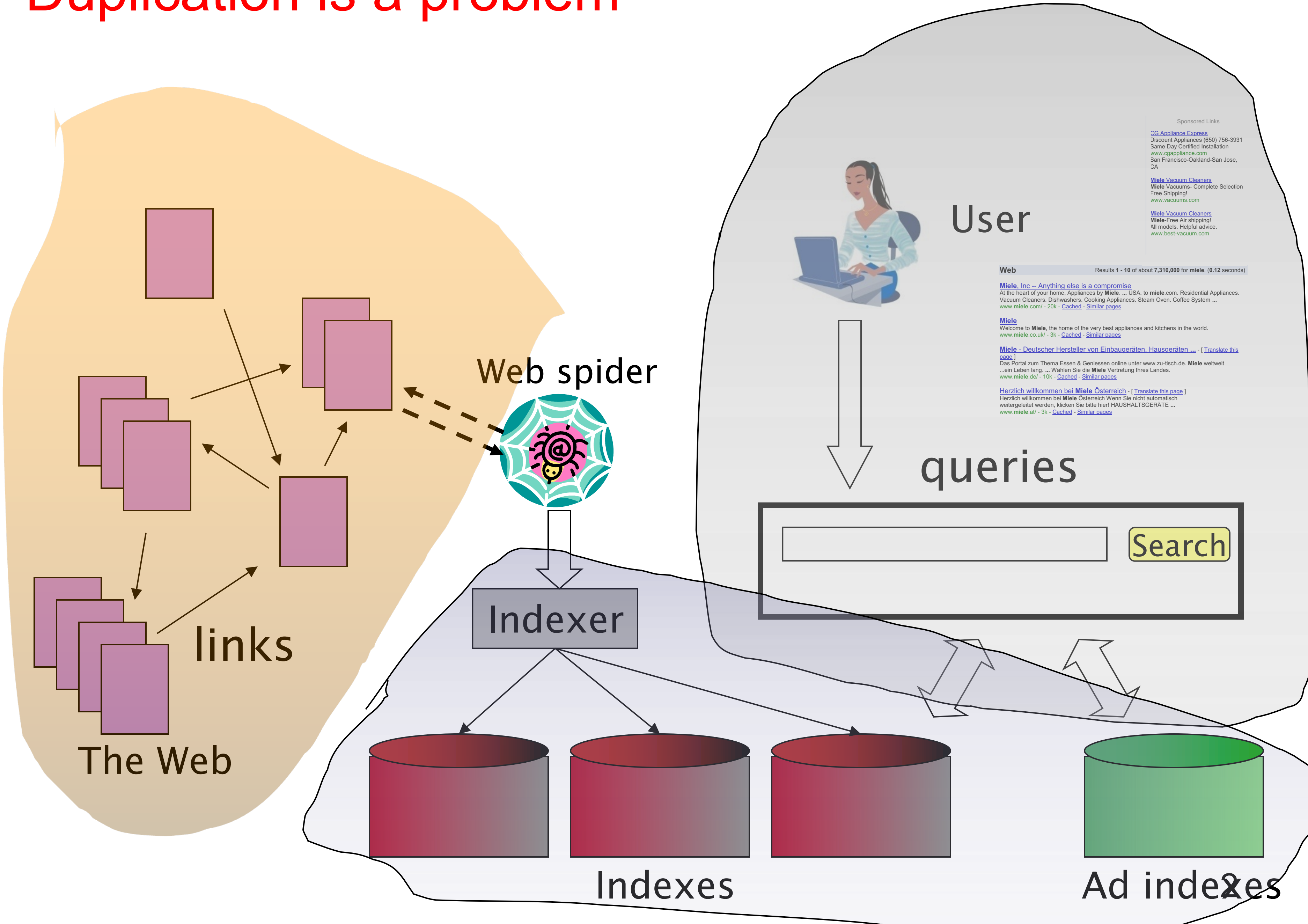
Near Duplicate Detection

Duplication is a problem



Duplicate documents

- The web is full of duplicated content
  - About 30% are duplicates
- Duplicates need to be removed for
  - Crawling
  - Indexing
  - Statistical studies
- Strict duplicate detection = exact match
  - Not as common
- But many, many cases of near duplicates
  - E.g., Last modified date the only difference between two copies of a page
  - Other minor difference such as web master, logo, …

