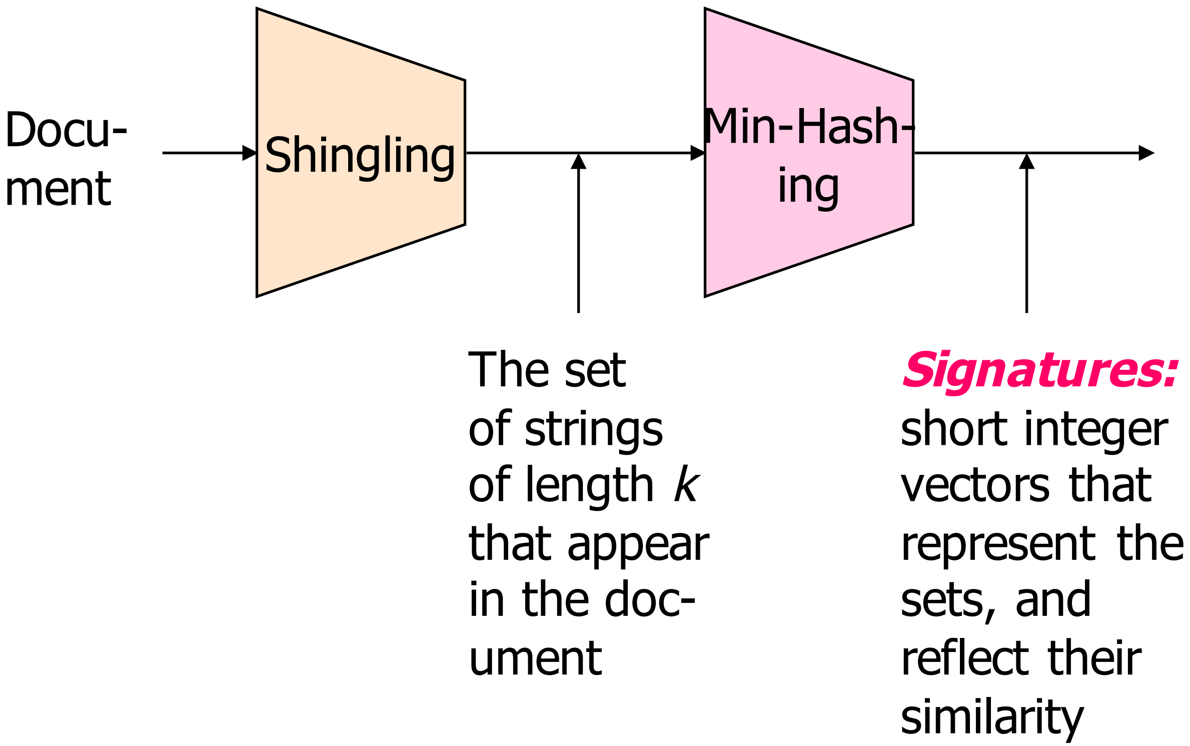
Finding Similar Sets (part 2)

Applications

Shingling

Minhashing

Locality-Sensitive Hashing



MinHashing

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

MinHashing

Data as Sparse Matrices

Jaccard Similarity Measure

Constructing Signatures

From Sets to Boolean Matrices

- Rows = **elements** of the universal set
- Columns = **sets**
- 1 in **row *e*** and **column *S*** if and only if element *e* is a member of set *S*
- Column similarity is the **Jaccard similarity** of the sets of their rows with 1: **intersction/union of sets**
- Typical matrix is **sparse** (many 0 values)
  - May not really represent the data by a boolean matrix
  - Sparse matrices are usually better represented by the list of non-zero values (e.g., triples)
  - But the matrix picture is conceptually useful.

