BIG DATA ANALYTICS
Finding Similar Items

ESSEC BUSINESS SCHOOL

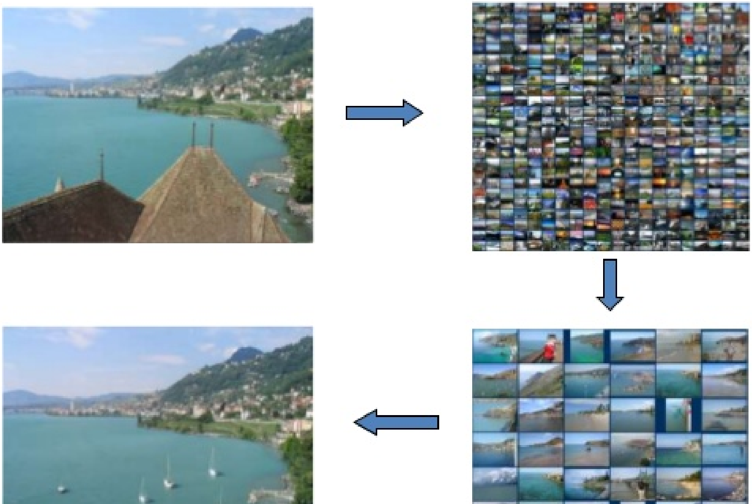
Finding similar items

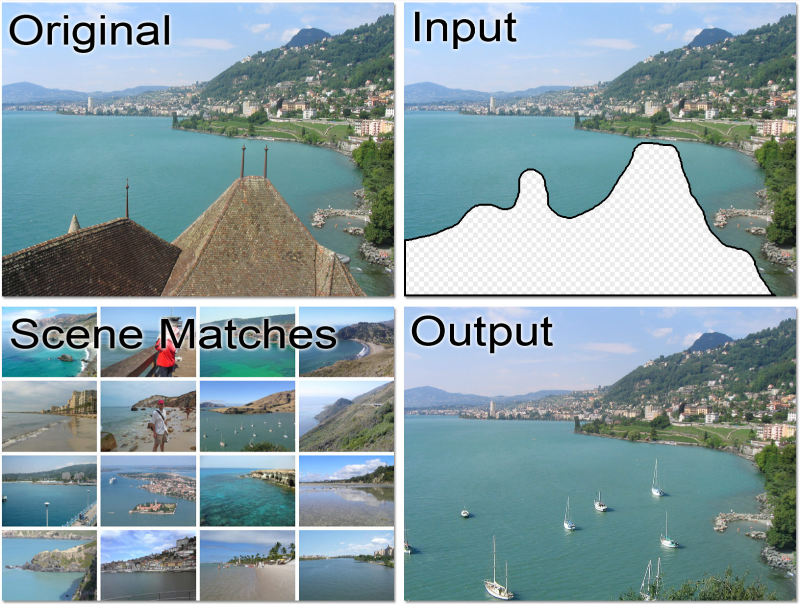
- A fundamental problem
- Web pages: finding near-duplicate pages:
  - plagiarisms
  - mirrors: have almost the same content

Articles from the Same Source

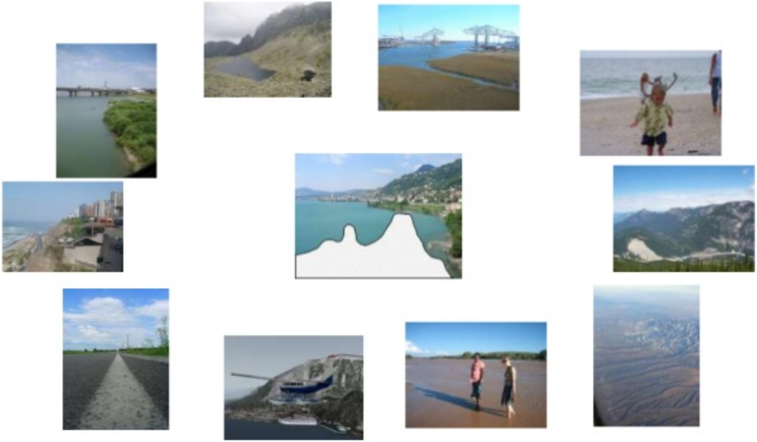
- A news article that gets distributed to many newspapers
- Each newspaper changes the article somewhat.
- The core: the original article
- Goal: find all versions of such an article

