Finding Near-Duplicates

Mining Massive Datasets

Prof. Carlos Castillo — https://chato.cl/teach

Universitat Pompeu Fabra Barcelona

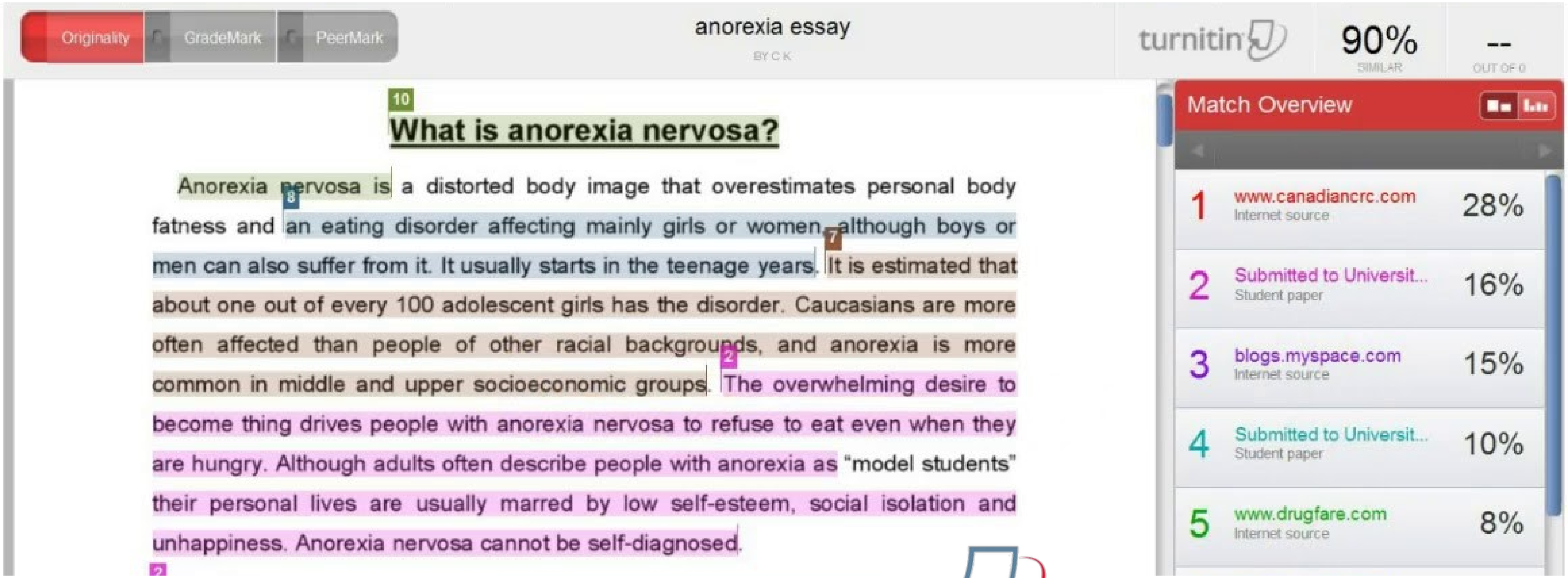
Source for this deck

- Mining of Massive Datasets 2nd edition (2014) by Leskovec et al. (Chapter 3) [slides ch3]

Fast near-neighbor applications

- For documents
  - Find “legitimate” duplicates
    - Copies of the same press release or cable
    - Mirrors of the same documents, for efficiency
  - Find “illegitimate” duplicates
    - Plagiarism
- For baskets
  - Find customers who purchase similar items

Example: plagiarism detection



anorexia essay

BY C K

10

What is anorexia nervosa?

Anorexia nervosa is a distorted body image that overestimates personal body fatness and an eating disorder affecting mainly girls or women, although boys or men can also suffer from it. It usually starts in the teenage years. It is estimated that about one out of every 100 adolescent girls has the disorder. Caucasians are more often affected than people of other racial backgrounds, and anorexia is more common in middle and upper socioeconomic groups. The overwhelming desire to become thing drives people with anorexia nervosa to refuse to eat even when they are hungry. Although adults often describe people with anorexia as "model students" their personal lives are usually marred by low self-esteem, social isolation and unhappiness. Anorexia nervosa cannot be self-diagnosed.

