Finding Similar Sets

Applications

Shingling

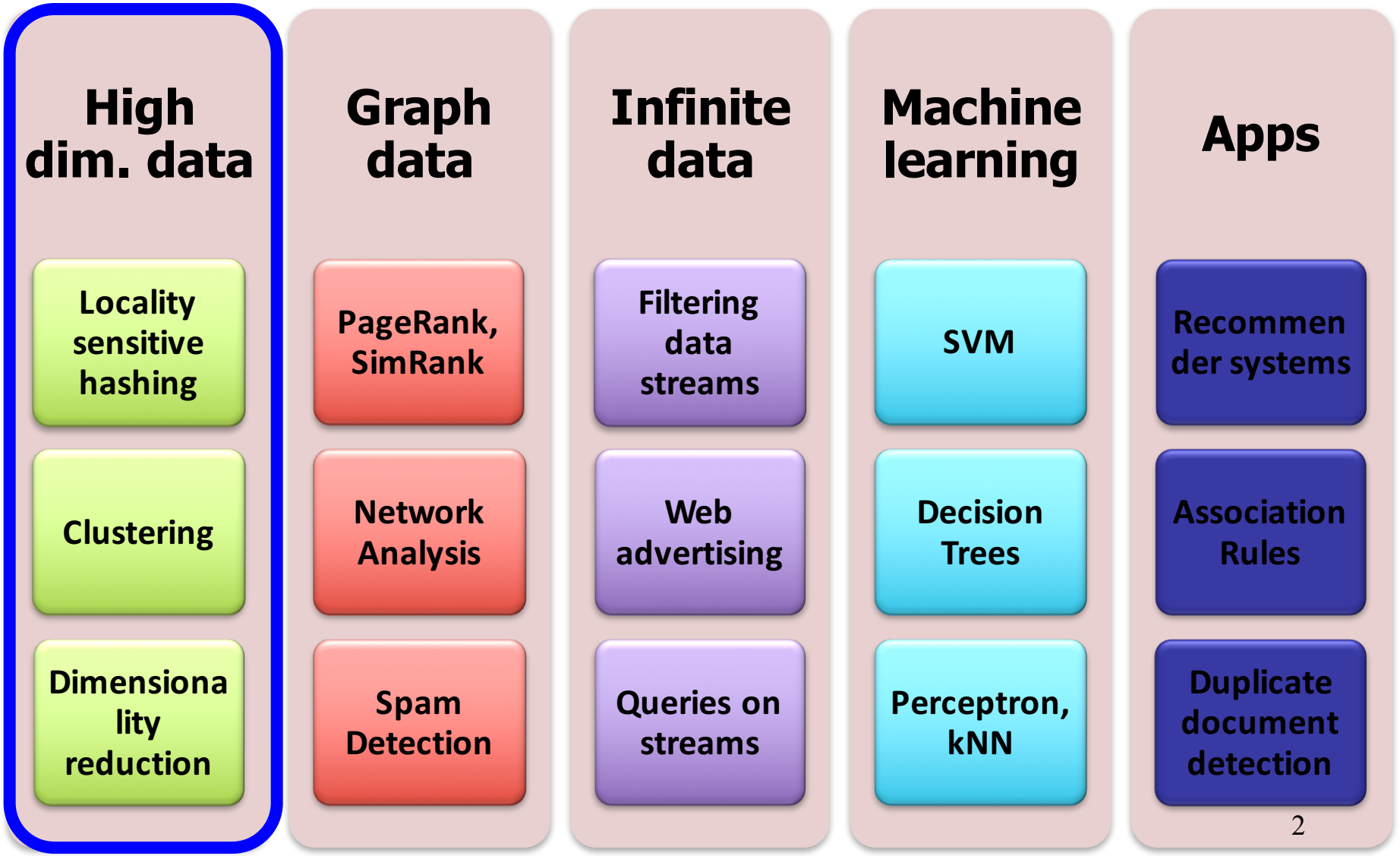
Minhashing

Locality-Sensitive Hashing

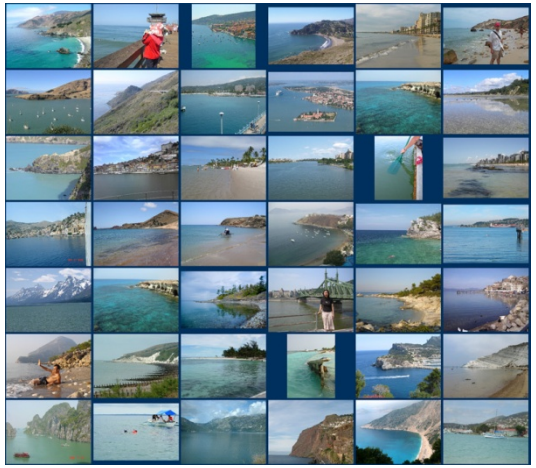
Anna Farzindar, Ph.D.

Thanks for source slides and material to: J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets

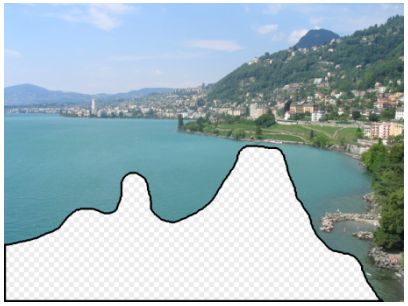
New thread: High dim. data



Scene Completion Problem



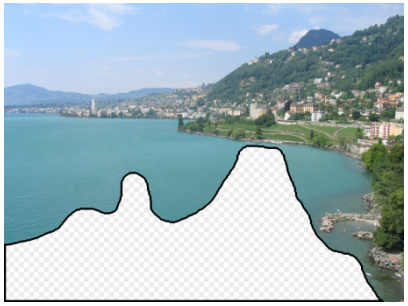
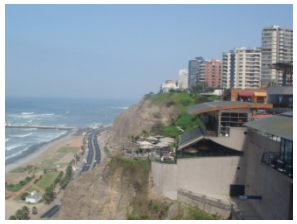
Scene Completion Problem

