their intersection divided by the size of their union.

****

**Jaccard Distance –** 1 minus the Jaccard Distance



**3 Essential Steps for Similar Docs**



**Shingles –** convert documents to sets

* Documents as High-Dim Data
* Definition of Shingles

Example: **k=2**; document **D1** = abcab

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

**Option:** Shingles as a bag (multiset), count ab twice: **S’(D1)** = {ab, bc, ca, ab}

* Compressing shingles

To compress long shingle, we can hash (or simply map) them to 4 bytes

* Similarity Metric for shingles
* Document D1 is a set of its k-shingles C1=S(D1)
* Equivalently, each document is a 0/1 vector in the space of k-shingles
* Each unique shingle is a dimension
* Working Assumption

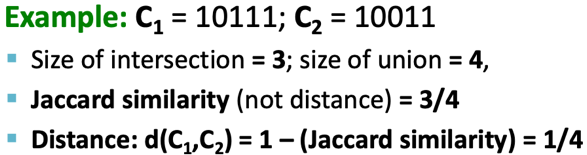
Documents that have lots of shingles in common have similar text, even if the text appears in different order.

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

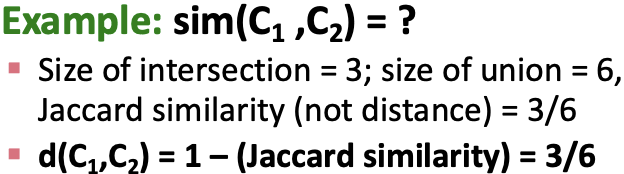
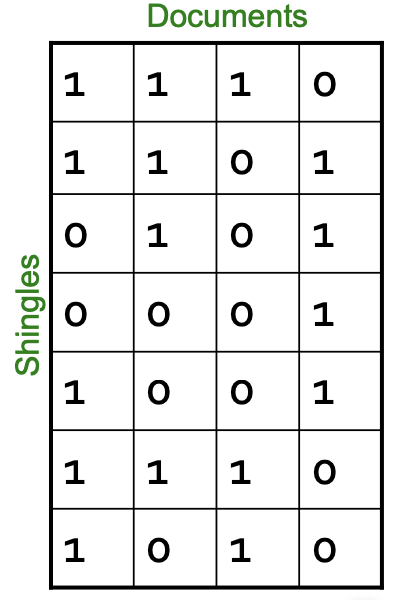
**Minhashing -**  convert large sets to short signatures, while preserving similarity

* Encoding Sets as Bit Vectors
* Interpret set intersection as bitwise **AND**, and set union as bitwise **OR**

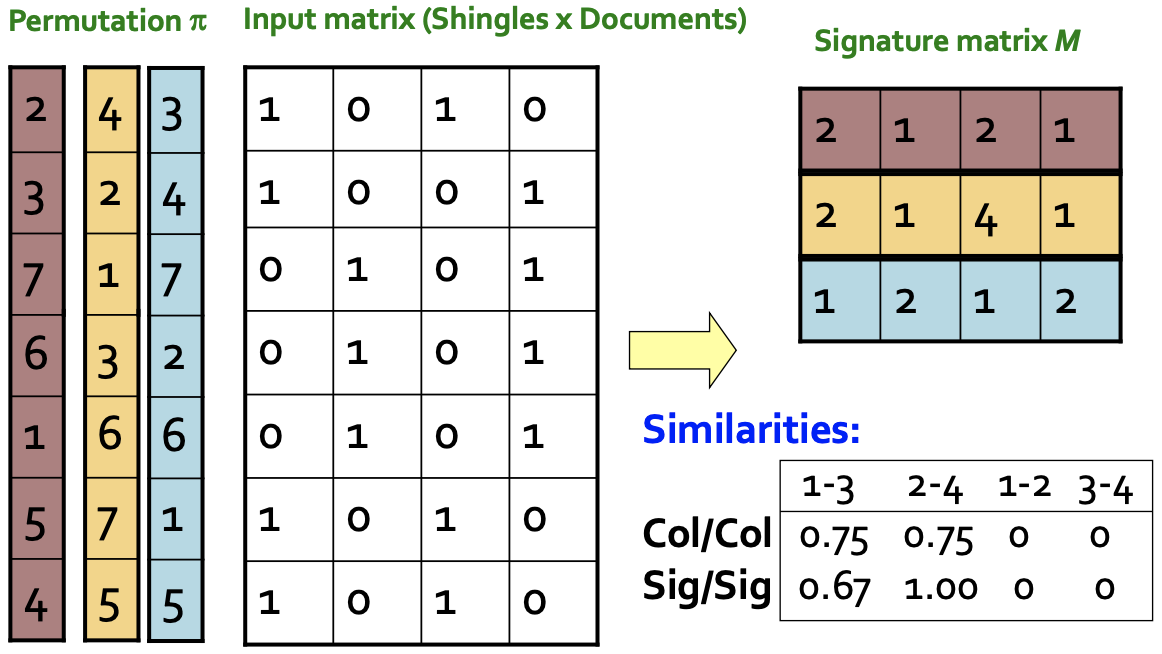


* From Sets to Boolean Matrices
* Rows = elements(shingles). Columns = sets(documents)

C1 C2 C3 C4



* Outline: Finding Similar Columns
* Documents -> sets of shingles
* **Similarity of columns == similarity of signatures**
* Hashing Columns(Signatures)
* **h(C)** is small enough that the signature fits in RAM
* **sim(C1, C2)** is the same as the “similarity” of signatures **h(C1)** and **h(C2)**
* Goal: Find a hash function h(·) such that:
* If sim(C1,C2) is high, then with high prob. h(C1) = h(C2)
* If sim(C1,C2) is low, then with high prob. h(C1) ≠ h(C2)
* Min-Hashing Example
* Check if the index of lowest number(0) in the permutation column is equals to 0 in the input matrix (Shingles & Documents matrix)
* if it is check if the index of the next lowest number(1) in the permutation column is equals to 0. Else if it is equals to 1 in the input matrix (Shingles & Documents matrix) add that number from the permutation to the signature matrix
* if it is
* check if the index of the next lowest number(2) in the permutation column is equal to 0
* if it is check if the index of the next lowest number(1) in the permutation column is equals to 0. Else if it is equals to 1 in the input matrix (Shingles & Documents matrix) add that number from the permutation to the signature matrix
* repeat until you get a complete signature matrix.



* Similarity for Signatures

We know: **Pr[h(C1) = h(C2)] = sim(C1, C2)**

The similarity of two signatures is the fraction of the hash functions in which they agree

**Locality-Sensitive Hashing –** Focus on pairs of signatures likely to be from similar document

* (candidate pairs)

**LSH**

False positives

False negatives

**b – bands , r – rows, s – similarity**

Columns C1 and C2 have similarity **s**

Pick any band (r rows)

Prob. that all rows in band equal = **sr**

Prob. that some row in band unequal = **1 - sr**

Prob. that no band identical = **(1 - sr )b**

Prob. that at least 1 band identical = **1 - (1 - sr )b**