Scene Completion Problem




Original
Input
Scene Matches
Output

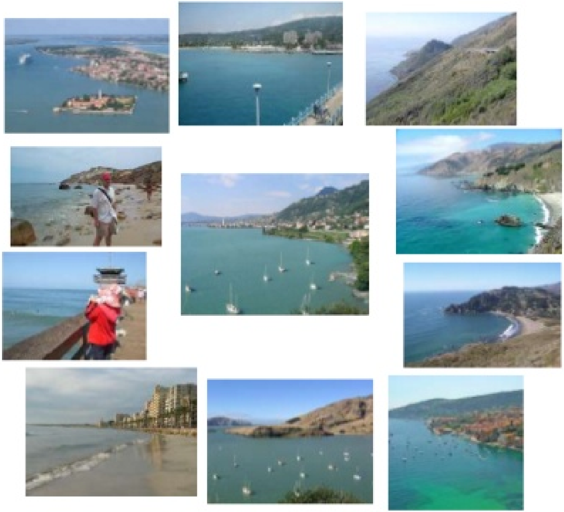
Scene Completion Problem



10 nearest neighbors from a collection of 20,000 images

http://graphics.cs.cmu.edu/projects/scene-completion/

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:
  - Pages with similar words
    - For duplicate detection, classification by topic
  - Customers who purchased similar products
    - Products with similar customer sets
  - Images with similar features
  - Users who visited similar websites

Problem for Today's Lecture

"Locality-Sensitive Hashing"（局部敏感哈希，LSH）

- Given: High dimensional data points $x_1, x_2, \ldots$
  - E.g.: Image is a long vector of pixel colors:

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

- $+$ distance function $d(x_1, x_2)$
- Goal: find all pairs of data points(x_i, x_j) that are $d(x_i, x_j) \leq s$
- Naïve solution would take $O(N^2)$ where N is the number of data points
- **MAGIC**: This can be done in $O(N)$!! How?

Main idea: candidates

- Pass 1: Take documents and hash them to buckets s.t. **documents that are similar hash to the same bucket**
- Pass 2: Only compare documents that are candidates (i.e. they hashed to the same bucket)
- Benefits: Instead of $O(N^2)$ comparisons, we need $O(N)$ comparisons!

Distance measure

- Goal: find near-neighbors in high-dim space
- We define "near neighbors" as points that are a "small distance" apart
- We need to define what "distance" means...

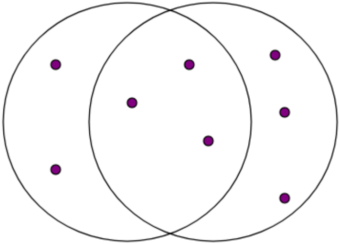
Jaccard distance/similarity

- **"Jaccard similarity"**: the relative size of their intersection
- The Jaccard similarity of sets S and T is

$$|S \cap T|/|S \cup T|$$

- Jaccard similarity of S and $T = SIM(S, T)$
- Jaccard distance: $DIST(S, T) = 1 - SIM(S, T)$

Jaccard distance/similarity



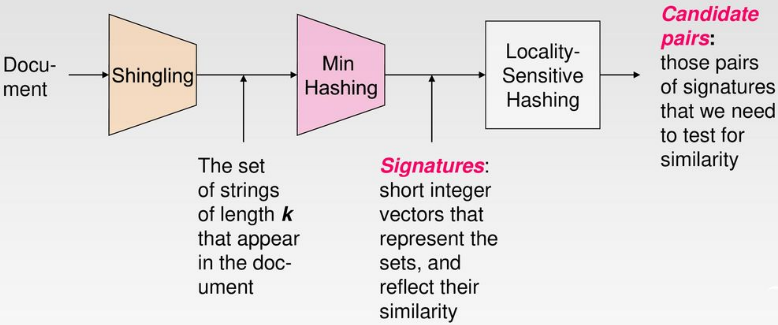
3 in intersection.
8 in union.
Jaccard similarity
= 3/8

3 Steps for Similar Docs

分片，最小哈希，局部敏感哈希

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

The Big Picture



Represent documents as 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!