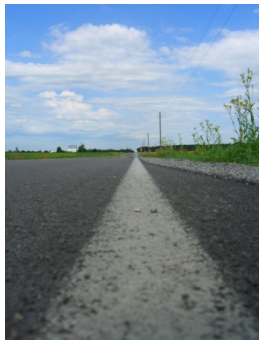


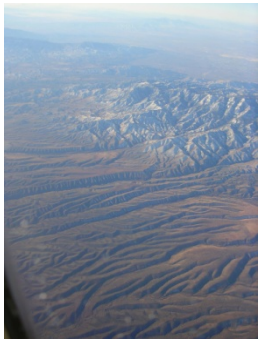
Scene Completion Problem





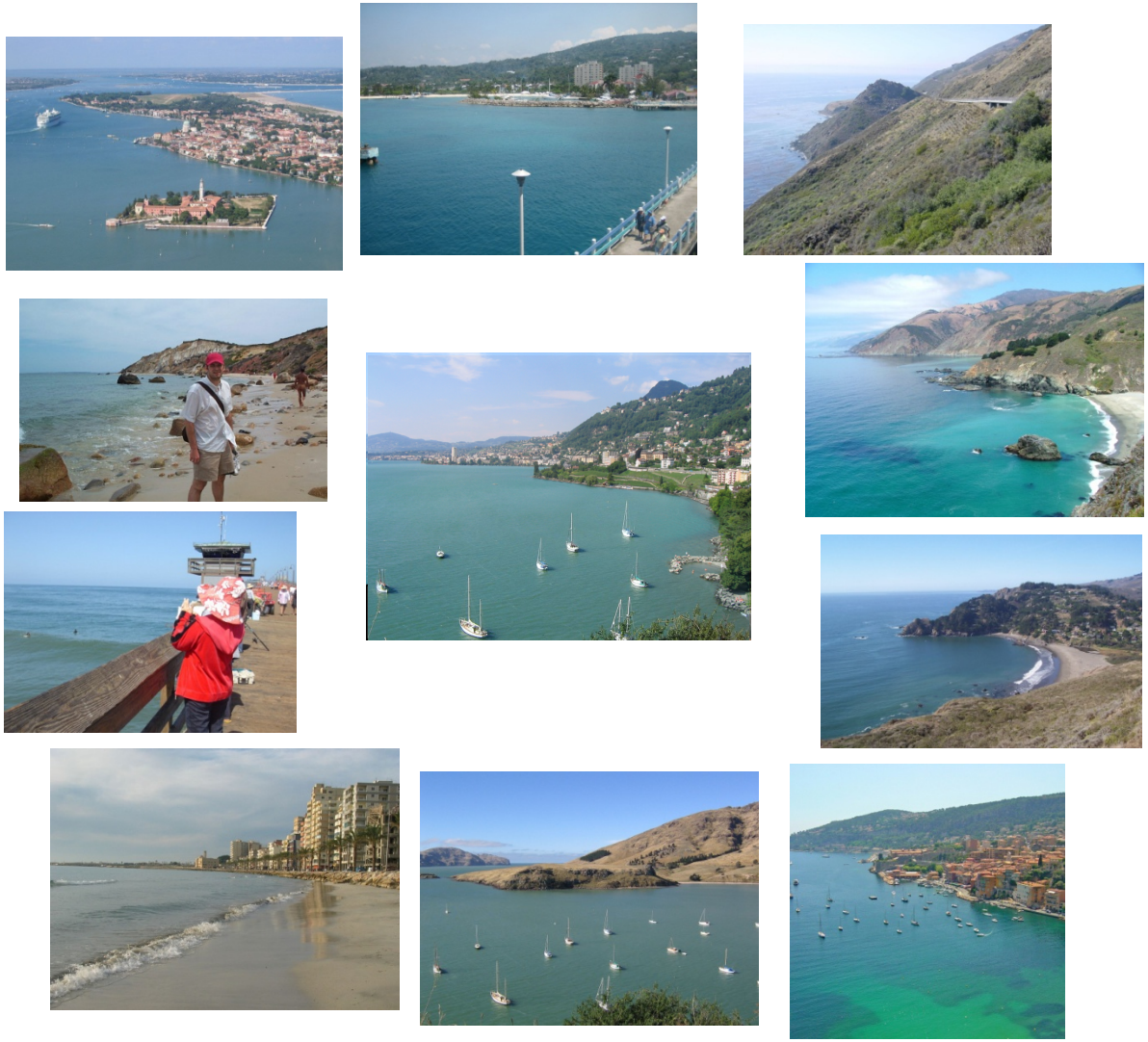





10 nearest neighbors from a collection of 20,000 images

Scene Completion Problem



10 nearest neighbors from a collection of 2 million images

A Common Metaphor

- **Many problems can be expressed as finding "similar" sets:**
  - **Find near-neighbors in high-dimensional space**
- **Examples:**
  - **Pages with similar words**
    - For duplicate detection, classification by topic
  - **Movie Rating, NetFlix users with similar tastes in movies**
    - For recommendation systems
  - **Customers who purchased similar products**
    - Products with similar customer sets
  - **Images with similar features**
    - Users who visited similar websites.

Problem for Today's Lecture

- **Given: High dimensional data points $x_1, x_2, \ldots$**
  - **For example:** Image is a long vector of pixel colors

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 2 & 1 \\ 0 & 1 & 0 \end{bmatrix} \rightarrow [1\ 2\ 1\ 0\ 2\ 1\ 0\ 1\ 0]$$

- **And some distance function $d(x_1, x_2)$**
  - Which quantifies the "distance" between x_1 and x_2
- **Goal:** Find **all pairs of data points** (x_i, x_j) that are within some distance threshold $d(x_i, x_j) \leq s$
- **Note:** Naïve solution would take $O(N^2)$ ☹
  - where N is the number of data points
  - $O(N^2)$ represents an algorithm whose performance is directly proportional to the **square of the size** of the **input data** set
- **MAGIC: This can be done in $O(N)$!! How?**

Finding Similar Items

Finding Similar Documents

- Given a body of **documents**, e.g., the Web, find **pairs of documents** with a lot of **text in common**, such as:
  - **Mirror sites**, or approximate mirrors,
    - **Application:** Don't want to show both in a search.
  - **Plagiarism**, including large quotations
  - **Similar news articles** at many news sites,
    - **Application:** Cluster articles by "same story."

