Introduction to Data Science

GIRI NARASIMHAN, SCIS, FIU

Similarity

- Fundamental problem in Data Science
  - Web pages
  - Documents
  - Customer/User profiles (Collaborative Filtering)
  - Complaint histories
  - Disease profiles
  - Detecting Plagiarism

Jaccard Similarity

- Defined on 2 sets, S and T
  - SIM(S,T) = $|S \cap T| / |S \cup T|$
- E.g., Documents and Web pages can be thought of as set of words
- *Bag Similarity* uses **bags** instead of sets

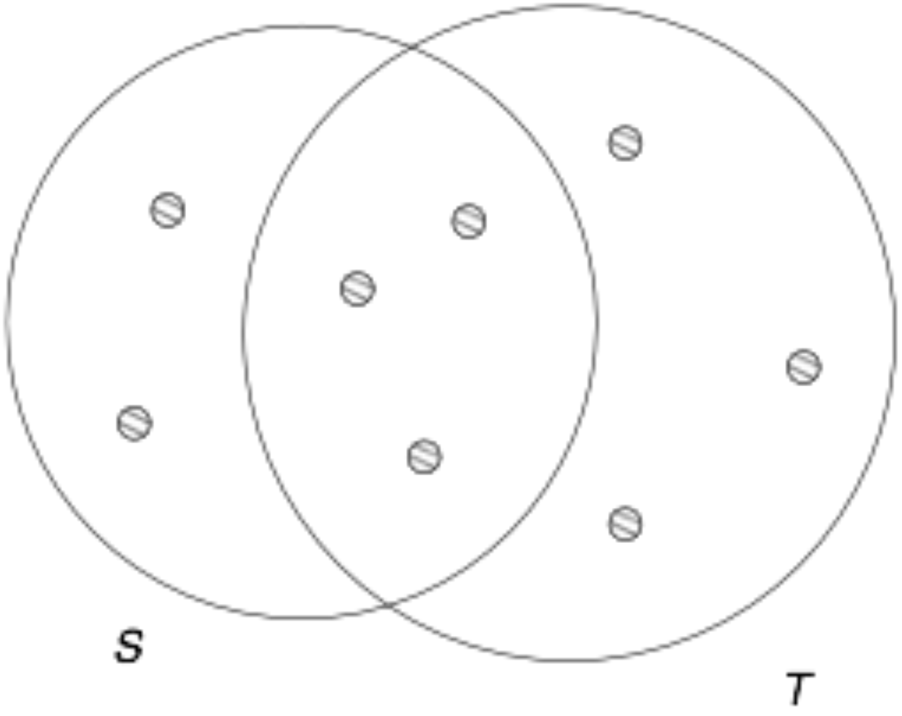


Figure 3.1: Two sets with Jaccard similarity 3/8

Applications of Jaccard Similarity

- Detecting Plagiarism
- Detecting Mirror pages
- Detecting same source articles – used by news aggregators

- **Collaborative filtering** – users recommended items liked by users with similar tastes
  - Online purchases
  - Movie ratings

Shingling of Documents

- **k-Shingles**
  - Any substring of a document of length k
  - Example: If document D is abcdabd then the set of 2-shingles = {ab, bc, cd, da, bd}
  - Since for large k, not all possible k-singles will be found, hashing is often used
- Compacted sets of shingles are called **signatures**
- Matrix Representations

Picking k for Shingling

- If k is too small, then almost all documents will be similar
- If k is too large, it can miss small common phrases
- Large k is needed for large docs
- For large k, hashing is used
- Emails: k = 5
- Larger documents: k = 9

Shingles from Words

- For news items, choose shingle as: **a stop word and next 2 words**

Shingles set size

- **Can be large** and can be roughly 4 times original document if each hash can be stored in 4 bytes.
- Need to replace large sets by **small signatures**
- Next we discuss how to construct small signatures

Characteristic Matrix

- To create small signatures, we imagine the **Characteristic Matrix**
- **Characteristic Matrix:** way to **visualize** a Set of sets and their Elements
  - **Rows** – Elements
  - **Columns** – Sets of elements
  - **Matrix** – 0/1 values
  - Matrix is assumed to be **sparse**

