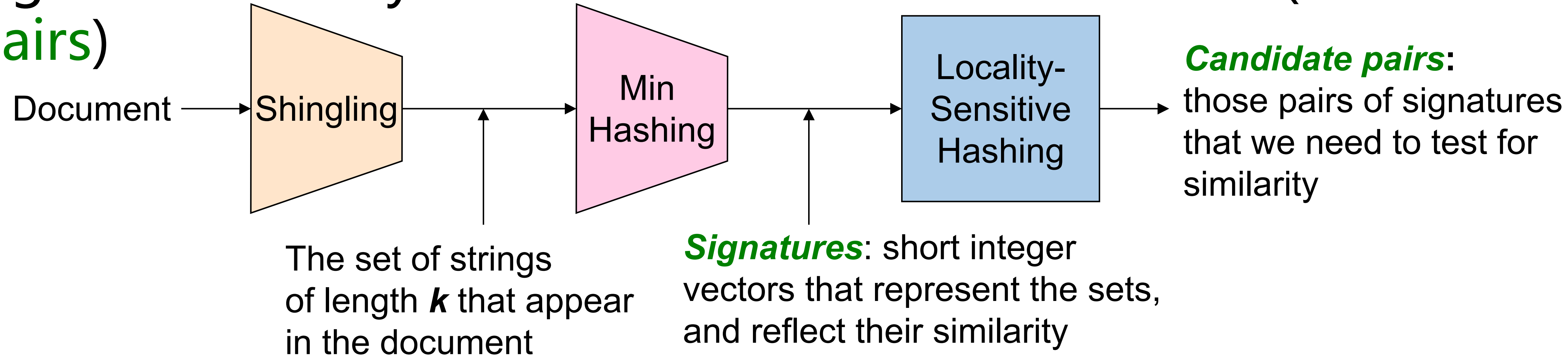
4.1 Three Steps for Finding Similar Documents

- Section 4.2: ***Shingling,*** convert documents to set representation (Boolean vector)
- Section 4.3: ***Min-Hashing,*** convert large sets to short signatures, while preserving similarity
- Section 4.4: ***Locality-Sensitive Hashing,*** focus on pairs of signatures likely to be from similar documents (candidate pairs)



Section 4.2: Shingling

Convert documents to sets

Content

- Step 1: Shingling, convert documents to sets
- **Approach 1 (simple approaches):**
  - Document = set of words appearing in document
  - Document = set of "important" words
  - But don't work well for this application. Why?
  - **Ans: Need to account for ordering of words!**
- **Approach 2: Shingles**

4.2.1 Define Shingles

- A ***k*-shingle** (or ***k*-gram, k分词,** 或称**k子串**) for a document is a sequence of k tokens that appears in the document
  - Tokens can be **characters**, **words** or something else, depending on the application
  - Assume tokens = characters for examples

- **Example:**
  - **k=2**, document $\mathbf{D_1}$ = abcab. What are 2-shingles?
  - 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}

- **Then, represent a document by its set of k-shingles.**

4.2.1 Define Shingles

- For **white space** (blank, tab, newline, etc) in documents, there are several options to deal with.
  - Common: replace any sequence of one or more white space characters by a single blank.
  - Or: eliminate blanks.
- **Example:**
  - **k=9**. Document $\mathbf{D_2}$ = "The pane was ready for touch down" ,
  - Document $\mathbf{D_3}$ = "The quarterback scored a touchdown" ,
  - When retaining the blanks, **S(**$\mathbf{D_2}$**)** has "touch dow" and "ouch down" and **S(**$\mathbf{D_3}$**)** has "touchdown"
  - When eliminating the blanks, both have " touchdown"

4.2.1 Define Shingles

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

- **Working Assumption, Caveat(提醒):** 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

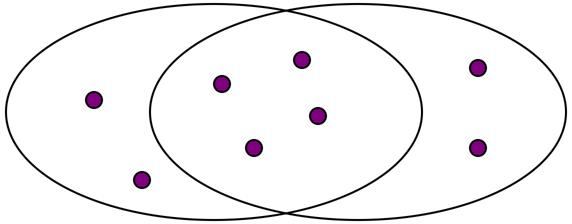
4.2.2 Compressing Shingles

华中科技大学 计算机科学与技术学院 School of Computer Science &Technology, HUST

- 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: S(D1) = {ab, bc, ca}
  - Hash the singles: h(D1) = {1, 5, 7}
- **Benefits of shingles:**
  - Documents that are intuitively similar will have many shingles in common
  - Changing a word only affects k-shingles within distance k-1 from the word

