Computational Tools for Data Science

Week 5 Lecture:

Similar Items

Based on MMDS Chapter 3

Applications of Finding Similar Items

- Similar documents (textual similarity)
  - Plagiarism
  - Mirror pages
  - News articles from the same source
- Recommendation Systems/Collaborative Filtering
  - Online purchases
  - Netflix recommendations
- Entity Resolution
- Matching Fingerprints

Challenges

- Many small pieces of a document can appear out of order in another
  - Addressed by ”shingling”
- Documents too large or too many they cannot fit in main memory
  - Addressed by using “signatures”
- Too many documents to compare all pairs
  - Addressed by “locality-sensitive hashing”

Prelude: Hash Functions

- A hash function takes data of arbitrary size to fixed size values
  - Mapping integers to their remainder modulo m
  - Mapping strings to 32 bit integers

- Hash values are used as indices in arrays, or keys in dictionaries, where the data is stored – known as ***hash tables***

- Example: strings of length 9, alphabet = {a,b,…,z,_}
  - $27^9 \approx 7.6 \times 10^{12} \approx 2^{43}$ possible strings (9 bytes each)
  - Hash to integer from 0 to 2^{32}-1 (4 bytes)

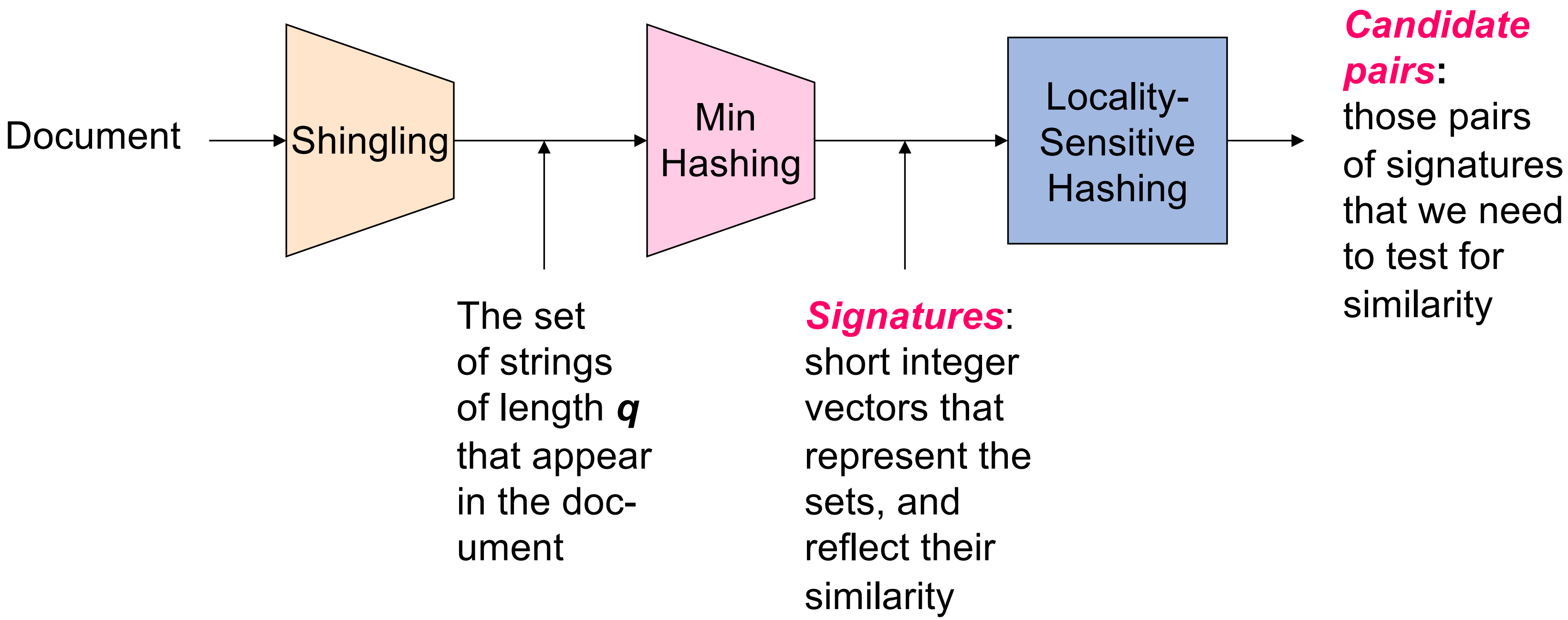
Hash Functions

- Want uniform coverage/few collisions

- Examples:
  - $h(x) = x\ mod\ 1000$
  - $h(x) = 133x + 27\ mod\ 1000$
  - $h(x) = 50x + 13$

- Many available online

Big Picture


Document
Shingling
Min Hashing
Locality-Sensitive Hashing
Candidate pairs: those pairs of signatures that we need to test for similarity
The set of strings of length *q* that appear in the document
Signatures: short integer vectors that represent the sets, and reflect their similarity

Converting documents to sets

- Simple approaches:
  - Document = set of words appearing in document
  - Document = set of “important” words appearing in document

- These don’t work well for this application. Why?