Example 3.6

$Element$	S_1	S_2	S_3	S_4
a	1	0	0	1
b	0	0	1	0
c	0	1	0	1
d	1	0	1	1
e	0	0	1	0

- Universal set: {a, b, c, d, e}
- Matrix represents sets chosen from universal set
- S1 = {a, d}, S2 = {c}, S3 = {b, d, e} and S4 = {a, c, d}
- Example: **rows are products** and **columns are customers**, represented by **set of items** they bought
- **Jacquard similarity of S1, S4**: intersection/union = 2/3.

Example: Jaccard Similarity of Columns

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

Sim (C_1, C_2) =

$2/5 = 0.4$

Outline: Finding Similar Columns

1. Compute **signatures** of columns = **small summaries** of columns
2. Examine **pairs of signatures** to find similar signatures
   - **Essential:** **similarities of signatures** and **columns** are related
3. **Optional:** check that columns with similar signatures are really similar.

Warnings

1. Comparing **all pairs of signatures** may take **too much time**, even if not too much space
   - A job for Locality-Sensitive Hashing

2. These methods can produce false negatives, and even false positives (if the optional check is not made).

Signatures

- Key idea: "**hash**" each column *C* to a small ***signature*** *Sig*(C), such that:

1. *Sig* (C) is **small** enough that we can fit a signature in **main memory** for each column
2. *Sim* (C_1, C_2) is the same as the "**similarity**" of *Sig* (C_1) and *Sig* (C_2).

Four Types of Rows

- Given columns 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

- Also, a = # rows of type a , etc.

- Note $Sim(C_1, C_2) = a/(a+b+c)$
  - Jacquard similarity: intersection/union
  - a is intersection, a+b+c is union

Minhashing

1. To ***minhash*** a set represented by a column of the matrix, **pick a random permutation of the rows**
2. **Define "hash" function *h* (*C*) = the number of the first (in the permuted order) row in which column *C* has 1**
3. Use several (e.g., 100) independent hash functions to **create a signature**.

Minhashing Example (3.7)

Element	S_1	S_2	S_3	S_4
a	1	0	0	1
b	0	0	1	0
c	0	1	0	1
d	1	0	1	1
e	0	0	1	0

Permute

Element	S_1	S_2	S_3	S_4
b	0	0	1	0
e	0	0	1	0
a	1	0	0	1
d	1	0	1	1
c	0	1	0	1

1. To minhash a set represented by a column of the characteristic matrix, pick a **permutation of the rows**
2. The **minhash value** of any column is the number of first row, in permuted order, in which column **has a 1**
3. For set S1, first 1 appears in row a, so:

- h(S1) =
- h(S2) =
- h(S3) =
- h(S4) =

Minhashing Example (3.7)

Element	S_1	S_2	S_3	S_4
a	1	0	0	1
b	0	0	1	0
c	0	1	0	1
d	1	0	1	1
e	0	0	1	0

Permute

Element	S_1	S_2	S_3	S_4
b	0	0	1	0
e	0	0	1	0
a	1	0	0	1
d	1	0	1	1
c	0	1	0	1

1. To minhash a set represented by a column of the characteristic matrix, pick a **permutation of the rows**
2. The **minhash value** of any column is the number of first row, in permuted order, in which column **has a 1**
3. For set S1, first 1 appears in row a, so:

- h(S1) = a
- h(S2) = c
- h(S3) = b
- h(S4) = a

Minhashing Example

Input matrix

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

Signature matrix *M*

2	1	2	1

Minhashing Example

Input matrix

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

Signature matrix *M*

2	1	2	1
2	1	4	1

Minhashing Example

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

Surprising Property: Connection between Minhashing and Jaccard Similarity

- The probability that minhash function for a **random permutation of rows** produces same value for two sets equals **Jaccard similarity** of those sets
  - **Probability that *h* (C_1) = *h* (C_2) is the same as *Sim* (C_1, C_2)**
- Recall four types of rows:

	C_1	C_2
a	1	1
b	1	0
c	0	1
d	0	0

- **Sim(C_1, C_2) for both Jacquard and Minhash are *a* /(*a* +*b* +*c*)!**
  - **Why?** Look down the permuted columns 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. (Don't count the *type-d* rows).