Other applications

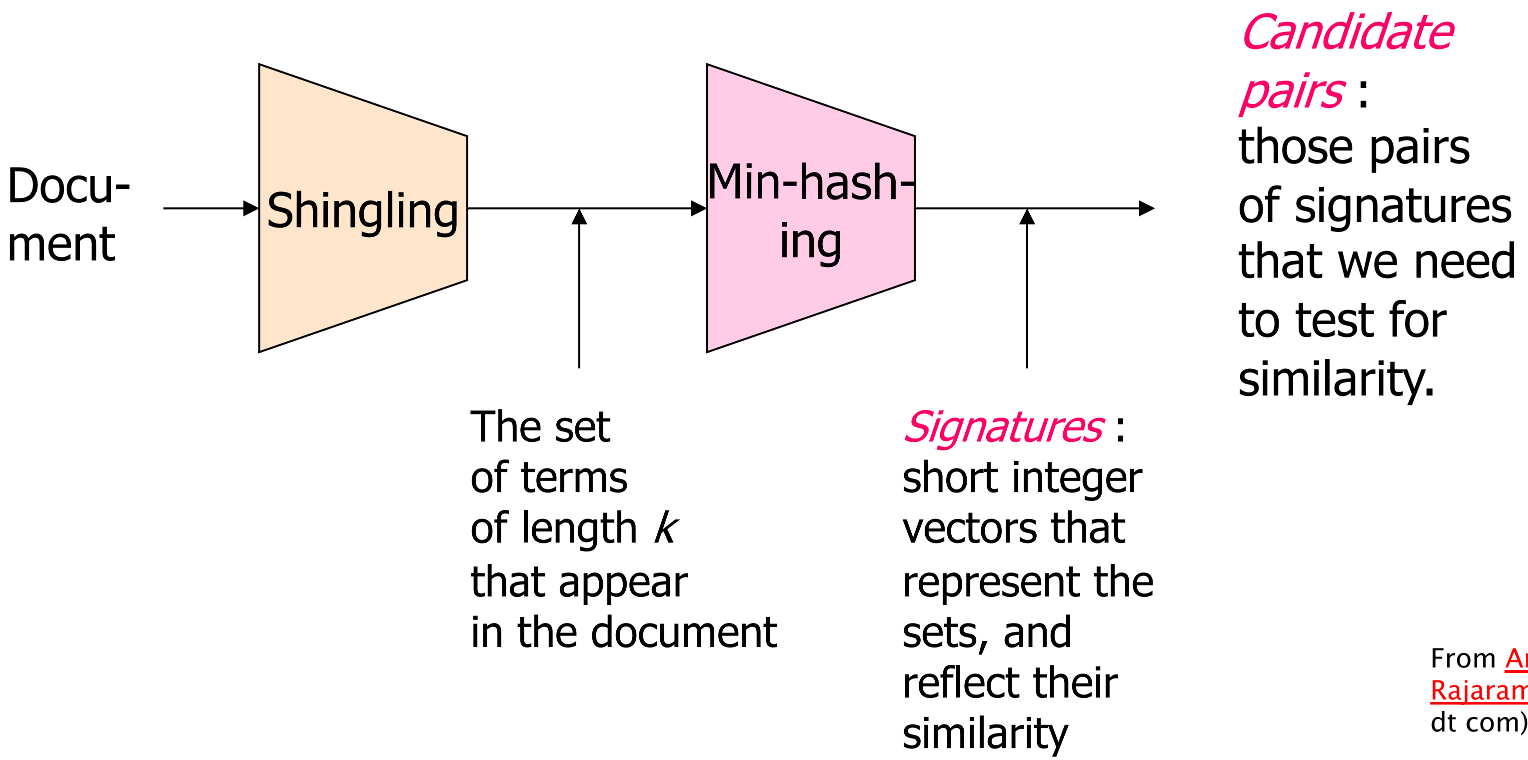
- Many Web-mining problems can be expressed as finding "similar" sets:
  1. Topic classification--Pages with similar words, Mirror web sites, Similar news articles
  2. Recommendation systems--NetFlix users with similar tastes in movies.
  3. movies with similar sets of fans.
  4. Images of related things.
  5. Community in online social networks
  6. Plagiarism