- **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 (Jaccard 相似度):**

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



e.g. 4/8

4.2.3 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***
- Typical matrix is sparse!
- Column similarity is the **Jaccard similarity** of the corresponding sets (rows with value 1). Each document is a column:
  - **Example: sim(C_1 ,C_2) = ?**
    - Size of intersection = 3; size of union = 6, Jaccard similarity (not distance) = 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

Characteristic/Boolean matrix (特征/布尔矩阵)

Note: We don’t really construct the matrix; just imagine it exists

4.2.3 Example

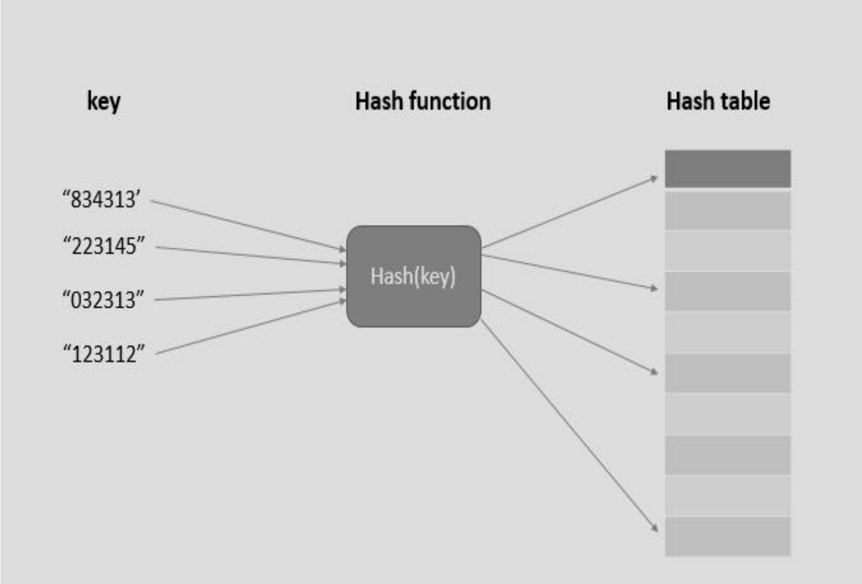
- **Example: Suppose we need to find near-duplicate documents among $N = 1$ million (一百万) documents**

- Naïvely, we would have to compute **pairwise Jaccard similarities** for **every pair of docs**
  - $N(N-1)/2 \approx$ **5*10^{11}** comparisons
  - At 10^5 secs/day and 10^6 comparisons/sec, it would take **5 days**

- For $N = 10$ million, it takes more than a year…

Outline: Finding Similar Columns

- **So far:**
  - Documents → 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**



Section 4.3: Minhashing

Convert large sets to short signatures, while preserving similarity

Content

1. Min-Hashing
2. The Min-Hash Property
3. Implementation Trick for Min-Hash

4.3.1 Finding Similar Columns

华中科技大学 计算机科学与技术学院 School of Computer Science &Technology, HUST

- **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 Section 4.4: **LSH (Locality-Sensitive Hashing)**
    - These methods can produce false negatives(伪反例), and even false positives (伪正例, if the optional check is not made)

False positive(伪正例):某些文档对不是相似的, 但它被认为是相似的
False negative(伪反例):某些文档对是相似的, 但它却认为是不相似的

4.3.1 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)** $sim(C_1, C_2)$ is the same as the "similarity" of signatures $h(C_1)$ and $h(C_2)$

- **Goal: Find a hash function *h(•)* such that:**
  - If $sim(C_1,C_2)$ is high, then with high prob. $h(C_1) = h(C_2)$
  - If $sim(C_1,C_2)$ 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!**

4.3.1 Min-Hashing

- **Goal: Find a hash function $h(\cdot)$ such that:**
  - if $sim(C_1,C_2)$ is high, then with high prob. $h(C_1) = h(C_2)$
  - if $sim(C_1,C_2)$ 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 (最小哈希)**