Represent documents as sets: shingles

- Construct from the document the set of short strings
- $\Longrightarrow$ documents that share sentences or phrases will have many common elements
- One of the most common approach: **shingling**

Define: k-Shingles

- A k-shingle is sequence of k tokens that appears in the doc
  - Tokens can be characters, words...
  - E.g., tokens = characters
- Example:
  - Document D is the string $abcdabd$ and $k = 2$
  - the set of 2-shingles for D is $\{ab, bc, cd, da, bd\}$

k-Shingles: white space

- White space: blank, tab, newline, etc.
- Replace any sequence of one or more white-space characters by a single blank
- $\implies$ distinguish shingles that cover two or more words from those that do not:
  - E.g. """The plane was ready for touch down" and "The quarterback scored a touchdown"

k-Shingles: Choosing the Shingle Size

- We can pick k to be any constant
- If k too small: most sequences of k characters to appear in most documents
    - $k = 1 \implies$ all Web pages will have high similarity
- 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

k一般取值在5到10

Similarity Metric for Shingles

- 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
- A natural similarity measure is the Jaccard similarity
- **Working assumption:**

  documents that have lots of shingles in common have similar text

Signatures

- **Sets of shingles are large**
- It may not be possible to store all the shingle-sets in main memory
- The solution: replace large sets by much smaller representations called "signatures."
- The important property: we can compare the signatures of two sets and estimate the Jaccard similarity
- The signatures provide the estimates of the Jaccard similarity
- The larger the signatures, the more accurate the estimates

Outline: finding similar columns

- So far:
    - Documents $\rightarrow$ Sets of shingles
- Next goal: Find similar documents while computing small signatures
- Similarity of documents $=$ similarity of signatures

Min-hashing

- The hash function depends on the similarity metric:
  - Not all similarity metrics have a suitable hash function
- For the Jaccard similarity: **Min-Hashing**
- Suitable for problems of finding subsets that have significant intersection

Encoding Sets as Bits Vectors

- Encode sets using 0/1 (bit, boolean) vectors
  - One dimension per element in the universal set
- Example: $C_1 = \{1011\}, C_2 = \{1001\}$
  - Size of intersection $= 2$; size of union $= 3$
  - Jaccard similarity $= 2/3$

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
  - Column similarity is the Jaccard similarity of the corresponding sets (rows with value 1)
  - Typical matrix is sparse!

Min-hashing

https://blog.csdn.net/cacique111/article/details/127280105

- The rows of the boolean matrix permuted under random 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 to create a signature of a column

Min-Hashing Example

Note: Another (equivalent) wa
to store row indexes or raw shingles (e.g. mouse, lion):

1	5	
2	3	
6	4	

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

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 *M*

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

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

J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

Motivation Locality-Sensitive Hashing

- Suppose we need to find near-duplicate documents among $N = 1$ million documents
- We would have to compute pairwise Jaccard similarities for every pair of docs:
  - $N(N-1)/2 \approx 5 \times 10^{11}$ 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...

Candidates from Min-Hash

- Pick a similarity threshold s $(0 < s < 1)$
- Columns x and y of M are a candidate pair if their signatures agree on at least fraction s of their rows:

$$M(i,x) = M(i,y) \quad \text{for at least frac. } s \text{ values of } i$$

- We expect documents x and y to have the same (Jaccard) similarity as their signatures
- Check in main memory that candidate pairs really do have similar signatures

Summary: 3 steps

- **Shingling**: Convert documents to sets
- **Min-Hashing**: Convert large sets to short signatures, while preserving similarity
- **Locality-Sensitive Hashing**: Focus on pairs of signatures likely to be from similar documents

References

- J. Leskovec, A. Rajaraman and J. D. Ullman *Mining of Massive Datasets* (2014), Chapter 3
- A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions", Comm. ACM 51:1, pp. 117 - 122, 2008.