Analysis of massive data sets

http://www.fer.hr/predmet/avsp

Prof. dr. sc. Siniša Srbljić

Doc. dr. sc. Dejan Škvorc

Doc. dr. sc. Ante Đerek

Faculty of Electrical Engineering and Computing
Consumer Computing Laboratory

Analysis of Massive Data Sets: Finding Similar Items (A)

Marin Šilić, PhD

Overview

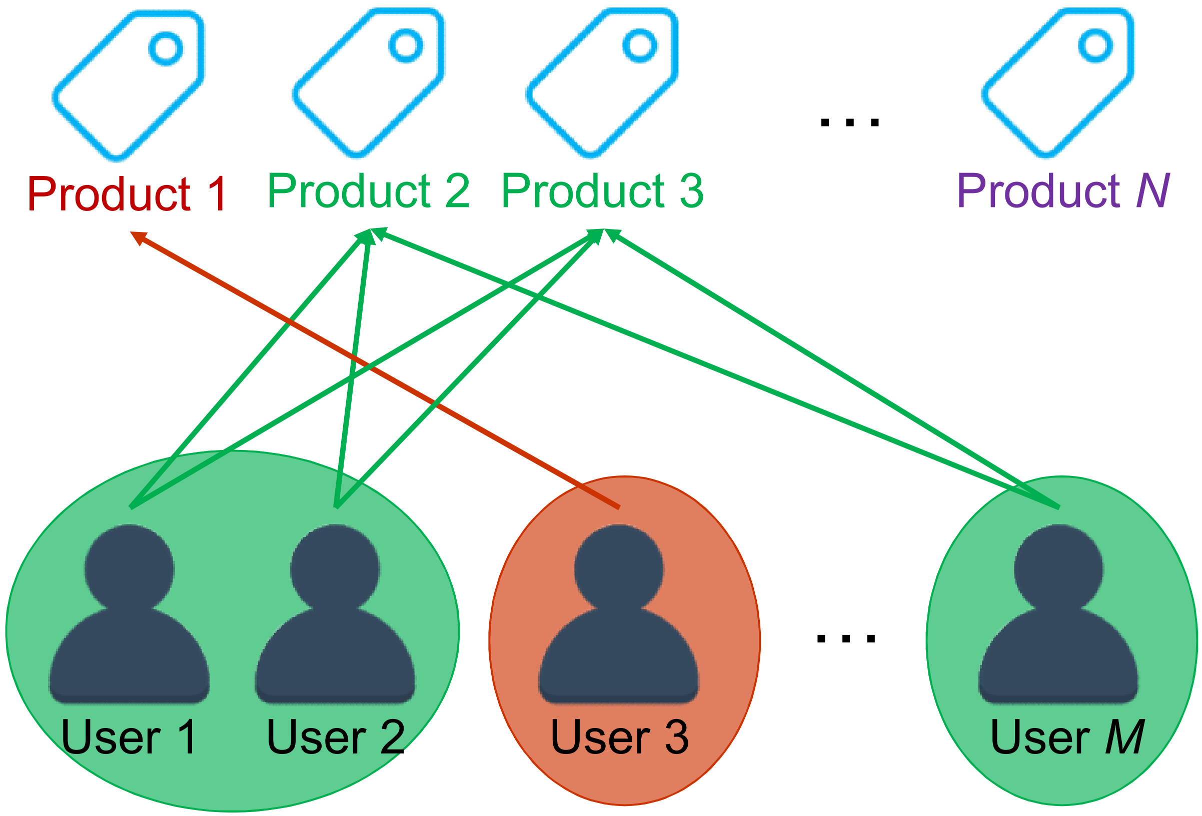
- Motivation
- Shingling
- MinHashing
- Locality Sensitive Hashing

Motivation

- **Fundamental Data-mining problem**
  - Examine data for similar items
  - Particular notion of similarity – similarity of sets
    - Find near-neighbors in “high-dimensional space”
  - Examples of finding similar sets
    - Collection of Products
    - Collection of Images
    - Collection of Web Pages

Motivation

- Collection of Products



Product 2 and Product 3 share same customer sets – similar products!

Product 1 and Product 3 don’t share same customer sets – dissimilar products!