- Need to account for ordering of words!

- We use **shingles**

Shingling

- A q-shingle for a document is a sequence of q consecutive "**tokens**" appearing in the document
  - Tokens can be **characters**, **words**, or something else depending on the application
  - For now assume tokens = characters

- Example: $\boldsymbol{q = 2}$, document **D** = abcab
  - Set of 2-shingles: **S(D)** = {ab, bc, ca}
  - **Option:** shingles as multiset, count ab twice: S'(D) = {ab, bc, ca, ab}

Whitespace

- Often makes sense to replace any sequence of one or more whitespace characters by a single blank

- Helps to distinguish shingles that cover one or more words from those that do not

- Example:
  - “The plane was ready for touch down” vs “The quarterback scored a touchdown”
  - Both contain ‘touchdown’ as a 9-shingle if whitespace is ignored

Shingles and Similarity

- Documents that are intuitively similar will have many shingles in common
- Changing a word only affects q-shingles within distance q from that word
- Reordering paragraphs only affects the $2q$ shingles that cross paragraph boundaries
- Example $q = 3$, “The dog which chased the cat” versus “The dog that chased the cat”
  - Only 3-shingles replaced are g_w, _wh, whi, hic, ich, ch_, and h_c

Choosing the value of q

- Too small:
  - Most documents will have most q-shingles
  - High similarity of documents even if they have none of the same sentences or phrases
- Too big:
  - Storing the shingles takes more space
- q should be chosen so that the probability of any given shingle appearing in any given document is low.
  - Depends on how long the typical document is and how large the set of typical characters is

Choosing the value of q

- Emails:
  - $q = 5$ could be good
  - $27^5 = 14{,}348{,}907$ possible shingle
  - Most emails contain much fewer than 14 million characters

- More subtle than this
  - More than 27 characters
  - Appear with different probability – some 5-shingles may be common
  - Rule of thumb: imagine there are only 20 characters – 20^q possible shingles

- For large documents (e.g., research articles) $q = 9$ is considered safe

Compressing Shingles

- For large q we might expect that most q-shingles do not appear in any of our documents
- Compress long shingles (e.g., q=10) by hashing them to (say) 4 bytes.
- Represent a document by the set of hash values of its shingles (still refer to them as shingles)
- Documents will still only have a small fraction of possible (hashed) shingles
- Two documents could (rarely) appear to have shingles in common, when in fact only the hash-values were shared

Similarity of Sets

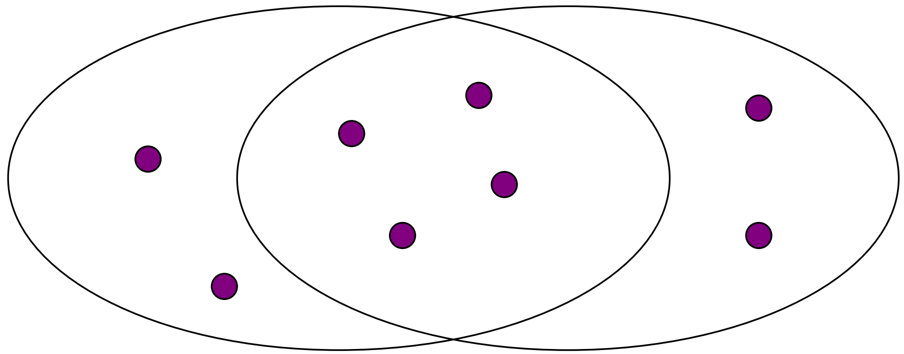
- The *Jaccard similarity* of two sets is the size of the intersection divided by the size of their union.