Algorithms for finding similarities

- Edit distance
  - Distance between A and B is defined as the minimal number of operations to edit A into B
  - Mathematically elegant
  - Many applications (like auto-correction of spelling)
  - Not efficient
- Shingling

Techniques for Similar Documents

- Shingling : convert documents, emails, etc., to sets.
- Minhashing : convert large sets to short signatures, while preserving similarity.



From [Anand Rajaraman] (anand @ kosmix dt com), [Jeffrey D. Ullman]

Shingles

- A k -shingle (or k -gram) for a document is a sequence of k terms that appears in the document.
- Example:
  - ***a rose is a rose is a rose*** →

    a rose is a

    rose is a rose

    is a rose is

    a rose is a

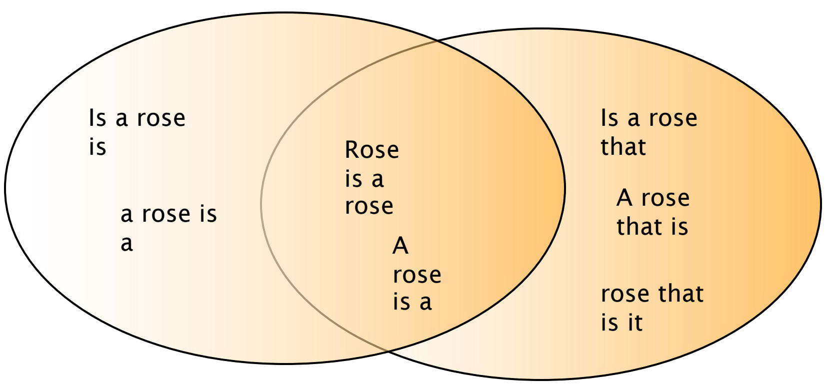
    rose is a rose

  The ***set*** of shingles is {*a rose is a, rose is a rose, is a rose is, a rose is a*}
- Note that “a rose is a rose” is repeated twice, but only appear once in the set
  - Option: regard shingles as a bag, and count “a rose is a” twice.
- Represent a doc by its set of k-shingles.
- Documents that have lots of shingles in common have similar text, even if the text appears in different order.
- Careful: you must pick k large enough.
  - If k=1, most documents overlap a lot.

Jaccard similarity

$$\mathrm{Jaccard}(C_i, C_j) = \frac{|C_i \cap C_j|}{|C_i \cup C_j|}$$

- ***a rose is a rose is a rose***
  - ➔ *{a rose is a, rose is a rose, is a rose is, a rose is a}*
- ***A rose is a rose that is it***
  - ➔ *{a rose is a, rose is a rose, is a rose that, a rose that is, rose that is it}*



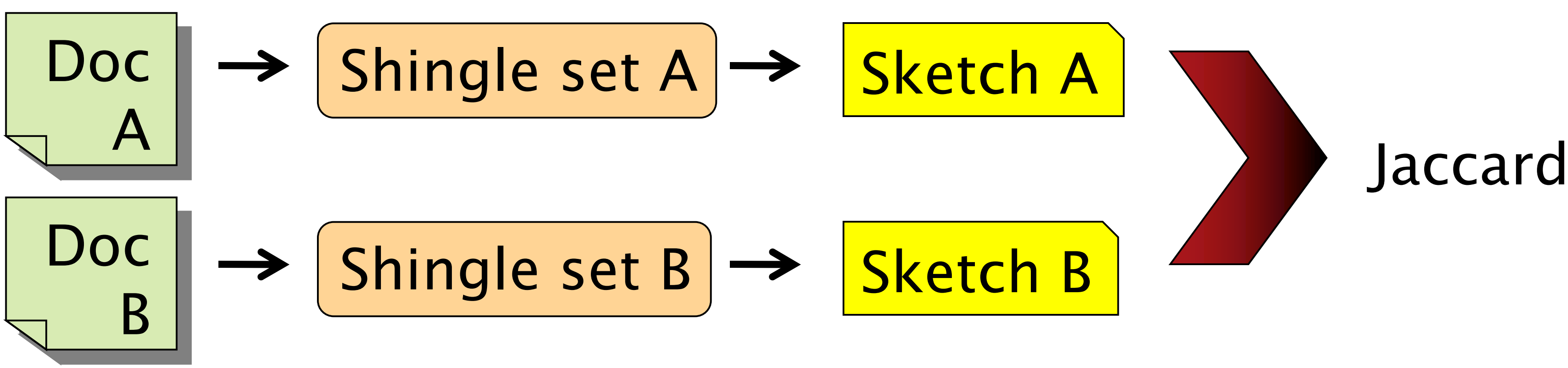
2 in intersection.
7 in union.
Jaccard similarity
= 2/7