Motivation

- **Collection of Images**
  - Image is represented as a long vector of pixel colors

$$image = \begin{bmatrix} p_{11} & \cdots & p_{1m} \\ \vdots & \ddots & \vdots \\ p_{n1} & \cdots & p_{nm} \end{bmatrix}$$

  - Define some distance function $d(i_1, i_2)$
    - Quantifies the distance between i_1 and i_2
  - Goal #1: Given some image i, find all k images that are within some distance threshold $d(i,k) \leq$ s
  - Goal #2: Find all pairs of images that are within some distance threshold $d(i_1, i_2) \leq$ s

Motivation

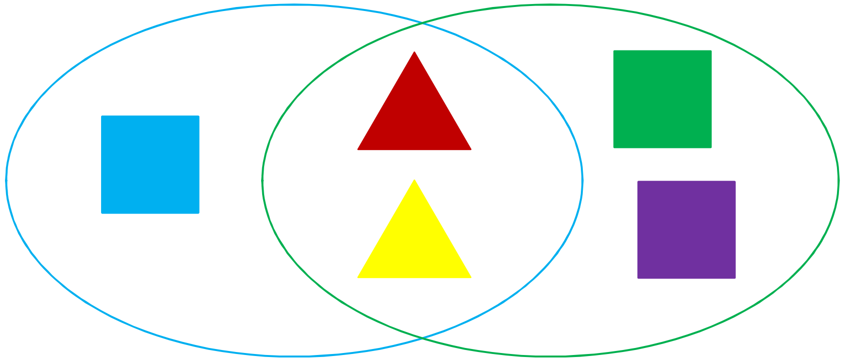
- **Collection of Web Pages**
  - Finding textually similar documents in a large corpus
    - Web Pages
    - News Articles
  - Testing whether two documents are identical is easy
    - Compare character-by-character
  - However, documents are not identical
    - Share large portion of text
  - Appliance
    - Plagiarism
    - Mirror Pages – don't show both in search results
    - Articles from the Same Source – cluster articles as same stories

Motivation

- **Distance Measure**
  - Goal: Find near-neighbors in a high-dimensional data
    - Near-neighbors are points that are small distance apart
    - Distance measure for each application should be defined
  - Jaccard distance/similarity
    - Used for quantifying similarity between high-dimensional data points which are represented as large sets
    - Jaccard similarity of two sets is the size of their intersection divided by the size of their union:
      $$sim(S_1, S_2) = |S_1 \cap S_2| / |S_1 \cup S_2|$$
    - Jaccard distance: $d(S_1, S_2) = 1 - sim(S_1, S_2)$

Motivation

- **Distance Measure**
  - Goal: Find near-neighbors in a high-dimensional data
    - Near-neighbors are points that are small distance apart
    - Distance measure for each application should be defined
  - Jaccard distance/similarity
    - Jaccard similarity: $sim(S_1, S_2) = |S_1 \cap S_2| / |S_1 \cup S_2|$
    - Jaccard distance: $d(S_1, S_2) = 1 - sim(S_1, S_2)$