- $Sim(S_1, S_2) \ = \ |S_1 \cap S_2| / |S_1 \cup S_2|$

- $Sim(S_1, S_2) = 0$ if and only if the sets have no elements in common

- $Sim(S_1, S_2) = 1$ if and only if $S_1 = S_2$

Jaccard Similarity



4 in intersection
8 in union
Jaccard similarity = 4/8
= 1/2

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 q large enough, or most documents will have most shingles
  - $q = 5$ is OK for short documents (e.g., emails)
  - $q = 9, 10$ is better for long documents

Minhashing and Signatures of Sets

Signatures

- If we have very many very large documents, we may not be able to store all of the sets of shingles in main memory

- **Idea:** Hash each set to a small **signature** $h(S)$ such that:
  - $h(S)$ is small enough that the signature fits in main memory
  - $Sim(S_1, S_2)$ is the same as the "similarity" of the signatures $h(S_1)$ and $h(S_2)$

- **Goal:** First find a function h' such that
  - If $\boldsymbol{Sim(S_1, S_2)}$ is high, then with high probability $\boldsymbol{h'(S_1) = h'(S_2)}$
  - If $\boldsymbol{Sim(S_1, S_2)}$ is low, then with high probability $\boldsymbol{h'(S_1) \neq h'(S_2)}$

- Concatenate many such h' to obtain desired h

Signatures

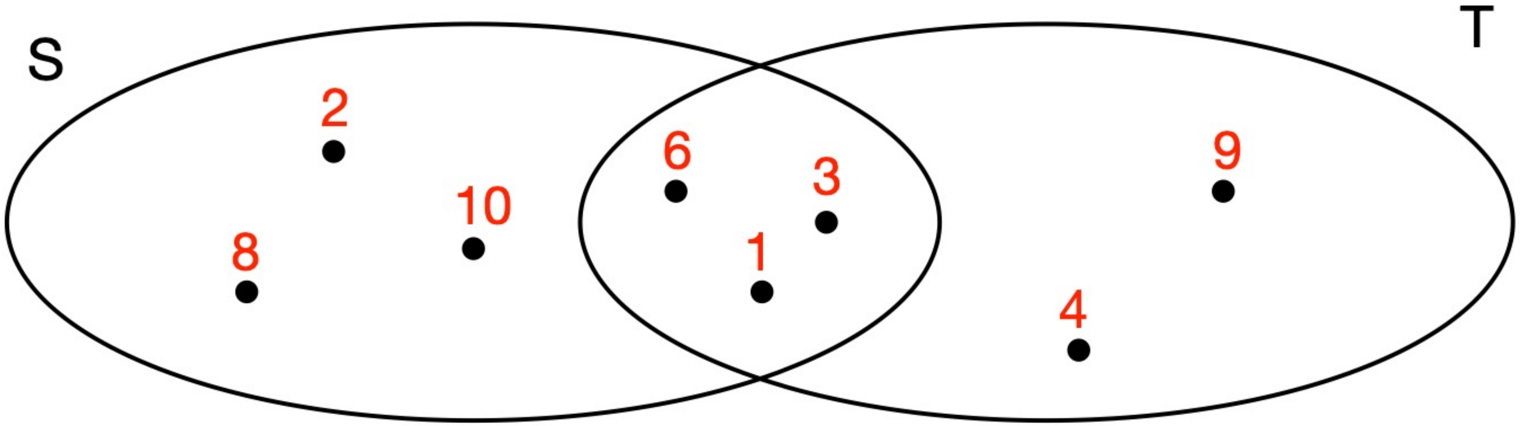
- **Solution:** Create **signatures** using "minhashing"

- Given a hash function h, the minhash of a set S with repsect to h, denoted $\hat{h}(S)$, is

$$\hat{h}(S) = \min\{h(s) : s \in S\}$$

- Use several (e.g., 100) independent hash functions to create signatures

Minhash and Jaccard Similarity



$$\Pr\left[\hat{h}(S) = \hat{h}(T)\right] = \frac{|S \cap T|}{|S \cup T|} = Sim(S,T)$$

In this case: $\hat{h}(S) = \hat{h}(T) = 1$

Signatures and Jaccard Similarity

- **Set signature**
  - Pick k hash functions $h_1, h_2, \dots, h_k$ independently
  - These give k minhashes $\hat{h}_1, \hat{h}_2, \dots, \hat{h}_k$
  - $\boldsymbol{sig(S)} = [\hat{h}_1(S), \hat{h}_2(S), \dots, \hat{h}_k(S)]$

- **Jaccard similarity estimation**
  - $Sim(S,T) \approx$ [# equal pairs in $sig(S)$ and $sig(T)$]$/k$