The size is the problem

- The shingle set can be very large
- There are many documents (many shingle sets) to compare
  - Billions of documents and shingles
- Problems:
  - Memory: When the shingle sets are so large or so many that they cannot fit in main memory.
  - Time: Or, when there are so many sets that comparing all pairs of sets takes too much time.
  - Or both.

Shingles + Set Intersection

- Computing exact set intersection of shingles between all pairs of documents is expensive/intractable
  - Approximate using a cleverly chosen subset of shingles from each (a *sketch*)
- Estimate (size_of_intersection / size_of_union) based on a short sketch



Set Similarity of sets C_i , C_j

$$\text{Jaccard}(C_i, C_j) = \frac{|C_i \cap C_j|}{|C_i \cup C_j|}$$

- View sets as columns of a matrix A; one row for each element in the universe. $a_{ij} = 1$ indicates presence of shingle i in set (document) j
- Example

C_1	C_2
0	1
1	0
1	1
0	0
1	1
0	1

Jaccard**(C1,C2) = 2/5 = 0.4**

Key Observation

- For columns C_1, C_2, four types of rows

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

- Overload notation: A = # of rows of type A
- **Claim**

$$\text{Jaccard}(C_i, C_j) = \frac{A}{A + B + C}$$

Estimating Jaccard similarity

C_1	C_2
0	1
1	0
1	1
0	0
1	1
0	1
0	0
0	0

- Randomly permute rows
- Hash $h(C_i)$ = index of first row with 1 in column C_i
- Property