$sim(S_1, S_2)$ = 2 / 5

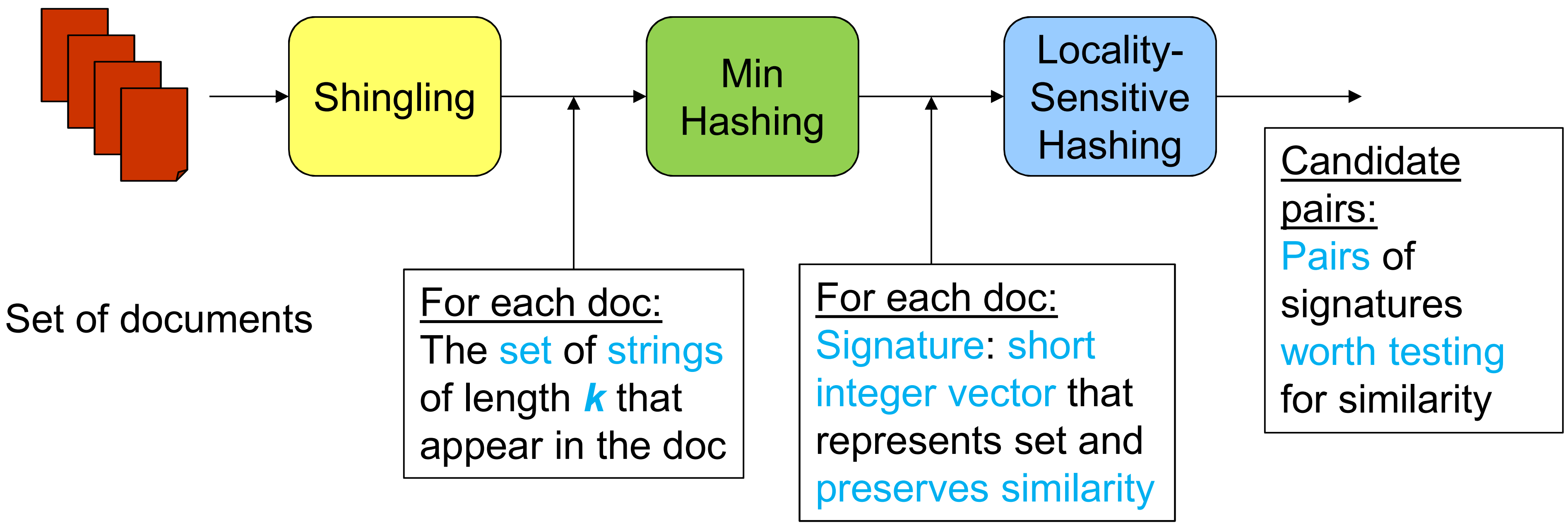
$d(S_1, S_2)$ = 3 / 5

Motivation

- **Challenges**
  - Challenge #1: How to convert the documents into sets?
    - Solution: Shingling
  - Challenge #2: Given the documents are represented as sets, how can we compare two documents in O(1)?
    - Solution: Min-Hashing
  - Challenge #3: Given the documents can be compared in O(1), how can we examine $N \sim 10^5$ documents for duplicates?
    - Solution: LSH

Motivation

- High-level Overview



Shingling

- **Challenge #1: Convert documents to sets**

  - Approaches
    - Document = set of words that appear in it
    - Document = set of "important" words that appear in it

  - We need to have a representation that incorporates the ordering of words
    - A different approach should be used – Shingles

Shingling

- **Shingle definition**
  - A *k*-shingle (or a *k*-gram) is a sequence of *k* tokens that appear in the doc
    - Tokens can be characters, words
    - It depends on the application
  - Example, $k = 3$, document $D = abeabe$
    - Set of 3-shingles $S(D) = \{abe, bea, eab\}$
    - Optionally, shingles can seen as bag (multiset), then we can count abe twice:
      - $S'(D) = \{abe, bea, eab, abe\}$

Shingling

- **Shingles Compression**
  - Some shingles may be long
  - Idea:
    - Hash singles to integer representation (4 bytes)
    - Represent each document by the set of hash values of its *k*-shingles
  - Example, $k = 3$, document $D = abeabe$
    - Set of 3-shingles $S(D) = \{abe, bea, eab\}$
    - Hash the shingles $h(D) = \{96356, 97406, 100166\}$

Shingling

- **Representation**
  - Each document is represented as a set of its *k*-shingles
    - Each shingle is a dimension
    - Vector of 0/1 representation
      - If the shingle is contained in the doc – 1, otherwise 0

D_1 = abeabe
D_2 = abe eab
D_3 = eab

$$\begin{array}{cc} & \begin{array}{ccc} D_1 & D_2 & D_3 \end{array} \\ \begin{array}{c} \text{abe} \\ \text{bea} \\ \text{eab} \end{array} & \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \end{array}$$

    - Similarity measure – Jaccard similarity:

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

$$sim(D_1, D_2) = 2/3, \; sim(D_1, D_3) = 1/3$$

Shingling

- **Fine tuning**
  - Documents that have a lot of similar text
    - Share a lot of shingles
    - Even if the text is in different order
  - The appropriate *k* should be chosen
    - It should be large enough, or large docs will have most shingles
    - A value $k = 5$ is fine for short docs
    - A value $k = 10$ is fine for long docs

Min-Hashing

- **Challenge #2: How can we compare two documents in O(1)?**
  - Encode sets using 0/1 vectors
  - Use Jaccard similarity/distance
  - Compute set intersection as bitwise AND, and set union as bitwise OR
  - Example: D_1 = 111, D_3 = 001
    - Intersection cardinality = 1
    - Union cardinality = 3
    - Jaccard similarity = 1/3
    - Jaccard distance = 1 – (Jaccard similarity) = 2/3

Min-Hashing

- **Representation**
  - Rows = shingles (set elements)
  - Columns = documents (sets)

	documents			
shingles	1	0	1	1
	0	0	1	1
	1	1	0	0
	0	0	1	1

  - Challenge #2 ~ Compute columns (docs) similarity in O(1)
    - If we have *k*-shingles, English alphabet , 26^k shingles
      - If *k = 5*, ~ 11M shingles
      - Obviously, we cannot compare two docs in O(1)!