Computing Signatures for many Sets at once

- SIG – matrix with $SIG(i, S) \; = \; i^{th}$ entry of the signature of S
- Initialize $SIG(i, S) = \; \infty$ for all i and S
- Let $U \; =$ set of all elements in all sets S

for $s \in U$ do
    Compute $h_1(s), h_2(s), \dots, h_k(s)$
    for each set S do
        if $s \in S$ then
            for $i \in \{1, \dots, k\}$ do
                $SIG(i, S) \leftarrow \min\{h_i(s), SIG(i, S)\}$

Example

- $S = \{1,3,4\},\ T = \{2,3,5\}$
- $h_1(x) = x\ mod\ 5$
- $h_2(x) = (2x + 1)\ mod\ 5$

Locality-Sensitive Hashing

Locality-Sensitive Hashing

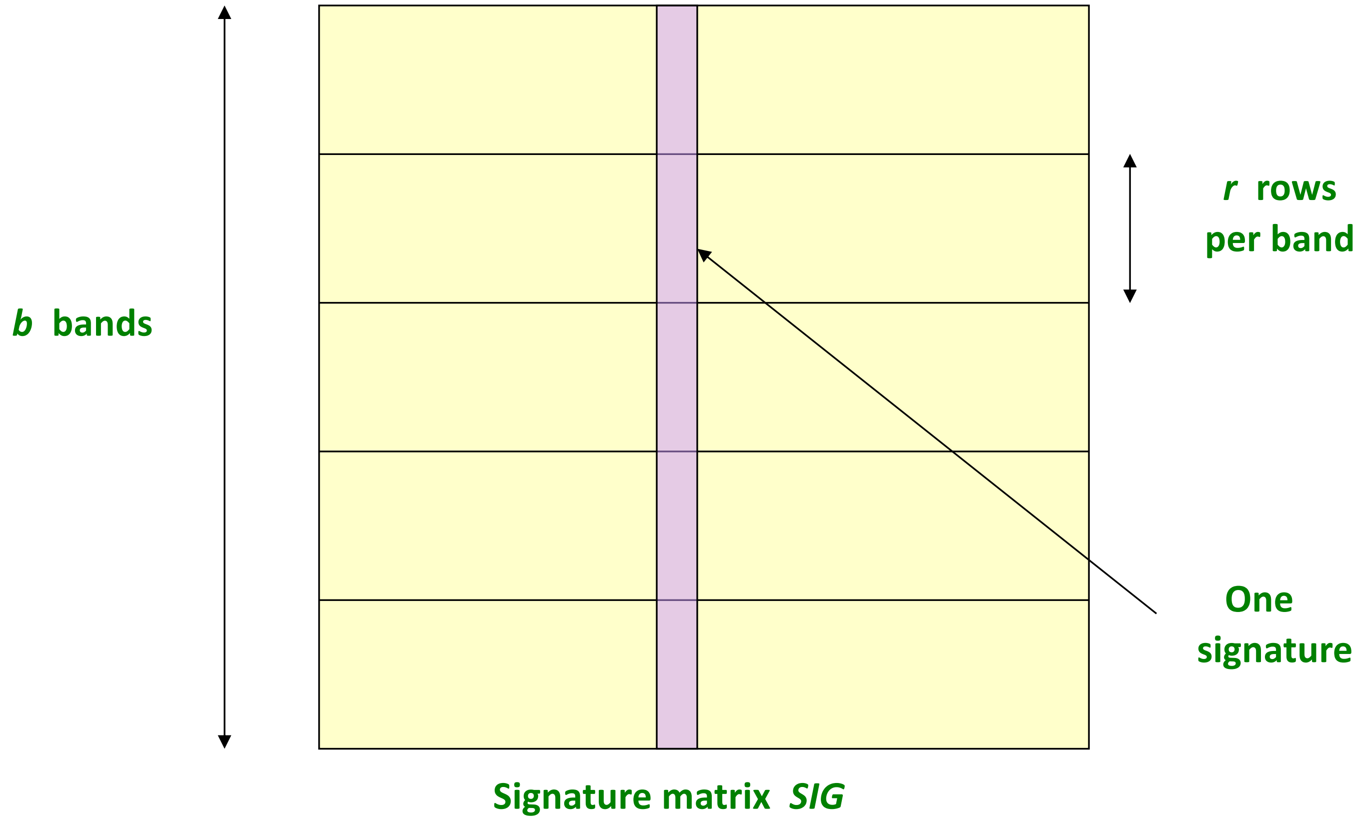
- **Goal:** Find documents with Jaccard similarity at least some threshold $0 < t < 1$
  - Balance false positives and false negatives
    - **false positives** = sets with similarity $< t$ that become candidates
    - **false negatives** = sets with similarity $> t$ that do not become candidates