Similarity for Signatures

- **Sets represented** by characteristic **matrix M**
- **To represent sets:** pick at random some number **n of permutations** of the rows of M
  - **100 permutations** or several hundred
- Call **minhash** functions determined by these permutations $h_1, h_2, \ldots, h_n$
- From **column representing set S**, construct **minhash signature for S**:
  - vector $[h_1(S), h_2(S), \ldots, h_n(S)]$, usually represented as column
- Construct a ***signature matrix***: **i^{th} column of M** replaced by **minhash signature for i^{th} column**
- **The *similarity of signatures* is the fraction of the hash functions in which they agree.**

Min Hashing – Example

Input matrix

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

Signature matrix *M*

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

Similarities:

¾

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

2/3

Min Hashing – Example

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

Input matrix

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 Hashing – Example

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

Input matrix

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

Minhash Signatures

- Pick (say) **100 random** permutations of the rows
- Think of ***Sig*** **(C)** as a column vector
- Let ***Sig*** **(C)[i]** =

  according to the ***i* th permutation**, the number of the first row that has a **1 in column** *C.*

Implementation – (1)

- **Not feasible to permute** a large characteristic matrix explicitly
  - Suppose **1 billion rows**
  - Hard to pick a **random permutation from 1…billion**
  - Representing a random permutation **requires 1 billion entries**
  - Accessing rows in permuted order leads to thrashing

- **Can simulate the effect of a random permutation by a random hash function**
  - **Maps row** numbers to as many buckets as there are rows
  - May have **collisions on buckets**
  - **Not important as long as number of buckets is large.**

Implementation – (2)

- A good approximation to permuting rows: pick around 100 hash functions
- For each:
  - column c (set representing a document)
  - hash function h_i
- Keep a "**slot**" in signature matrix M (i,c)
- Intent: **M (i,c) will become the smallest value of $h_i(r)$ for which column c has 1 in row r**
  - $h_i(r)$ gives order of rows for i^{th} permuation.

Implementation – (3)

```
for each row r
    for each column c
        if c has 1 in row r
            for each hash function h_i do
                if h_i (r ) is a smaller value than M (i, c ) then
                    M (i, c ) := h_i (r );
```

Computing Minhash Signatures: Example 3.8

Row	S_1	S_2	S_3	S_4	$x+1 \mod 5$	$3x+1 \mod 5$
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

Two hash functions give permutations of rows:
h1 = x+1 mod 5, h2 = 3x +1 mod 5

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

Initial signature matrix

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

For row 0: Replace existing signature values with lower hash values for S1 and S4, since both have 1 in row

Computing Minhash Signatures: Example 3.8 (part 2)

Row	S_1	S_2	S_3	S_4	$x+1 \mod 5$	$3x+1 \mod 5$
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	1	∞	2	1
h_2	1	∞	4	1

For row 1: replace h1 and h2 values for S3, since row has a 1 and values are lower

	S_1	S_2	S_3	S_4
h_1	1	3	2	1
h_2	1	2	4	1

For row 2: replace values for S2 since set has a 1 value. Do not replace values for S4, because existing values are lower

Computing Minhash Signatures: Example 3.8 (part 3)

Row	S_1	S_2	S_3	S_4	$x+1 \mod 5$	$3x+1 \mod 5$
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