Min-Hashing

- **Key Idea: Hashing Columns (Signatures)**
  - Each column that represents doc D hashes to a small signature $h(D)$ such that:
    - $h(D)$ is small enough to fit in RAM and so we can efficiently compare two signatures in constant time $O(1)$
    - $sim(D_1, D_2)$ is the same as the similarity of signatures $h(D_1)$ and $h(D_2)$
  - Goal: Find a hash function $h(\cdot)$ such that:
    - If $sim(D_1, D_2)$ is high, then with a high probability $h(D_1) = h(D_2)$
    - If $sim(D_1, D_2)$ is low, then with a high probability $h(D_1) \neq h(D_2)$

Min-Hashing

- **Goal: Find a hash function $h(\cdot)$ such that:**
  - If $sim(D_1, D_2)$ is high, then $h(D_1) = h(D_2)$
  - If $sim(D_1, D_2)$ is low, then $h(D_1) \neq h(D_2)$
- **The hash function depends on the similarity measure**
  - It is very hard to find the appropriate hash function for some similarity measure
- **However, there is a hash function for the Jaccard similarity**
  - Min-Hashing

Min-Hashing

- **Min-Hashing**
  - Create a random permutation $\boldsymbol{\pi}$ of 0/1 matrix rows
  - Define a hash function $h_\pi(D)$
    - The index of the first (in the permuted order $\boldsymbol{\pi}$) row in which column D has a value 1

$$h_\pi(D) = \min_\pi \pi(D)$$

  - To create a signature (hash) of the document
    - Generate a number (for instance 100) of independent permutations
    - For each permutation, compute the hash function and represent each document with the obtained signature

Min-Hashing

- **Min-Hashing Example**

Permutation		
0	1	3
3	5	2
4	3	0
1	2	5
2	4	4
5	0	1

	Documents			
Shingles	1	0	1	1
	0	0	1	1
	1	1	0	0
	0	0	1	1
	0	0	1	0
	1	1	0	0

Signatures			
0	4	0	0
0	0	1	1
0	0	2	2

Min-Hashing

□ Min-Hashing Example

Permutation		
0	1	3
3	5	2
4	3	0
1	2	5
2	4	4
5	0	1

	Documents			
Shingles	1	0	1	1
	0	0	1	1
	1	1	0	0
	0	0	1	1
	0	0	1	0
	1	1	0	0

Signatures			
0	4	0	0
0	0	1	1
0	0	2	2

i, j	**1, 2**	**3, 4**	**2, 4**
$Jacc\_sim(D_i, D_j)$	0.67	0.75	0.0
$Jacc\_sim(S_i, S_j)$	0.67	1.0	0.0

Min-Hashing

- **Claim:** $p[h_\pi(D_1) = h_\pi(D_2)] = Jaccard\_Sim(D_1, D_2)$
- **Proof: There are 3 row types:**
  - [1, 1] *x* type, suppose we have *X* rows of type *x*
  - [1, 0] and [0, 1] *y* type, suppose we have *Y* rows of this type
  - [0, 0] *z* type, suppose we have *Z* rows of *z* type
- **What is $Jaccard\_Sim(D_1, D_2)$ equal to?**
  - It is $X/(X + Y)$

Min-Hashing

- **Claim:** $p[h_\pi(D_1) = h_\pi(D_2)] = Jaccard\_Sim(D_1, D_2)$
- **Proof: There are 3 row types:**
  - [1, 1] *x* type, suppose we have *X* rows of type *x*
  - [1, 0] and [0, 1] *y* type, suppose we have *Y* rows of this type
  - [0, 0] *z* type, suppose we have *Z* rows of *z* type
- **On the other hand, what is $p[h_\pi(D_1) = h_\pi(D_2)]$ equal to?**
  - We rearrange rows of D_1 and D_2 using π
  - Then, we search the first row that contains a 1 in either doc
  - What is the probability that we encounter *x* row type?
  - It is $X/(X + Y)$

Min-Hashing

- **Challenge #2: How can we compare two documents in O(1)?**
  - We can pick k = 100 random permutations of the rows
    - Define $sig[D][i]$ as the index of the 1st 1-row according to the *i*-th permutation in the doc D
    - $sig[D][i] = \min(\pi_i(D))$
  - We can see that each document D is represented with its signature S, where the length of $|S| = \mathrm{k} = 100$
    - As can be seen, our goal is reached
    - We can compare two documents in constant time O(1)

Min-Hashing

Implementation trick

- Generating random permutations π_i is prohibitively costly
- Instead, we can use random hash functions to generate permutations
  - We pick $k = 100$ hash functions h_i
  - Then, for each row r, we compute $h_i(r)$
  - We use values $h_i(r)$ as random row permutation
- How to pick a random hash function?
  - Universal hashing:
  - $h_{a,b}(r) = \left((ax + b \; mod \; p) \; mod \; |R|\right)$
  - a and b are random integers
  - p is a prime number ($p > |R|$)
  - R is the set of *k*-shingles