- **Idea:**
  - Filter all but a few **candidate pairs**
  - Check candidates using set signature similarity estimation
    - Optional: compute exact Jaccard similarity for candidates

LSH for Minhash Signatures

- **Big idea:** has signatures several times
- Arange that (only) similar columns are likely to hash to the same value
- Candidate pairs are those that hash to the same value at least once

Partition Signature matrix into b Bands



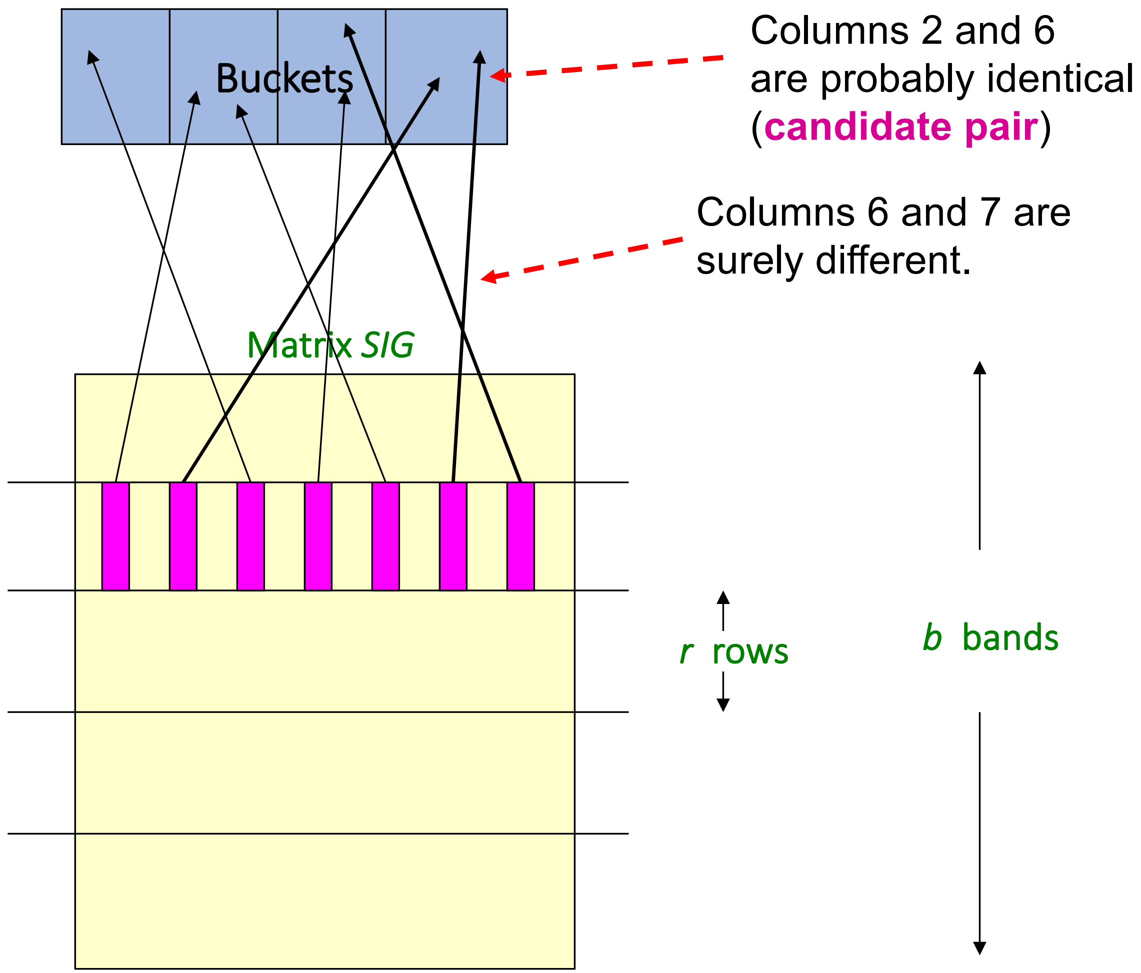
Partition SIG into Bands

- Divide SIG into b bands of r rows each

- For each band, hash its portion of each column to a hash table with m buckets
  - Make m as large as possible

- Candidate pairs are those that hash to the same bucket for ≥ 1 band
  - **Ideally:** this is equivalent to the signatures being equal on ≥ 1 band

- Tune b and r to catch most similar pairs, but few nonsimilar pairs


Buckets
Columns 2 and 6
are probably identical
(candidate pair)
Columns 6 and 7 are
surely different.
Matrix SIG
r rows
b bands