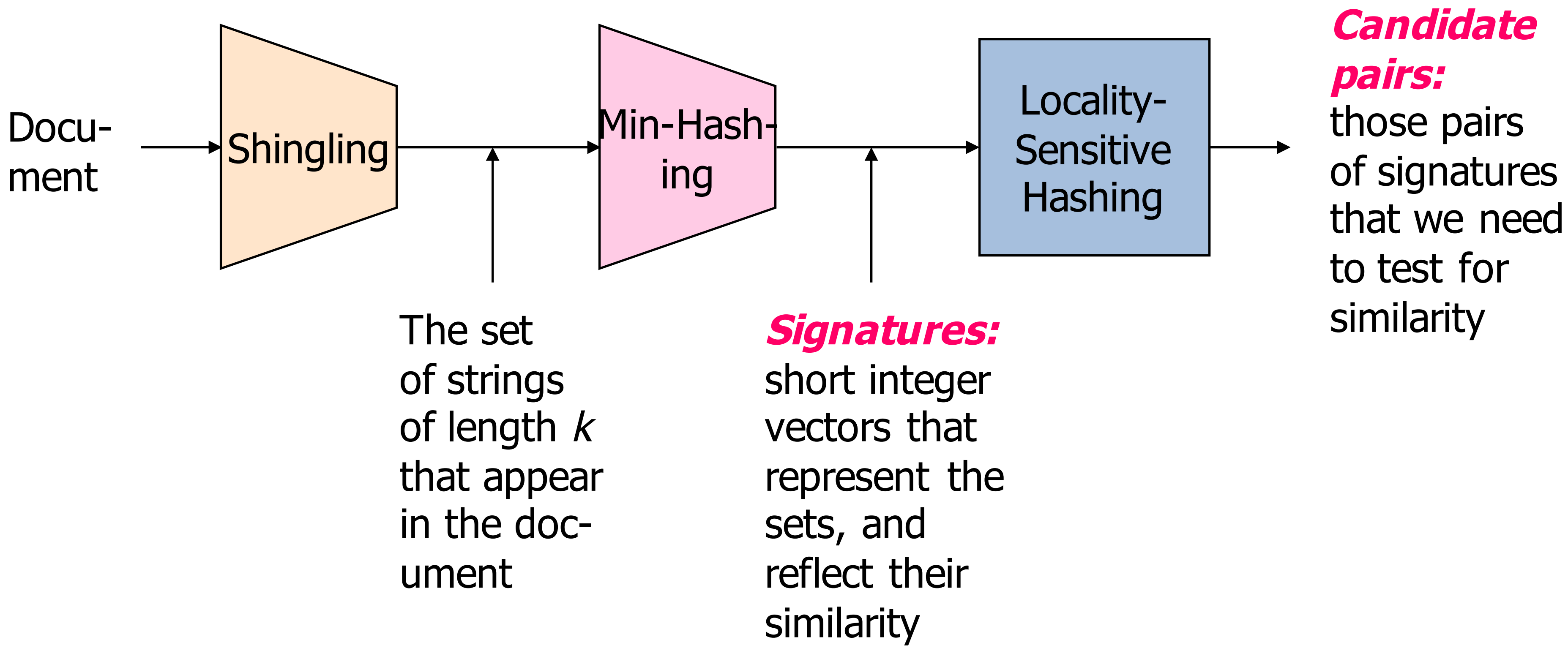
	S_1	S_2	S_3	S_4
h_1	1	3	2	1
h_2	0	2	0	0

For row 3: don't replace h1 values--all are below 4; replace h2 values with 0 for S1, S3, S4

	S_1	S_2	S_3	S_4
h_1	1	3	0	1
h_2	0	2	0	0

For row 4: replace h1 value for S3, don't replace h2 value since current value is lower

Note: result is same as permutations to find first 1



Locality Sensitive Hashing

Step 3: ***Locality-Sensitive Hashing:***
Focus on pairs of signatures likely to be from similar documents

Example

Row	C1	C2
1	1	0
2	0	1
3	1	1
4	1	0
5	0	1

$h(x) = x \bmod 5$
$g(x) = 2x+1 \bmod 5$

	Sig1	Sig2
$h(1) = 1$	1	-
$g(1) = 3$	3	-
$h(2) = 2$	1	2
$g(2) = 0$	3	0
$h(3) = 3$	1	2
$g(3) = 2$	2	0
$h(4) = 4$	1	2
$g(4) = 4$	2	0
$h(5) = 0$	1	0
$g(5) = 1$	2	0