Row	$r + 1 \; mod \; 5$	$3r + 1 \; mod \; 5$
0	1	1
1	2	4
2	3	2
3	4	0
4	0	3

LSH

- **Challenge #3: Given the documents can be compared in O(1), how can we examine N ~ 10^5 documents for duplicates?**
  - Naïve solution:
    - Compare each pair of signatures and determine if they are duplicates
    - It takes $O(N^2)$
      - Assume we can have 10^6 comparisons per second, it would take ~ 84 hours
      - If we have N = 10^6, it would take more than a year
  - Feasible solution:
    - Focus on pairs of signatures that are likely to be from similar docs

LSH

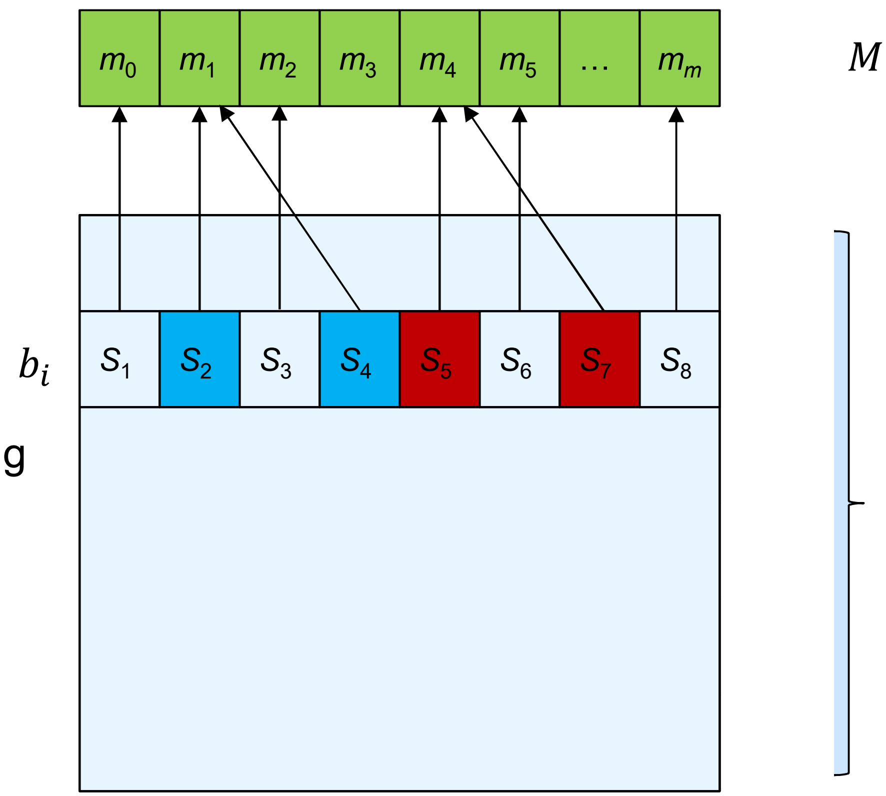
- **Goal: Find docs whose Jaccard similarity is at least s**
  - For instance at least $s = 0.75$
- **LSH – Locality Sensitive Hashing**
  - Idea: Use a special function f that tells whether D_1 and D_2 is a candidate pair
- **For Min-Hash matrices**
  - Hash columns of signature matrix to many buckets
  - Each pair of documents that hashes into the same bucket is a candidate pair

LSH

- **LSH for Min-Hashing**
  - Two similar docs have signatures that agree on at least fraction s of their rows
  - Signatures partially agree
    - The length of the signature is k (number of hash functions used in Min-Hash)
    - Some parts of the signature are identical, some parts differ
    - The idea is to split the signature into b equal parts – bands
    - For each band b, hash its portion of signature to a hash table with M buckets
    - Candidate pairs are signatures that hash to the same bucket in at least 1 band
      - By hash to the same bucket, we mean "identical in that band"

LSH

Hashing Bands



Signature Matrix

k rows representing hash functions used in Min-Hash

Each column is a doc signature

Split the signatures into b equal bands, each band contains $r = k/b$ rows

According to the band b_i candidates for similarity are: (S_2, S_4) and (S_5, S_7)