Simplifying Assumption

- There are **enough buckets** that columns are unlikely to hash to the same bucket unless they are **identical** in a particular band

- Hereafter, we assume that **“same bucket”** means **“identical in that band”**

- Assumption needed only to simplify analysis, not for correctness of algorithm

Example - Bands

- Suppose 100,000 documents/sets
- Signatures of 100 integers
- Similarity threshold: $t = 0.8$
- Approximately 5,000,000,000 pairs of signatures
- Choose 20 bands with 5 rows each

Suppose S_1 and S_2 are 80% similar

- Remember: 20 bands of 5 rows each

- Probability $sig(S_1)$ and $sig(S_2)$ are identical in one particular band: (0.8)^5 $\approx$ 0.328

- Probability $sig(S_1)$ and $sig(S_2)$ are ***not*** identical in any band: (1-0.328)^20 $\approx$ .00035
  - i.e., about 1/3000th of the 80%-similar underlying sets are **false negatives**

Suppose S_1 and S_2 are 40% similar

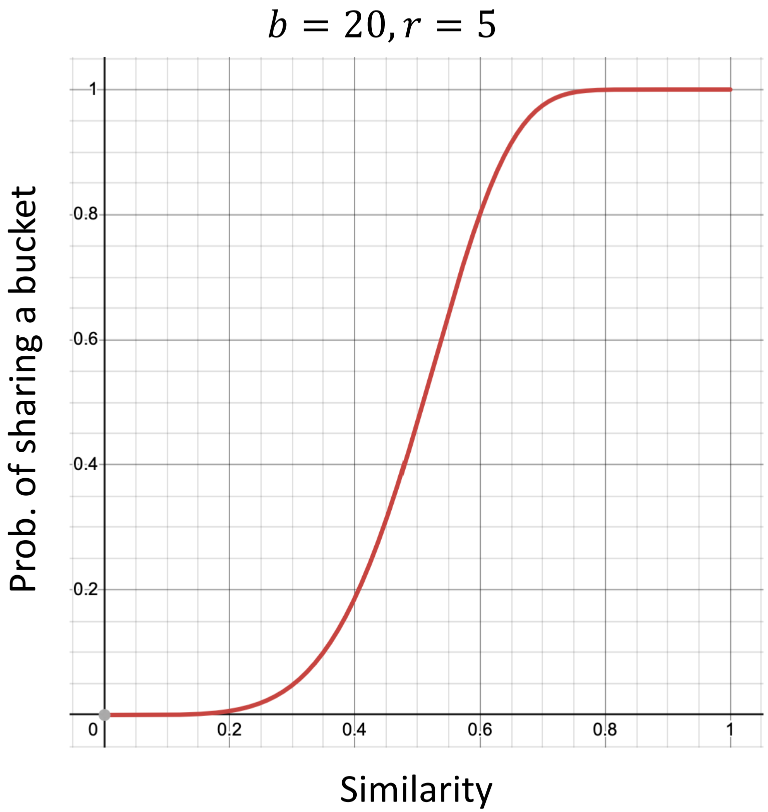
- Remember: 20 bands of 5 rows each

- Probability $sig(S_1)$ and $sig(S_2)$ are identical in one particular band: (0.4)^5 $\approx$ 0.01
- Probability $sig(S_1)$ and $sig(S_2)$ are identical in at least one band: 1 - (1-0.01)^20 $\approx$ .19
  - i.e., about 1/5th of the 40%-similar sets are **false positives**