1	www.canadiancrc.com Internet source	28%
2	Submitted to Universit... Student paper	16%
3	blogs.myspace.com Internet source	15%
4	Submitted to Universit... Student paper	10%
5	www.drugfare.com Internet source	8%

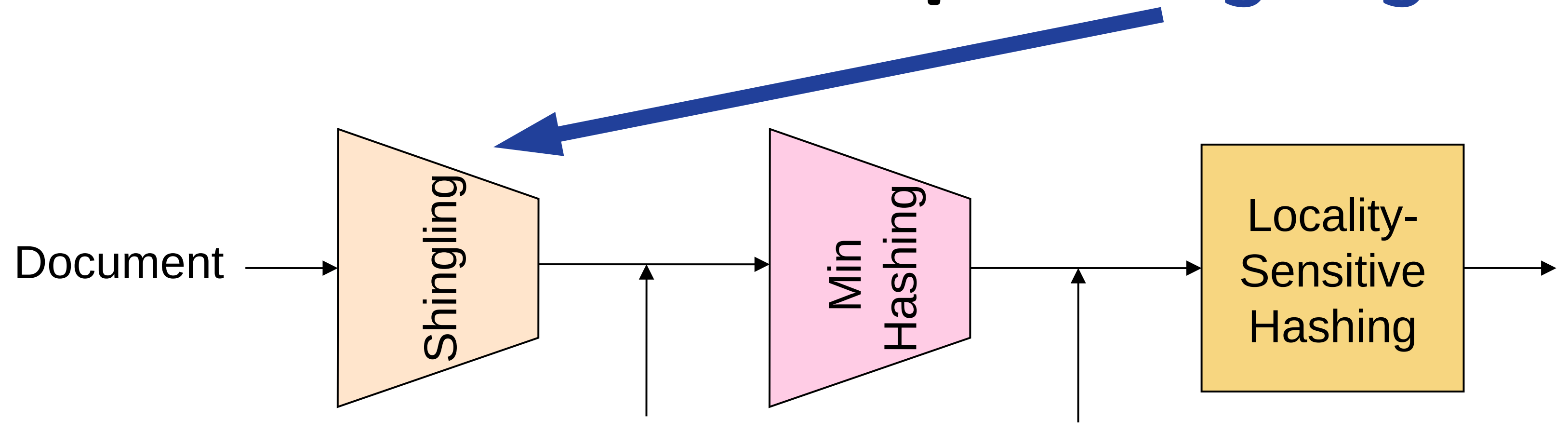
turnitin

Fast near-neighbor challenges

- Too many documents to compare all pairs
  - OK to pay linear or log cost, but not quadratic
- Documents cannot fit in main memory
  - They are too large or too many
- Many small pieces of one document can appear out of order in another

Shingling
(ngrams)

First step: shingling



Sets of
***k** letters or words*
that appear
consecutively
in the document

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

Candidate pairs:
those pairs
of signatures
that we need
to test for
similarity

Naïve solution: feature selection over bag of words

- Document = set of terms

  $\rightarrow$ Document = set of important terms

- Now, compute all pairs similarity

- Doesn't work for at least two reasons, why?

Naïve solution: feature selection over bag of words

- Document = set of terms

  $\rightarrow$ Document = set of important terms

- Now, compute all pairs similarity
- Doesn't work for at least two reasons, why?
  - Doesn't preserve the ordering
  - Unimportant terms are also relevant (stylistic)

Shingles

- An ***ngram*** in a document is a sequence of n tokens that appears in the doc
- **Shingles** are either ngrams (word-level) or sequences of characters, depending on the application
- **Character-level example:** **k=2**; document $\mathbf{D_1}$ = abcab

  Set of 2-shingles: $\mathbf{S(D_1)}$ = {ab, bc, ca}

  – **Option:** Shingles as a bag (multiset), count ab twice: $\mathbf{S'(D_1)}$ = {ab, bc, ca, ab}

Example: 4-grams (shingle = 4 consecutive words)

E.g., 4-shingles of
“`My name is Inigo Montoya. You killed my father. Prepare to die`”:

{

- my name is inigo
- name is inigo montoya
- is inigo montoya you
- inigo montoya you killed
- montoya you killed my
- you killed my father
- killed my father prepare
- my father prepare to
- father prepare to die