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

Small Signatures and MinHash

- Permute the rows
- Minhash(S_i) = row number of the first 1 in column S_i
- Minhash of the 4 columns are:
  - (a, c, b, a)
- Pr{Minhash(S_i) = Minhash(S_j)} equals
  - Jaccard similarity SIM(S_i, S_j)
- MinhashSignature(S_i) = result from N perm
  - Say N = 100

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

Computing Minhash Signatures

- **Permuting** a large characteristic matrix is too **expensive**
- **Simulate** permutations using **hashing**
  - It is a close **approximation**, except for collisions
  - Ignore **collisions**, which cause **errors** in the computation
  - **Sparsity** helps in lowering the errors
  - Instead of N permutations, we pick N hash functions
    - $h_1, h_2, \ldots, h_N$

Computing Minhash Signatures

- Given hash function $h_1, h_2, \ldots, h_N$, we want to compute MinHash values
- Let SIG(k,c) = signature matrix for k-th hash function and column c
- For row r, compute $h_1(r), h_2(r), \ldots, h_N(r)$
- If col c has 0 in row r, do nothing
- Else, for each k = 1, 2, …, N,
  - set $SIG(k,c) = \min\{SIG(k,c), h_k(r)\}$
- Initialize all SIG values to infty

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	∞	∞	∞	∞
h_2	∞	∞	∞	∞

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

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

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

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

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

Pair	True SIM	Approx SIM
(1,2)	0	0
(1,4)	2/3	1
(3,4)	1/5	1/2

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

Minhash Overview

- Takes very large documents and computes small signatures such that
  - Jaccard Similarity is retained
- **Example**: 1 M docs, N = 250 hash functions; 4 bytes per hash value
  - 1KB per doc signature
  - 1 GB to store all signatures
  - 0.5 Trillion pairs of docs
  - Similarity computation = 1 microsec
  - To compute all pairs = ~ 6 days (= 0.5184 trillion microsecs)

Find Closest Pair of Documents

- Cannot wait 6 days for an answer
- Clustering algorithms need this repeatedly
- **Approach**: Use a special hash function
  - Hash items so that similar items are likely to end up in the same bucket.
  - Avoid pairs in different buckets & reduce number of pairs to inspect
- These hash functions are called **Locality Sensitive Hashing** (**LSH**)
- Small Prob of error due to hashing
  - False Positives (cause extra work) and False Negatives (miss good pairs)