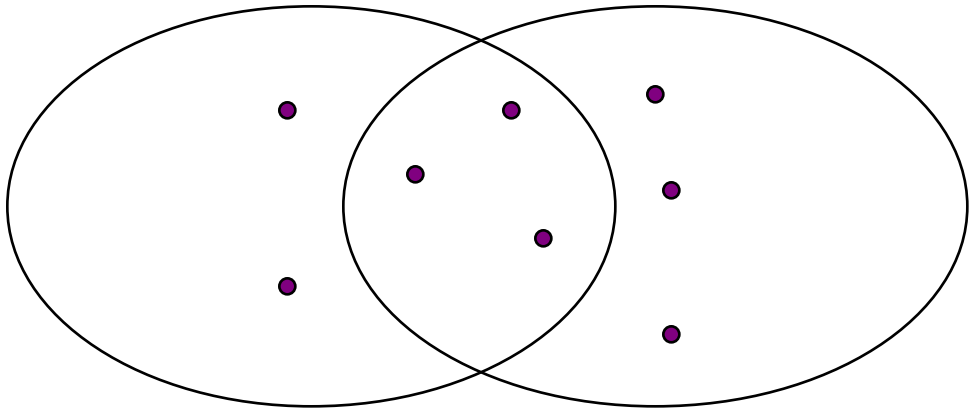
Distance Measures

- **Goal: Find near-neighbors in high-dimensional space**
  - We formally define "near neighbors" as points that are a "small distance" apart
- For each application, we first need to define what "**distance**" means
- **Today: Jaccard distance/similarity.**

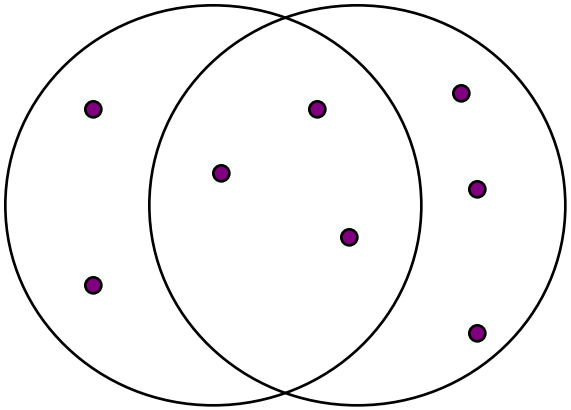
Jaccard Similarity of Sets

- The *Jaccard similarity* of two sets is the **size of their intersection** divided by the **size of their union**

$$Sim\ (C_1, C_2) = |C_1 \cap C_2| / |C_1 \cup C_2|$$



Example: Jaccard Similarity



3 in intersection
8 in union

Jaccard similarity = 3/8

- **Jaccard distance** = 1 – Jaccard Similarity or 5/8 in this example

Application: Collaborative Filtering

- Online purchase
  - Recommend items from other buyers
  - ⇒Need to find similar buyers or items
  - ⇒Buyer-to-item or item-to-item recommendation
- Buyer = a set of items he/she purchased
  - Similar buyers: users who bought many common items
- Item = a set of buyers who purchased it
  - Similar items: items having many common buyers.

Collaborative Filtering @Amazon

- Item-to-item recommendation

Customers Who Bought This Item Also Bought

Page 1 of 19

Photive iPad Air Smart Case. Lightweight Smart Cover Case for the New iPad Air with Built in...
848
$19.95 Prime

iPad Air Case, SUPCASE Heavy Duty Beetle Defense Series Full-body Rugged Hybrid Protective Case...
2,193
$19.99 Prime

Tech Armor Apple iPad Air 2 / iPad Air (first generation) High Defintion (HD) Clear Screen...
4,686
#1 Best Seller in Tablet Screen Protectors
$9.95 Prime

Fintie iPad Air Case - SmartShell Case for Apple iPad Air (iPad 5) 2013 Model, Ultra Slim...
2,471
$9.99 Prime

Fintie Apple iPad Air Case - 360 Degree Rotating Stand Case Cover with Auto Sleep / Wake Feature for...
1,795
$14.99 Prime

Finding Similar Buyers

Who is most similar to B1?