}

Compressed representation of shingles

- To **compress long shingles**, we can **hash** them to (say) 4 bytes
- **Represent a document by the set of hash values of its *k*-shingles**
- **Idea:** Two documents could (rarely) appear to have shingles in common, when in fact only the hash-values were shared

- **Example:** **k=2**; document $\mathbf{D_1}$= abcab
  Set of 2-shingles: $\mathbf{S(D_1)}$ = {ab, bc, ca}
  Hash the singles: $\mathbf{h(D_1)}$ = {1, 5, 7}

Documents as sets of shingles

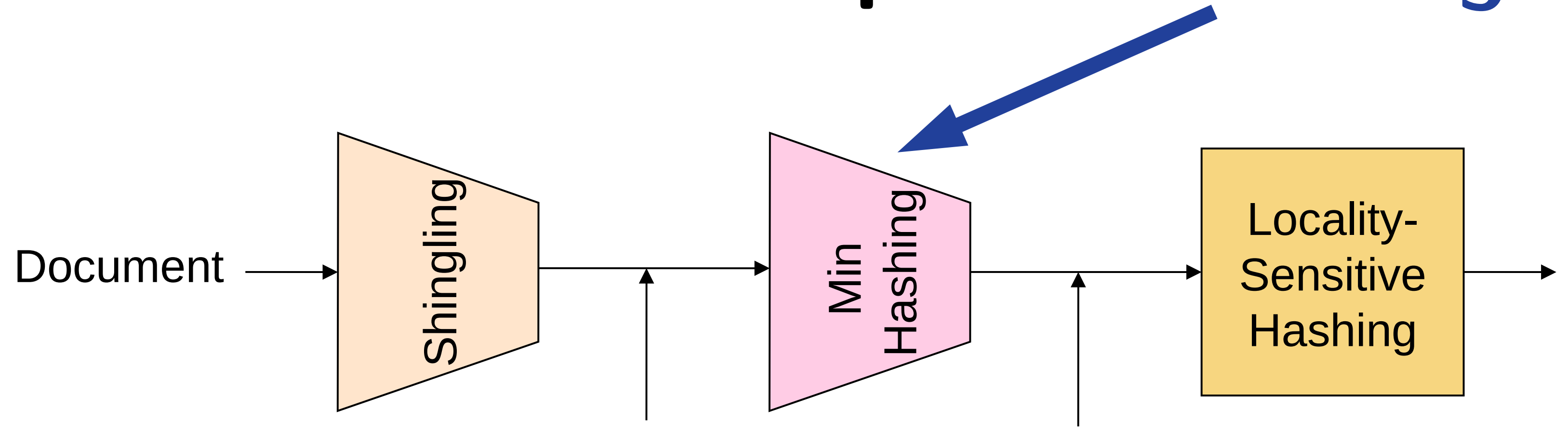
- A document is now a set of shingles
  - Dimensionality reduced from "words in a dictionary" to "number of distinct shingles"
  - Higher dimensionality but more sparse
- Working assumption
  - Documents that have lots of shingles in common have similar text, even if the text appears in different order
- 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

Using shingles directly

- **Suppose we need to find near-duplicate documents among million documents**
- Naïvely, we would have to compute **all pairwise Jaccard similarities** ≈ **$5*10^{11}$** comparisons
- At 10^5 secs/day and 10^6 comparisons/sec, it would take **5 days**
- For 10 million, it takes more than a year…

Min hashing

Next step: min hashing



Sets of
k *letters or words*
that appear
consecutively
in the document

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

Candidate pairs:
those pairs
of signatures
that we need
to test for
similarity

Sets can be bit vectors

- Many similarity problems involve **finding subsets with substantial intersection**

- **Remember we can encode sets using bit vectors**
  - set intersection = bitwise **AND**
  - set union = bitwise **OR**

$$J(C_1, C_2) = \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|}$$

- **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 – (Jaccard similarity) = 1/4**

From sets to boolean matrices

- **Rows = items** (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 very sparse!

$$\begin{array}{c|c|c|c|c|}
 & \multicolumn{4}{c}{\text{Documents}} \\ \cline{2-5}
 & 1 & 1 & 1 & 0 \\ \cline{2-5}
 & 1 & 1 & 0 & 1 \\ \cline{2-5}
 & 0 & 1 & 0 & 1 \\ \cline{2-5}
\text{Shingles} & 0 & 0 & 0 & 1 \\ \cline{2-5}
 & 1 & 0 & 0 & 1 \\ \cline{2-5}
 & 1 & 1 & 1 & 0 \\ \cline{2-5}
 & 1 & 0 & 1 & 0 \\ \cline{2-5}
\end{array}$$