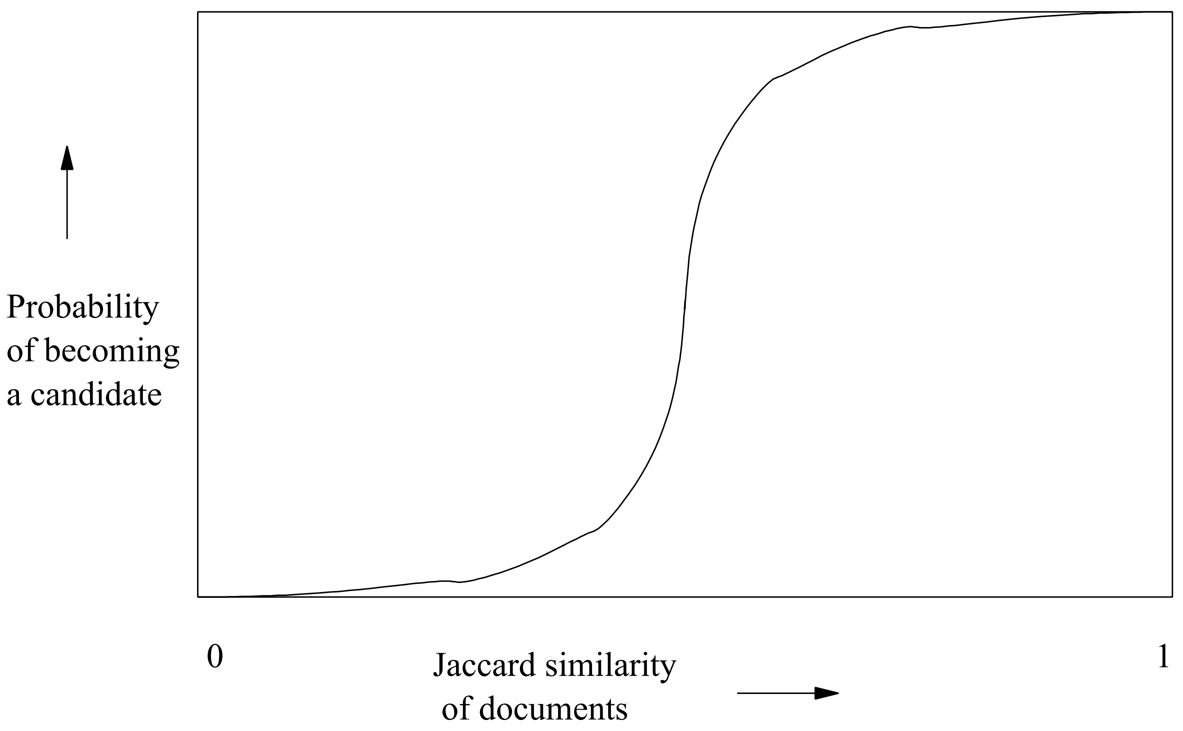
LSH for MinHash

- Divide signature matrix into b bands of r rows each
- For each band, hash column vector of r items to large # of buckets
- Use same hash function for each band but use separate buckets
  - Use different sets of buckets for different bands
- Any pair that appears in the same bucket in any band becomes a candidate for further inspection.All other pairs are discarded.
- If 2 columns are similar, then they must be identical in at least 1 band
- Each pair gets b chances to be in the same bucket

Analysis of LSH with Banding

- Assume b bands and r rows
- Consider a pair of docs with similarity value s
- Prob that their Minhash signatures agree in any particular row = s
- We want prob that this pair of docs becomes a candidate
- Prob signatures agree in all rows of one band = s^r
- Prob signature disagrees in at least one row of a band = $1 – s^r$
- Prob signatures disagree in at least one row in each band = $(1-s^r)^b$
- Prob that signatures agree in all rows of at least one band = $1 - (1-s^r)^b$

Behavior of $1 - (1-s^r)^b$



- **Independent** of **b** and **r**
  - Curve has to get from (0,0) to (1,1)
  - It's always an **S-curve**
- Threshold = value of s at steep rise
  - > threshold, pair is likely a candidate
  - Set (b,r) to achieve desired threshold

LSH-based Algorithm for Similar Items

- Pick k and construct k-shingles from each document
- Pick t, b, and r ($t \sim (1/b)^{1/r}$)
- Pick n = br hash functions
- Apply LSH technique, find candidates, check true similarity

Distance Measures

- A distance measure D must satisfy the following properties
  - **Non-negativity**: D(x,y) >= 0
    - D(x,y) = 0 if and only if x = y
  - **Symmetry**: D(x,y) = D(y,x)
  - **Triangle Inequality**: D(x,y) < = D(x,z) + D(z,y)

Important Distance Measures

- $D([x1, \ldots, xn], [y1, \ldots, yn]) = (|x1-y1|^r + \ldots + |xn-yn|^r)^{1/r}$
- If r= 2, this is the standard **Euclidean distance**
- Other values are commonly referred to as **Euclidean norms**
- **Jaccard Distance** = 1 – Jaccard Similarity
- **Cosine Distance** = **Dot Product** of 2 vectors
- **Edit Distance** = measure of changes to turn **x** into **y**
- **Hamming Distance** = # of components in which 2 vectors differ

Finding Identical Items

- LSH works for items with low similarity
- What if we only want to find identical items
  - Not good just to look at say first few characters
  - Not good to compare entire documents to check
  - Even if we hashed, we would need too many buckets
  - Idea: **Compute hash value based on random positions**

Finding near-identical items

- Advanced topic – please read from text.