LSH

- **Example of Bands**
  - Suppose we have $N = 10^5$ docs
    - We use $k = 100$ hash functions – therefore 100 rows in signature matrix M
    - We use $b = 20$, therefore $r = k/b = 5$
  - Goal: find pairs with at least $s = 0.75$ similarity
    - We assume $sim(S_1, S_2) \geq 0.75$
    - S_1 and S_2 to hash to the same bucket in at least one band – we want them to be identical in at least one band
    - $P[S_1 = S_2 \ in \ exactly \ one \ band] = s^r = 0.75^5 = 0.2373$
    - $P[S_1 \neq S_2 \ in \ all \ bands] = (1 - s^r)^b = (1 - 0.75)^{20} = 0.0044$
      - 0.4% of 75% similar docs are false negatives (missed)
      - We would find 99.56% pairs of 75% similar docs

LSH

- **Example of Bands**
  - Suppose we have $N = 10^5$ docs
    - We use $k = 100$ hash functions – therefore 100 rows in signature matrix M
    - We use $b = 20$, therefore $r = k/b = 5$
  - Goal: find pairs with at least $s = 0.75$ similarity
    - We assume $sim(S_1, S_2) = 0.2$
    - S_1 and S_2 are bellow our threshold, so we want them not to hash to common buckets (different in all bands)
    - $P[S_1 = S_2 \ in\ one\ band] = sim^r = 0.2^5 = 0.00032$
    - $P[S_1 \neq S_2 \ in\ all\ bands] = (1 - sim^r)^b = 0.9936$
    - $P[S_1 = S_2 \ in\ at\ least\ one\ band] = 1 - (1 - sim^r)^b = 0.0064$
      - 0.6% of 20% similar docs are candidate pairs – false positives

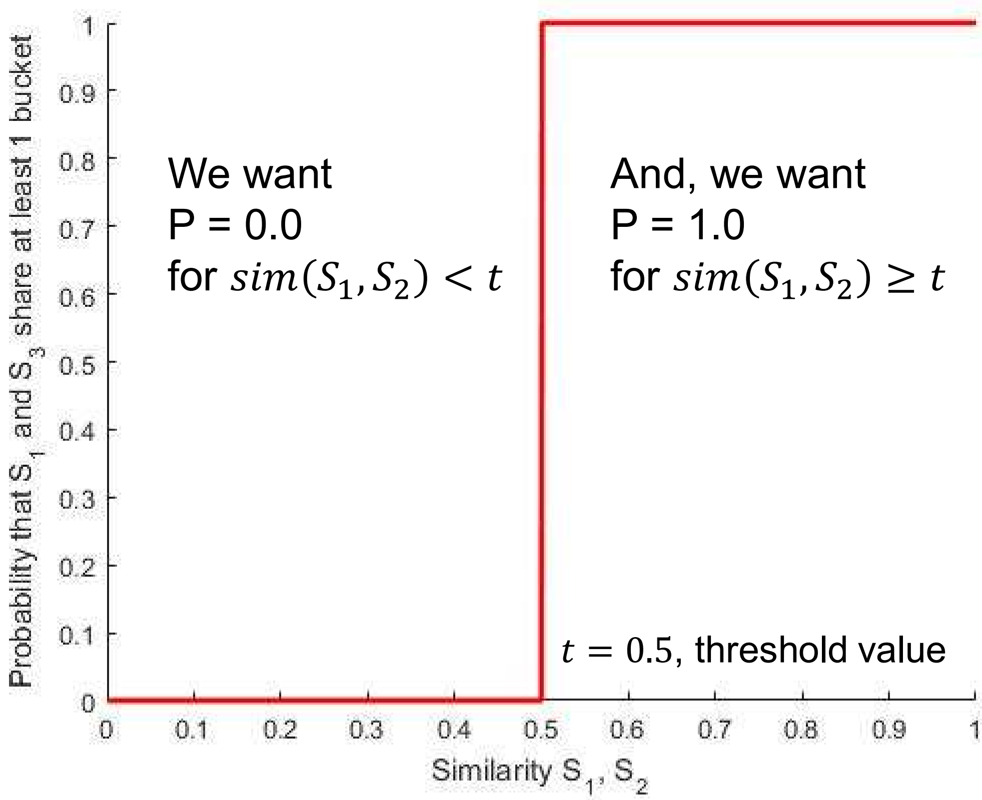
LSH

- **LSH trade-off**
  - Balance between false negatives and false positives
    - Choose appropriate values for:
      - k, the number of hash functions used in Min-Hash
      - b, the number of bands used in LSH
      - r, the number of rows per band
    - For instance, in previous example
      - If we had $b = 10$, and $r = 10$, what would have happened?
      - The number of false positives would go down
      - The number of false negatives would go up