B1	{A, B}
B2	{A, B, D}
B3	{A, C, D}

Jaccard(B1, B2) =2/3
Jaccard(B1, B3) =1/4

Buyer-to-item recommendation

B2 most similar to B1
B2 also bought D

Recommend **D to B1**

Who is most similar to B2?

Jaccard(B2, B1) = Jaccard(B1, B2)
Jaccard(B2, B3) = ? 1/2

Who is most similar to B3?

Jaccard(B3, B1) = Jaccard(B1, B3)
Jaccard(B3, B2) = Jaccard(B2, B3)

Finding Similar Items

B1	{A, B}
B2	{A, B, D}
B3	{A, C, D}

A	{B1, B2, B3}
B	{B1, B2}
C	{B3}
D	{B2, B3}

Buyer as a set

Item as a set

Finding Similar Items

Which item is most similar to A?

A	{B1, B2, B3}
B	{B1, B2}
C	{B3}
D	{B2, B3}

Jaccard(A, B) =2/3 → A is most similar to B and D
Jaccard(A, C) =1/3
Jaccard(A, D) =2/3

Which item is most similar to B?

Jaccard(B, A) = Jaccard(A, B) = 2/3
Jaccard(B, C) = 0 → B is most similar to A
Jaccard(B, D) = 1/3

B1	{A, B}
B2	{A, B, D}
B3	{A, C, D}

Item-to-item recommendation

B1 has bought A, B (or put them in basket)
A is most similar to B and D
B is most similar to A

Recommend: D to B1

Formulated as frequent itemset problem?

- Find similar items
  - Items bought together by many users
  - ⇒User = transaction
  - ⇒Similar items = frequent item pair

B1	{A, B}
B2	{A, B, D}
B3	{A, C, D}

- Find similar users
  - Users that bought many common items
  - Item = transaction
  - => Find frequent user pairs

A	{B1, B2, B3}
B	{B1, B2}
C	{B3}
D	{B2, B3}

Application: Collaborative Filtering

- Recommend movies
  - Recommend similar movies or
  - Movies from similar users

- User = a set of movies he/she has watched

- Movie = a set of users who has watched it

What if we consider ratings too?

- User A watched movie HP1 (Harry Potter 1)
  - And rated it 4
- May need different similarity function/solution

⇒Recommendation systems (Chapter 9)

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

Task: Finding Similar Documents

- **Goal: Given a large number (in the millions or billions) of documents, find "near duplicate" pairs**
- **Applications:**
  - Mirror websites, or approximate mirrors
    - Don't want to show both in search results
  - Similar news articles at many news sites
    - Cluster articles by "same story"
- **Problems:**
  - Many **small pieces** of one document can appear out of order in another
  - **Too many** documents **to compare** all pairs
  - Documents are so large or so many that they **cannot fit in main memory.**