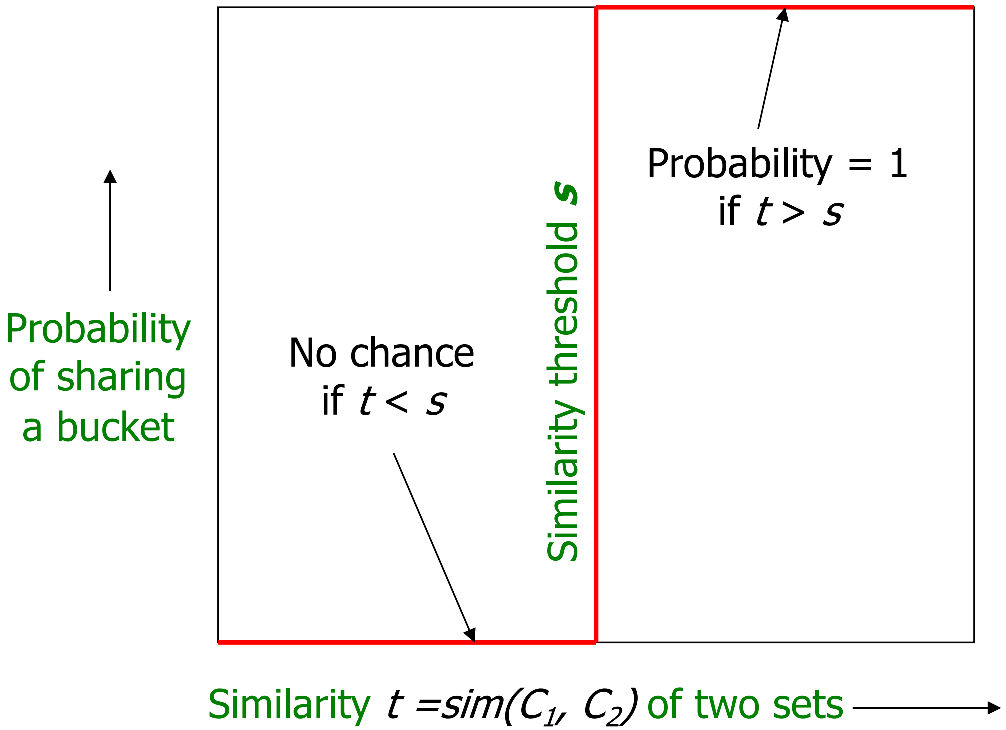
General Case

- b bands of r rows each
- S_1 and S_2 have similarity t
- Probability identical on a given band $= t^r$
- Probability not identical on a given band $= 1 - t^r$
- Probability no band identical $= (1 - t^r)^b$
- Probability at least one band is identical $= 1 - (1 - t^r)^b$