4.3.1 Min-Hashing

- Imagine the rows of the boolean matrix permuted under **random permutation (随机行排列, 随机行置换)** π
  - Thought experiment – not real

- 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 for each column
- Result is a signature matrix (签名矩阵): Columns = sets (e.g., documents), Rows = Min-hash values for each permutation π

4.3.1 Min-Hashing Example

Note: Another (equivalent) way is to store row indexes:

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

$h_1(1)=2$ (permutation 1, column 1)

2nd element of the permutation is the first to map to a 1

这儿2表示第1行换到第2行去的随机置换, 其他类似

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 (签名矩阵) *M*

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

$h_2(3)=4$ (permutation 2, column 3)

4th element of the permutation is the first to map to a 1

4.3.2 The Min-Hash Property

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

0	0
0	0
1	**1**
0	0
0	1
1	0

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

4.3.2 The Min-Hash Property

- **<u>Claim:</u>** $\Pr[h_\pi(C_1) = h_\pi(C_2)] = sim(C_1, C_2)$
- **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:** $sim(C_1, C_2) = a/(a + b + c)$
- **Then:** $\Pr[h(C_1) = h(C_2)] = Sim(C_1, C_2)$
  - 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

0	0
0	0
1	**1**
0	0
0	1
1	0

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

4.3.2 Similarity for Signatures

- We know: $\Pr[h_\pi(C_1) = h_\pi(C_2)] = sim(C_1, C_2)$
- 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 Min-Hash signatures (最小哈希签名)

华中科技大学 计算机科学与技术学院 School of Computer Science &Technology, HUST

Example: Minhash Signatures

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 (签名矩阵) *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

4.3.2 Min-Hash Signatures

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

- **Pick K=100 random permutations of the rows**
- Think of ***sig*(C)** as a column vector
- **s*ig*(C)[i]** = according to the *i*-th permutation, the index of the first row that has a 1 in column *C* (签名矩阵中第i个哈希函数在第C列上的元素)

$$\boldsymbol{sig(C)[i] = \min(\pi_i(C))}$$

- **Note:** The sketch (signature) of document *C* is small **~100 bytes!**
- **We achieved our goal! We "compressed" long bit vectors into short signatures**

4.3.3 Implementation Trick for Min-Hash

华中科技大学 计算机科学与技术学院 School of Computer Science &Technology, HUST

- **Permuting rows even once is prohibitive** ☹
- **Row hashing!** ☺
  - Pick **K = 100** hash functions $\boldsymbol{k_i}$
  - Ordering under $\boldsymbol{k_i}$ gives a random row permutation!
- **Q: How to pick a random hash function h(x)?**
- **A: Universal hashing:**
- $h_{a,b}(x)=((a\cdot x+b) \textit{ mod } p) \text{ mod } N$, *where:*
  - a,b … random integers
  - p … prime number (素数p > N)

One-pass implementation

- For each column C and hash-function k_i keep a "slot" for the min-hash value
- Initialize all $sig(C)[i] = \infty$
- **Scan rows looking for 1s**
  - Suppose row j has 1 in column C
  - Then for each k_i:
    - If $k_i(j) < sig(C)[i]$, then $sig(C)[i] \leftarrow k_i(j)$

Signature matrix M

∞	∞	∞	∞
∞	∞	∞	∞
∞	∞	∞	∞

k_i:第i个随机选择的哈希函数
$sig(C)[i]$: 签名矩阵中第i个哈希函数在第C列上的元素
$k_i(j)$: 对第j行进行第i个哈希函数的计算结果值

4.3.3 Example

❑Example

Sig1	Sig2
∞	∞
∞	∞

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

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

4.3.3 Example

华中科技大学 计算机科学与技术学院 School of Computer Science & Technology, HUST

❑Example

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

$h(x) = x \; mod \; 5$

$g(x) = (2x + 1) \; mod \; 5$

	Sig1	Sig2
$h(0) = 0$	0	∞
$g(0) = 1$	1	∞

$h(1) = 1$	0	1
$g(1) = 3$	1	3

$h(2) = 2$	0	1
$g(2) = 0$	0	0

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

$h(4) = 4$	0	1
$g(4) = 4$	0	0

Then, final signature matrix M is:

0	1
0	0