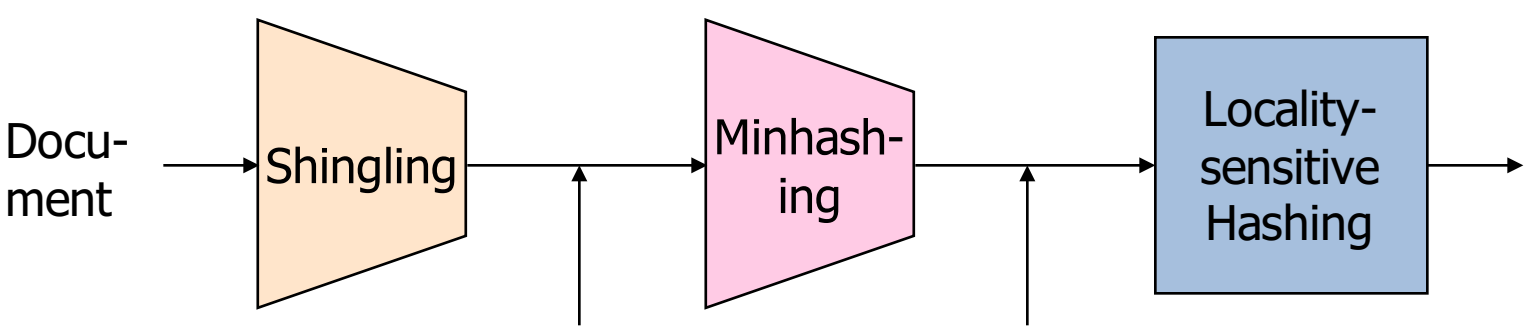
3 Essential Steps for Finding Similar Docs

1. ***Shingling:*** Convert documents **to sets**

2. ***Min-Hashing:*** Convert large sets to short **signatures**, while **preserving similarity**

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

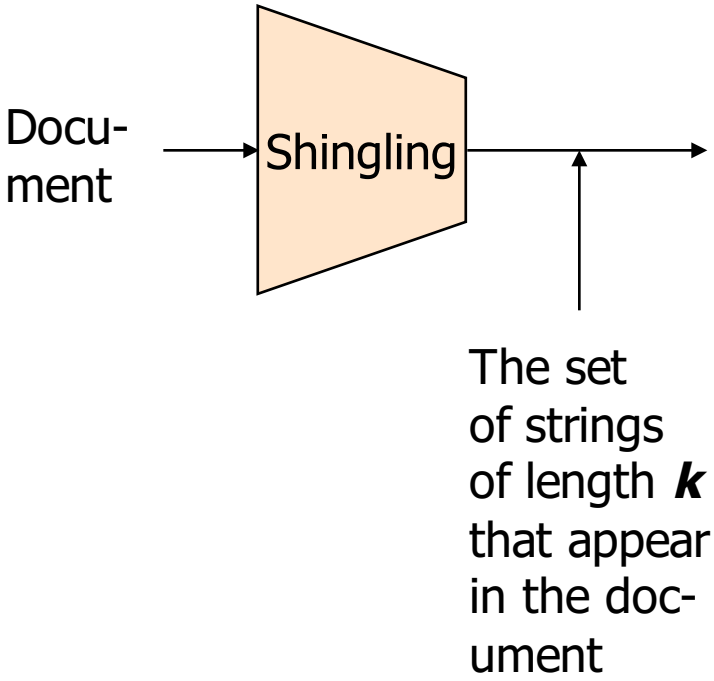
The Big Picture



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

The set of strings of length *k* that appear in the doc-ument

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



Shingling

Step 1: ***Shingling:*** Convert documents to sets

Documents as High-Dimensional Data

- **Step 1: *Shingling:* Convert documents to sets**

- **Simple approaches:**
  - Document = set of words appearing in document
  - Document = set of "important" words
  - Don't work well for this application. Why?

- **Need to account for ordering of words!**
- A different way: **Shingles!**

Define: Shingles

- A *k*-shingle (or *k*-gram) for a document is a sequence of *k* tokens that appears in the doc
  - 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
  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}.

Shingles

- Web page as a string of characters
- Shingle = subsequence of k-characters

- Web page = abcdabd, k = 2
- 2-shingles
  - **ab, bc, cd, da, bd**

- Max # of k-shingles for a page of n characters?
  - **7 characters, 6 max 2-shingles, two identical, 5 unique**
  - **Max = 7 – 2 + 1 = 6**
  - **N – k + 1 ~ n**

White Spaces

- Better not omit them
- Could turn multiple into one

- D1: “scored a touch down” => “scored a touch down”
- D2: “touchdown at last”

- D1 and D2 have a common 9-shingle if space omitted

Shingle Size

- Too small
  - Many documents will falsely become similar
- Too big
  - Might miss truly similar documents.