LSH

- **LSH Ideally**

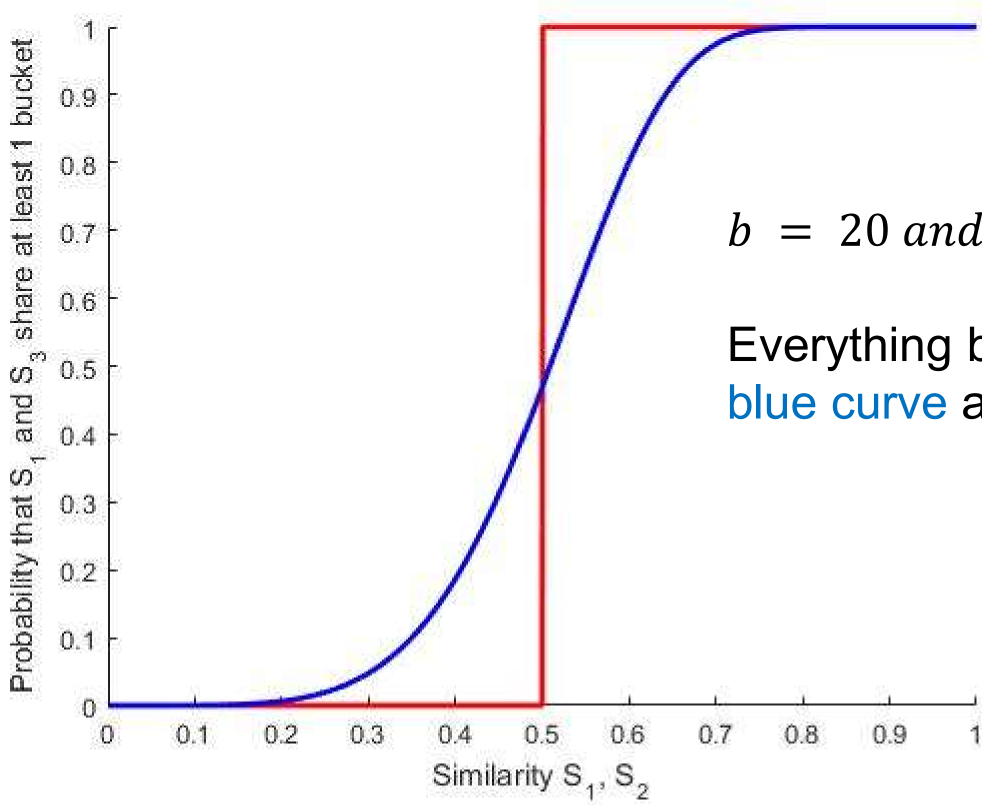
  $P[S_1\ and\ S_2\ share\ at\ least\ 1\ bucket] = 1 - (1 - sim^r)^b$



LSH

- **LSH in Practice**

$$P[S_1 \; and \; S_2 \; share \; at \; least \; 1 \; bucket] = 1 - (1 - sim^r)^b$$



