Finding similar objects

How would you do it?

- Finding very similar items might be computationally demanding task
- We can relax our requirement to finding **somewhat similar** items

Running example: comparing documents

- Documents have common text, but no common topic
- Easy special cases:
  - Identical documents
  - Fully contained documents (letter by letter)
- General case:
  - Many small pieces of one document appear out of order in another. What do we do then?

Finding similar documents

- Given a collection of documents, find pairs of documents that have lots of text in common
  - Identify mirror sites or web pages
  - Plagiarism
  - Similar news articles

Key steps

- Convert documents (news articles, emails, etc) to sets
- Convert large sets to **small signatures**, while preserving the similarity
- Compare the signatures instead of the actual documents

Data model: sets

- Data points are represented as sets (i.e., sets of shingles)

- Similar data points have large intersections in their sets

  - Think of documents and shingles
  - Customers and products
  - Users and movies

Similarity measures for sets

- Now we have a set representation of the data
- Jaccard coefficient
- **A, B** sets (subsets of some, large, universe **U**)

$$sim(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Find similar objects using the Jaccard similarity

- Naïve method?
  - Linear scan

- Problems with the naïve method?
  - There are too many objects
  - Each object consists of too many sets

Speeding up the naïve method

- Represent every object by a signature (summary of the object)
- Examine pairs of signatures rather than pairs of objects
- Find all similar pairs of signatures
- **Check point:** check that objects with similar signatures are actually similar

Still problems

- Comparing large number of signatures with each other may take too much time (although it takes less space)
- The method can produce pairs of objects that might not be similar (false positives). The check point needs to be enforced

Creating signatures

- For object **x**, signature of **x (sign(x))** is much smaller (in space) than **x**

- For objects **x, y** it should hold that **sim(x,y)** is almost the same as **sim(sing(x),sign(y))**

Intuition behind Jaccard similarity

- Consider two objects: **x,y**

	x	**y**
a	1	1
b	1	0
c	0	1
d	0	0

- **a**: # of rows of form same as **a**
- **sim(x,y)= a /(a+b+c)**

A type of signatures –minhashes

- Randomly **permute** the rows
- **h(x):** first row (in permuted data) in which column **x** has an **1**
- Use several (e.g., 100) independent hash functions to design a signature

	x	y
a	1	1
b	1	0
c	0	1
d	0	0

	x	y
a	0	1
b	0	0
c	1	1
d	1	0

“Surprising” property

- The probability (over all permutations of rows) that **h(x)=h(y)** is the same as **sim(x,y)**
- Both of them are **a/(a+b+c)**
- So?
  - **The similarity of signatures is the fraction of the hash functions on which they agree**

Minhash algorithm

- Pick **k** (e.g., 100) permutations of the rows
- Think of **sign(x)** as a new vector
- Let **sign(x)[i]**: in the **i**–th permutation, the index of the **first row that has 1** for object **x**

Example of minhash signatures

- Input matrix

	x1	x2	x3	X4
1	1	0	1	0
2	1	0	0	1
3	0	1	0	1
4	0	1	0	1
5	0	1	0	1
6	1	0	1	0
7	1	0	1	0

1
3
7
6
2
5
4

	x1	x2	x3	X4
1	1	0	1	0
3	0	1	0	1
7	1	0	1	0
6	1	0	1	0
2	1	0	0	1
5	0	1	0	1
4	0	1	0	1

1	2	1	2

Example of minhash signatures

- Input matrix

	x1	x2	x3	X4
1	1	0	1	0
2	1	0	0	1
3	0	1	0	1
4	0	1	0	1
5	0	1	0	1
6	1	0	1	0
7	1	0	1	0

4
2
1
3
6
7
5

	x1	x2	x3	X4
4	0	1	0	1
2	1	0	0	1
1	1	0	1	0
3	0	1	0	1
6	1	0	1	0
7	1	0	1	0
5	0	1	0	1
	2	1	3	1

Example of minhash signatures

- Input matrix

	x1	x2	x3	X4
1	1	0	1	0
2	1	0	0	1
3	0	1	0	1
4	0	1	0	1
5	0	1	0	1
6	1	0	1	0
7	1	0	1	0

3
4
7
6
1
2
5

	x1	x2	x3	X4
3	0	1	0	1
4	0	1	0	1
7	1	0	1	0
6	1	0	1	0
1	1	0	1	0
2	1	0	0	1
5	0	1	0	1

3	1	3	1

Example of minhash signatures

- Input matrix

	x1	x2	x3	X4
1	1	0	1	0
2	1	0	0	1
3	0	1	0	1
4	0	1	0	1
5	0	1	0	1
6	1	0	1	0
7	1	0	1	0

$\approx$

x1	x2	x3	X4
1	2	1	2
2	1	3	1
3	1	3	1

	actual	signs
(x1,x2)	0	0
(x1,x3)	0.75	2/3
(x1,x4)	1/7	0
(x2,x3)	0	0
(x2,x4)	0.75	1
(x3,x4)	0	0

Is it now feasible?

- Assume a billion rows
- Hard to pick a random permutation of 1…billion
- **Even representing a random permutation requires 1 billion entries!!!**
- How about accessing rows in permuted order?
- ☹

Being more practical

- Approximating row permutations: pick **k=100** (?) hash functions **$(h_1, \ldots, h_k)$**

for each row **r**

  for each column **c**

    if c has **1** in row **r**

      for each hash f

        if h_i (r) is a small

          M (i,c) = h_i (r);

M(i,c) will become the smallest value of **h_i(r)** for which column **c** has **1** in row **r**; i.e., **h_i (r)** gives order of rows for **i–th** permutation.

Example of minhash signatures

- Input matrix

	x1	x2
1	1	0
2	0	1
3	1	1
4	1	0
5	0	1

	x1	x2
1	0	1
2	2	0

h(r) = r + 1 mod 5
g(r) = 2r + 1 mod 5