Hashing set representations

- We don't want to compare c_1, c_2, they might be too large, slowing down the computation
- Instead, we compute signatures $h(c_1)$, $h(c_2)$ that are smaller in size than c_1 and c_2
- **Desired properties**:
  $c_1 = c_2 \Rightarrow \text{Prob.}(h(c_1) = h(c_2))$ is large
  $c_1 \neq c_2 \Rightarrow \text{Prob.}(h(c_1) \neq h(c_2))$ is large

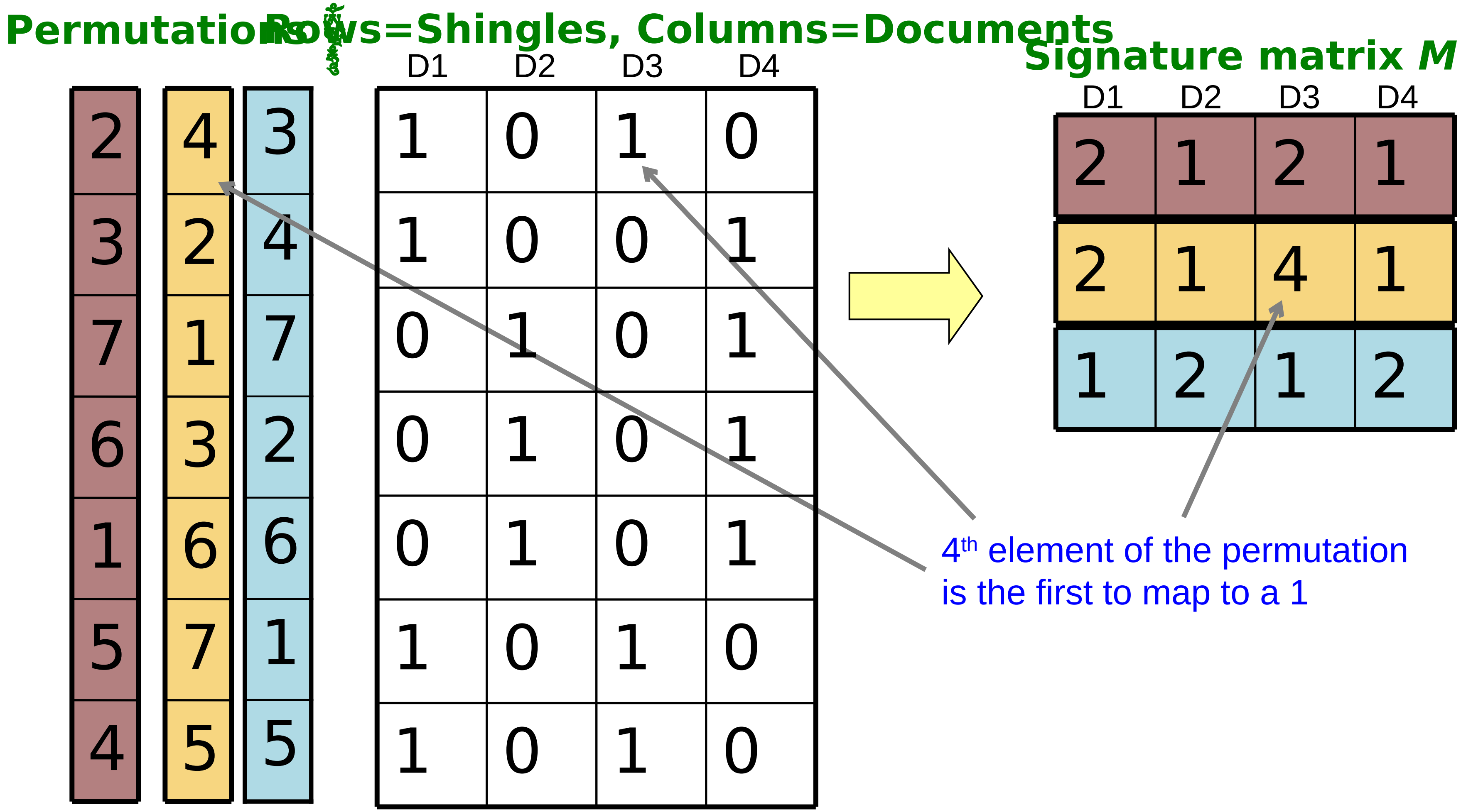
Hashing set representations (cont.)