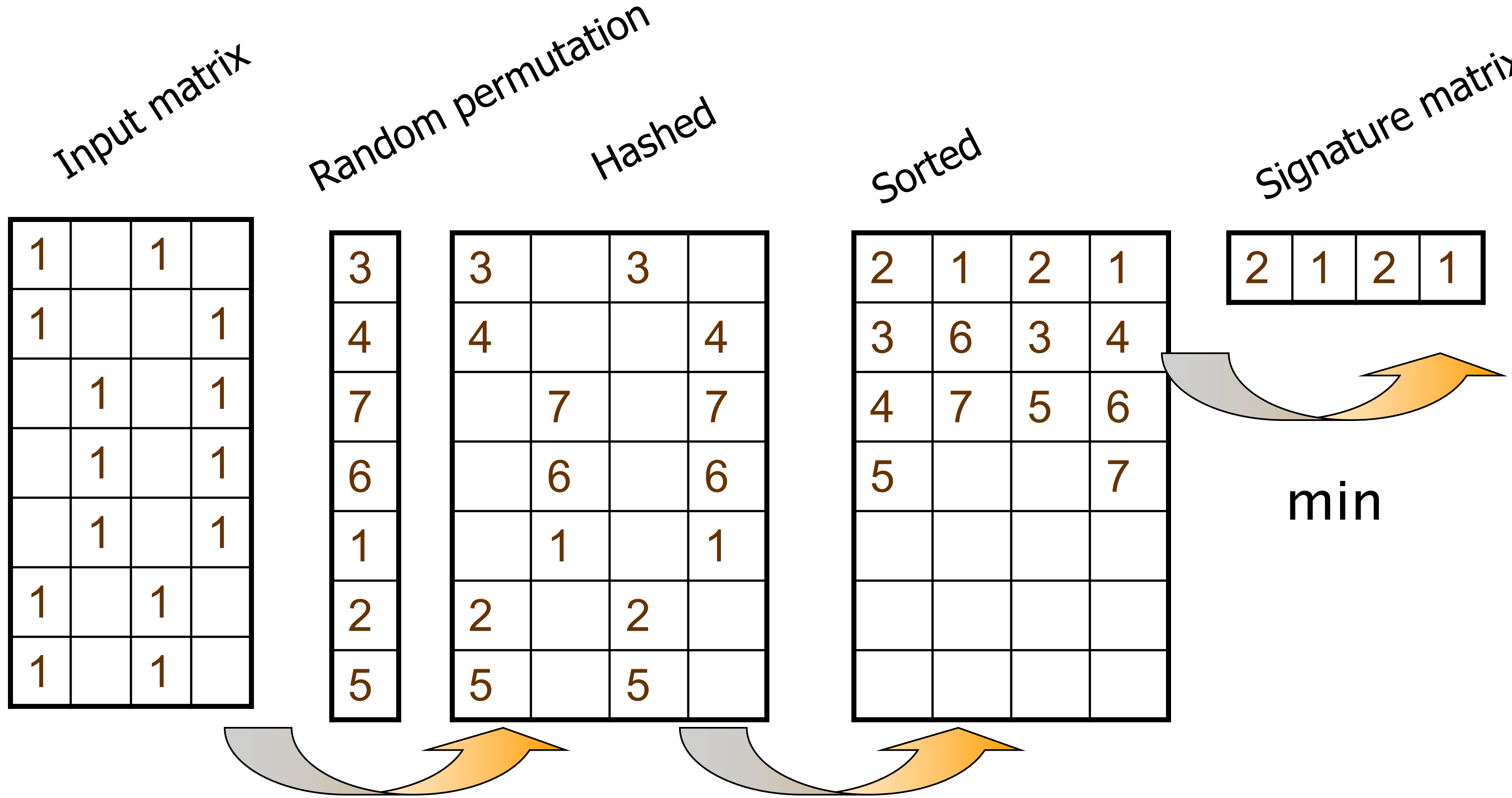
$$P\left[\ h(C_i) = h(C_j)\ \right] = \text{Jaccard}\left(C_i, C_j\right)$$

- Why?
  - Both are A/(A+B+C)
  - Look down columns C_1, C_2 until first non-Type-D row
  - $h(C_i) = h(C_j) \leftarrow\rightarrow$ type A row

Representing documents and shingles

- To compress long shingles, we can hash them to (say) 4 bytes.
- Represent a doc by the set of hash values of its k-shingles.
- Represent the documents as a matrix
  - 4 documents
  - 7 shingles in total
  - Column is a document
  - Each row is a shingle
- In real application the matrix is sparse—there are many empty cells

	doc1	doc2	doc3	Doc4
Shingle 1	1		1	
Shingle 2	1			1
Shingle 3		1		1
Shingle 4		1		1
Shingle 5		1		1
Shingle 6	1		1	
Shingle 7	1		1	



4 docs

Hash

sort

Similarities:

- 1~3: 1
- 2~4: 1
- 1~4: 0

Repeat the previous process

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

Input matrix

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

More Hashings produce better result

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

Input matrix

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:

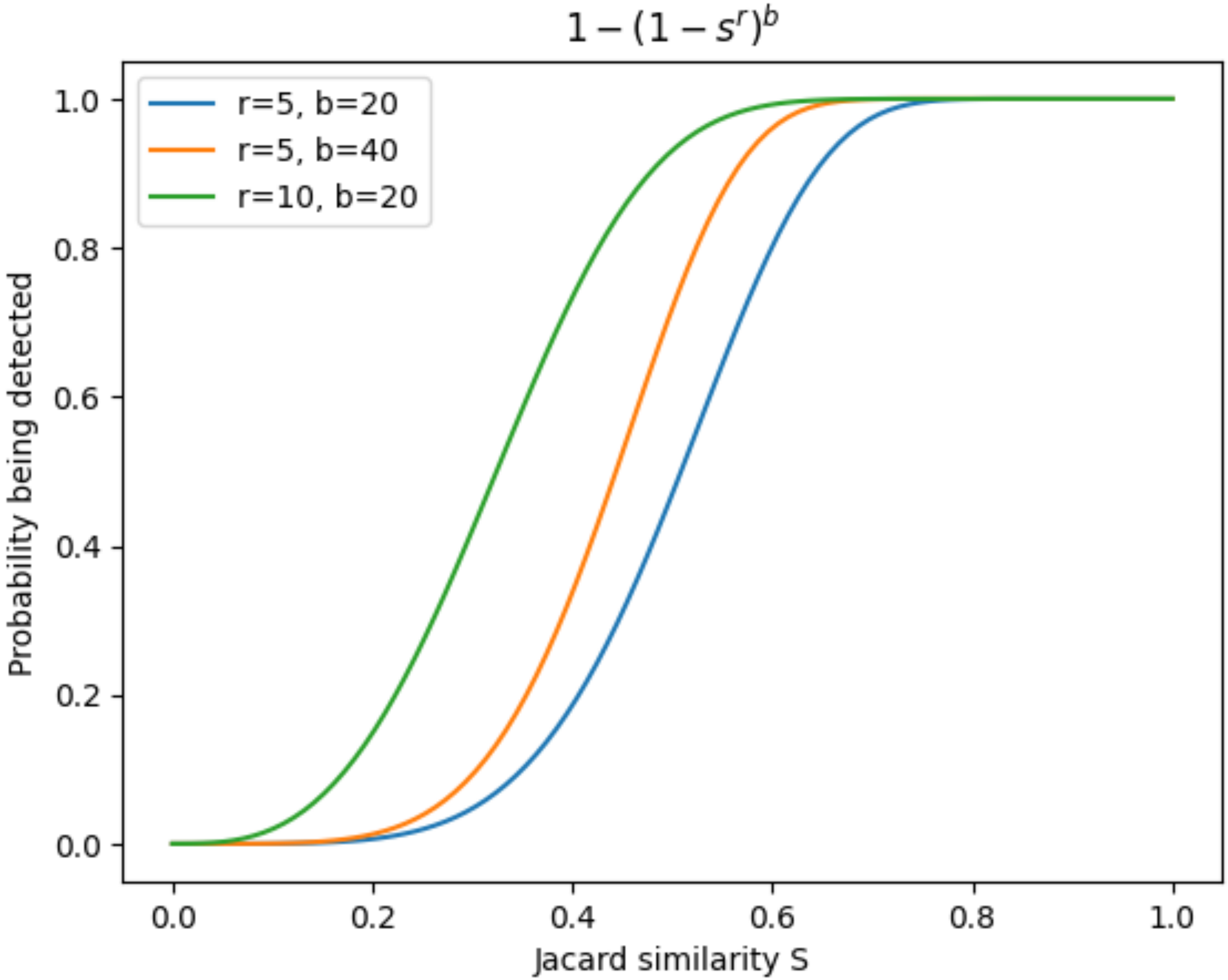
	1-3	2-4	1-2	3-4
Col/Col	0.75	0.75	0	0
Sig/Sig	0.67	1.00	0	0

Sketch of a document

- Create a "sketch vector" (of size ~200) for each document
  - Documents that share ≥ t (say 80%) corresponding vector elements are near duplicates
  - For doc D, $\text{sketch}_D[\,i\,]$ is as follows:
    - Let f map all shingles in the universe to $0..2^m$ (e.g., f = fingerprinting)
    - Let π_i be a *random permutation* on $0..2^m$
    - Pick MIN $\{\pi_i(f(s))\}$ over all shingles s in D

How to detect similar pairs

- Exhaustive comparison is prohibitive
- Hashing the signature into buckets
- If two documents are found in the same bucket, then they are probability similar


$1-(1-s^r)^b$
r=5, b=20
r=5, b=40
r=10, b=20
Probability being detected
Jacard similarity S
0.0
0.2
0.4
0.6
0.8
1.0