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 (eg. Email, Tweet)
  - ***k*** = 10 is better for long documents
  - news articles, blog posts (in between)

- May want to **compress long shingles**.

Compressing Shingles

- To **compress long shingles**, we can **hash** them to (say) 4 bytes
  - Called ***tokens***
- **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}.

Why is compression needed?

- How many k-shingles?
  - Rule of thumb: imagine **20 characters** in alphabet
    - **26 characters + whitespace = 27**
    - **since "z,q,x" used rarely**
    - A **k-shingle** is a sequence of k consecutive words (or k-gram)
  - Estimate of number of k-shingles is 20^k
  - 4-shingles: 20^4 or 160,000 or $2^{17.3}$
  - 9-shingles: 20^9 or 512,000,000,000 or $\mathbf{2^{39}}$
- Assume we use **4 bytes** to represent a **bucket**
- Buckets numbered in range 0 to $2^{32} - 1$
- Much smaller than possible number of **9-shingles** and represent each shingle with 4 bytes, not 9 bytes
  - Compression.

Thought Question

- Why is it better to hash 9-shingles (say) to 4 bytes than to use 4-shingles?
- Hint: How random are the 32-bit sequences that result from 4-shingling?

Why hash 9-shingles to 4 bytes rather than use 4-shingles?

- With 4-shingles, most sequences of four bytes are unlikely or impossible to find in typical documents
- Effective number of different shingles much less than $2^{32} - 1$
- With 9-shingles, 2^{39} possible shingles
  - Many more than 2^{32} buckets
- After hashing, may get any sequence of 4 bytes.

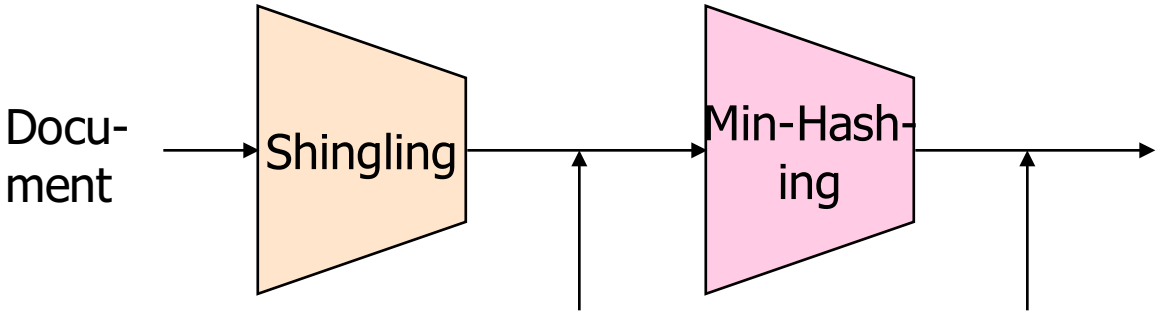
Similarity Metric for Shingles

- **Document D_1 is a set of its k-shingles $C_1=S(D_1)$**
- Equivalently, each document is a vector of 0s,1s in the space of *k*-shingles
  - Each unique shingle is a dimension
  - Vectors are very sparse,
- **A natural similarity measure is the Jaccard similarity.**

Motivation for Minhash/LSH

Use k-shingles to create Signatures: short integer vectors that represent sets and reflect their similarity

- **Suppose we need to find near-duplicate documents among million documents**
- Naïvely, we would have to compute **pairwise Jaccard similarities** for **every pair of docs**
  - i.e, **N (N-1) /2 ≈ 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…



The set of strings of length *k* that appear in the doc-ument

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

MinHashing

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

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

Four Types of Rows

- Given columns C1 and C2, rows may be classified as:

	C1	C2
a	1	1
b	1	0
c	0	1
d	0	0

  - Also, *a* = # rows of type *a* , etc.
- Note *Sim*(C1, C2) = *a*/(*a* +*b* +*c*).

When Is Similarity Interesting?

1. When the **sets are so large** or so many that they **cannot fit** in main **memory**
2. Or, when there are **so many sets** that **comparing all pairs** of sets takes **too much time**
3. Or both.

Summary : The Big Picture