- Naïve approach (non-LSH-based):
  - 1) Compute signatures of columns: small summaries of columns
  - 2) Examine all pairs of signatures to find similar columns
    - Essential: Similarities of signatures and columns are related
  - 3) Optional: verify 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)

Hash function for Jaccard metric: min hashing

- Imagine the rows of the boolean matrix permuted under **random but fixed 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

Minhash example


Permutations
Rows=Shingles, Columns=Documents
Signature matrix *M*
4th element of the permutation is the first to map to a 1

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

D1	D2	D3	D4
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

D1	D2	D3	D4
2	1	2	1
2	1	4	1
1	2	1	2

Exercise

Permutation

Rows=Shingles, Columns=Documents

	D1	D2	D3	D4
3	1	0	1	0
2	1	0	0	1
1	0	1	0	1
4	0	1	0	1
7	0	1	0	1
5	1	0	1	0
6	1	0	1	0

Signature matrix *M*

D1	D2	D3	D4

Index of the bit vector position where the first 1 occurs according to the ordering of the permutation

Answer in
Nearpod draw it
Code to be given in class

Minhash approximates Jaccard

- **Choose a random permutation π**
- **<u>Claim:</u>** $\Pr[h_\pi(C_1) = h_\pi(C_2)] = sim(C_1, C_2)$
- **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. $y \in C_1 \cap C_2$
  - $\Pr[\min(\pi(C_1))=\min(\pi(C_2))]=|C_1 \cap C_2|/|C_1 \cup C_2|= sim(C_1, C_2)$

A single hash function is too coarse for our purposes

- We will use many permutations (say, 100)
- **A signature is a collection of minhashes:** one for each permutation
- Jaccard(c_1, c_2) = E[minhashsim(c_1, c_2)]
  - **minhashsim(c1,c2) = #matches / #permutations**

Example: three permutations

Permutation π

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

Rows=Shingles, Columns=Documents

D1	D2	D3	D4
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*

D1	D2	D3	D4
2	1	2	1
2	1	4	1
1	2	1	2

Similarities:

	1-3	2-4	1-2
Complete	0.75	0.75	0
Signatures	0.67	1.00	0

Minhash signatures

- **Pick $\pi_1 \ldots \pi_{100}$ random permutations of the rows (K=100)**
- Think of ***sig*(C)** as a column vector
  - ***sig*(C)[i] =** according to the *i*-th permutation, the index of the first row that has a 1 in column C
  - ***sig*(C)[i] = min (π_i(C))**
- The signature or "sketch" of document C has fixed size!
  - We achieved our goal: we "compressed" long bit vectors into short signatures

Implementation

- **Permuting rows even once is prohibitive**
- Create [illegible]$_1$ … [illegible]$_{100}$ by using **K = 100** hash functions ***k**$_i$*
  - Ordering of {1,2,…,n} under ***k**$_i$* (computing h(1), h(2), …, h(n) and sorting in increasing order) gives a random permutation!
- **One-pass implementation**
  - For each column ***C*** and hash function ***k**$_i$* keep a variable for the min-hash value
  - Initialize all ***sig(C)[i]* = ☜**
  - **Keep the min hash value in a row containing a 1:**
    - Suppose row ***j*** has 1 in column ***C***
      - Then for each ***k**$_i$* If ***k**$_i$**(j)* < *sig(C)[i]***, then ***sig(C)[i]* 🌍 *k**$_i$**(j)***

Summary

Things to remember

- **Shingling**: Convert documents to sets
  - 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 $\mathbf{Pr}[h_{\pi}(C_1) = h_{\pi}(C_2)] = sim(C_1, C_2)$
  - We used hashing to get around generating random permutations

Exercises for TT08-TT09

- Mining of Massive Datasets 2nd edition (2014) by Leskovec et al.
  - Exercises 3.1.4 (Jaccard similarity)
  - Exercises 3.2.5 (Shingling)
  - Exercises 3.3.6 (Min hashing)
  - Exercises 3.4.4 (Locality-sensitive hashing)