S-Curve

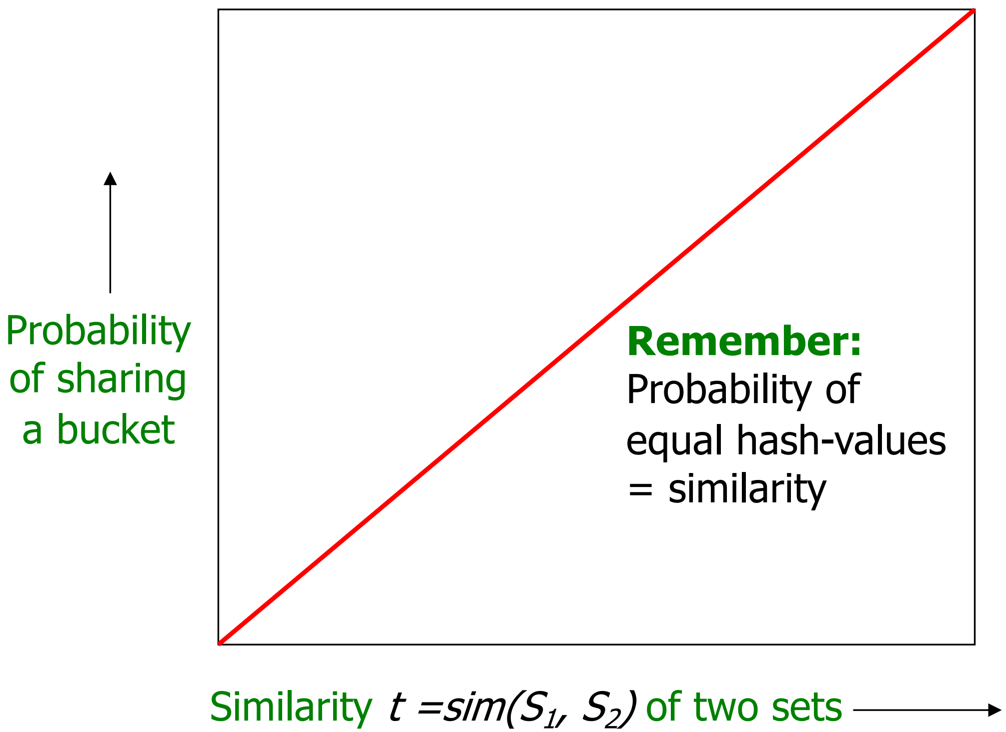


What We Want



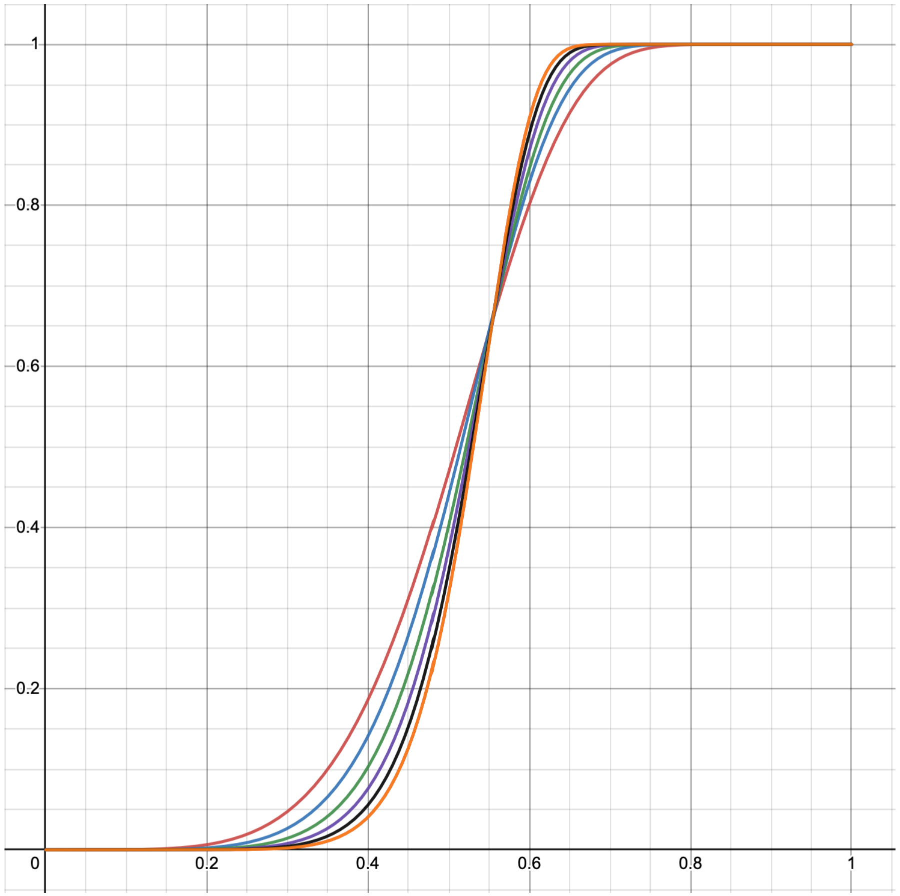
Probability of sharing a bucket
No chance if $t < s$
Similarity threshold s
Probability = 1 if $t > s$
Similarity $t = sim(C_1, C_2)$ of two sets

What 1 band of 1 row gives you



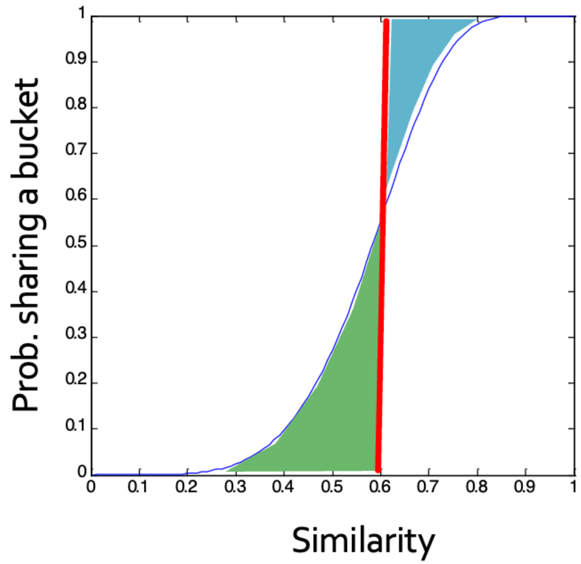
b bands of r rows each

$r = 5, b = 20$
$r = 6, b = 37$
$r = 7, b = 66$
$r = 8, b = 120$
$r = 9, b = 218$
$r = 10, b = 395$



Picking r and b

- **Picking *r* and *b* to get the best S-curve**
  - 50 hash-functions (r=5, b=10)



Blue area: False Negative rate
Green area: False Positive rate

Choose b and r so that $t \approx \left(\frac{1}{b}\right)^{1/r}$

Putting it all together

- Convert documents to their sets of shingles (must choose shingle size)
- Optional: Compress shingles via hashing
- Pick several (e.g., 100) hash functions and compute signatures of minhashes for each document/set
- Pick a similarity threshold $0 < t < 1$ and select b and r so that

$$t \approx \left(\frac{1}{b}\right)^{1/r}$$

- Use locality-sensitive hashing to find candidate pairs
- Check similarity of the signatures of the candidate pairs
  - This eliminates false positives
- Optional: Check actual Jaccard similarity of the sets/documents