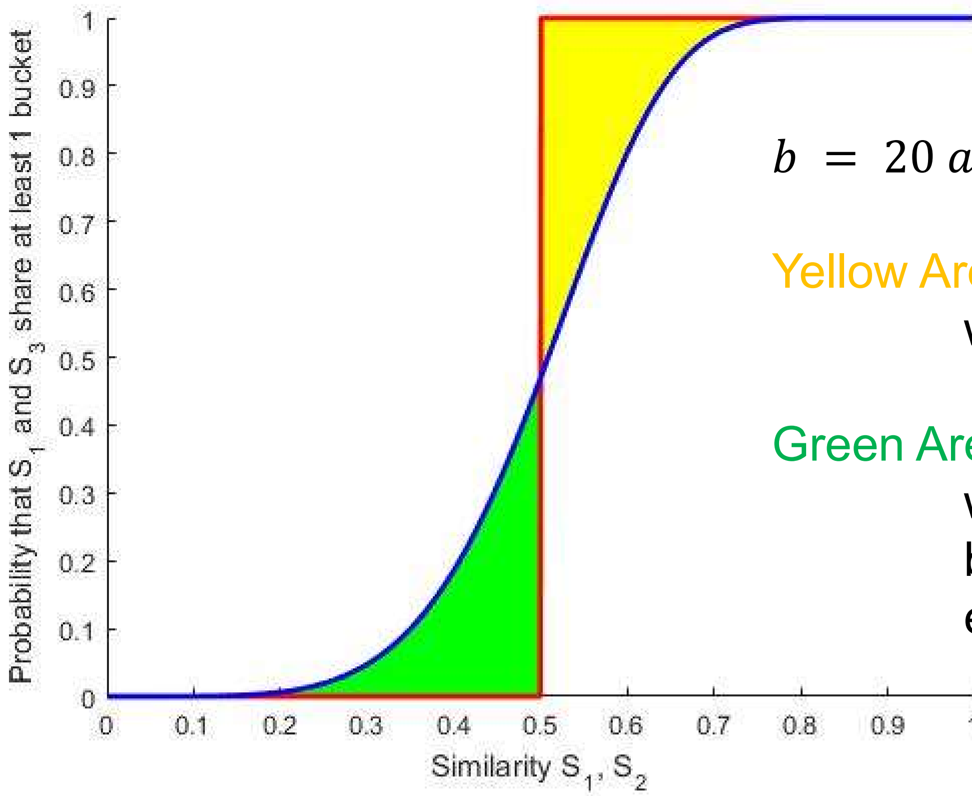
$b = 20 \; and, r = 5$

Everything bellow the blue curve are candidates

LSH

- **LSH in Practice**

$$P[S_1\ and\ S_2\ share\ at\ least\ 1\ bucket] = 1 - (1 - sim^r)^b$$



$b = 20\ and, r = 5$

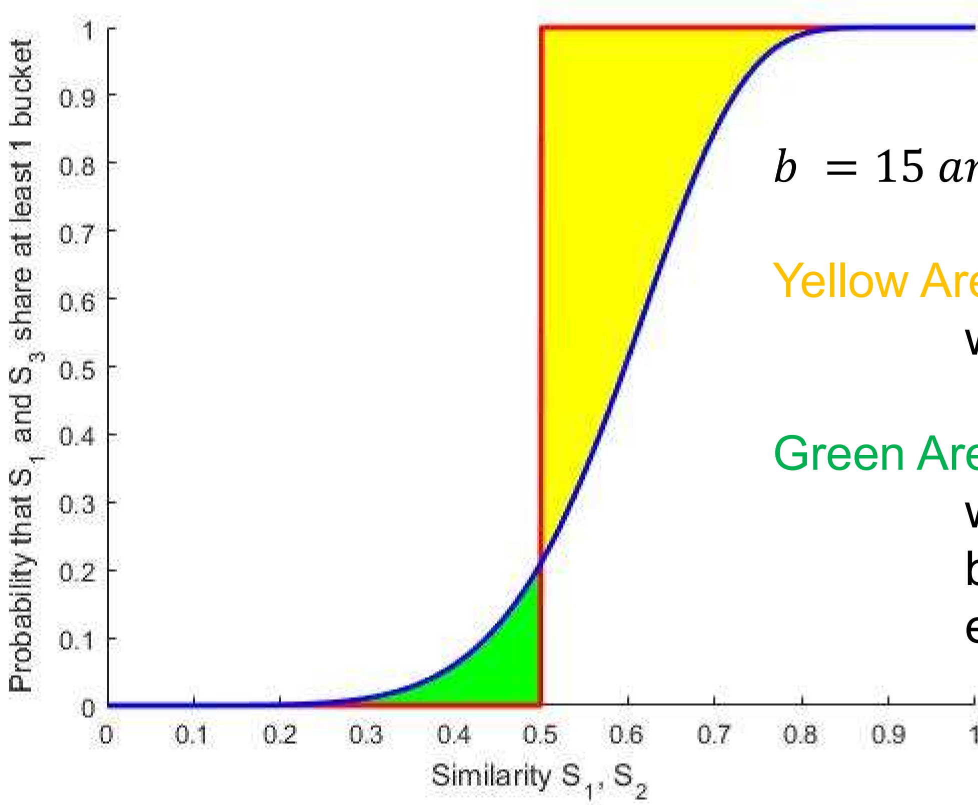
Yellow Area = False Negatives (FN)
we missed them!

Green Area = False Positives (FP)
we can eliminate them
by checking docs similarity
explicitly

LSH

- **LSH in Practice**

$$P[S_1\ and\ S_2\ share\ at\ least\ 1\ bucket] = 1 - (1 - sim^r)^b$$



$b = 15\ and, r = 5$

Yellow Area = False Negatives (FN)
we missed them!

Green Area = False Positives (FP)
we can eliminate them
by checking docs similarity
explicitly

LSH

- **Summary**
  - Choose k, b, r to get the most of pairs with similar signatures, but eliminate pairs that do not have similar signatures
  - Check candidate pairs whether they really have similar signatures
  - Optional:
    - In another pass check that docs with similar signatures are really similar documents