DSA5101: Finding Similar Items

LX Zhang

Department of Mathematics

National University of Singapore

Motivation

- Customers who purchased similar products
  -- Online shopping
  -- Amazon
- Identify nearly duplicated webpages
  -- Plagiarisms
  -- Mirrors that have almost the same content but differ in information about the host
- Images with similar patterns
- How to model similarity of two webpages/text document?
- How to deal with big data?
  -- Extract all similar pairs of objects from a large collection
  -- (online version) Is this object similar to something seen before?

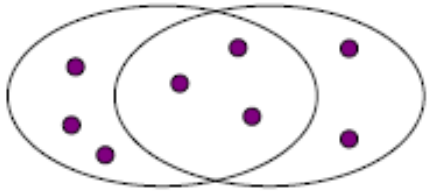
Similarity Models

- **Jaccard Similarity and Distance**

  -- Given two sets of abstract elements S_1, S_2;

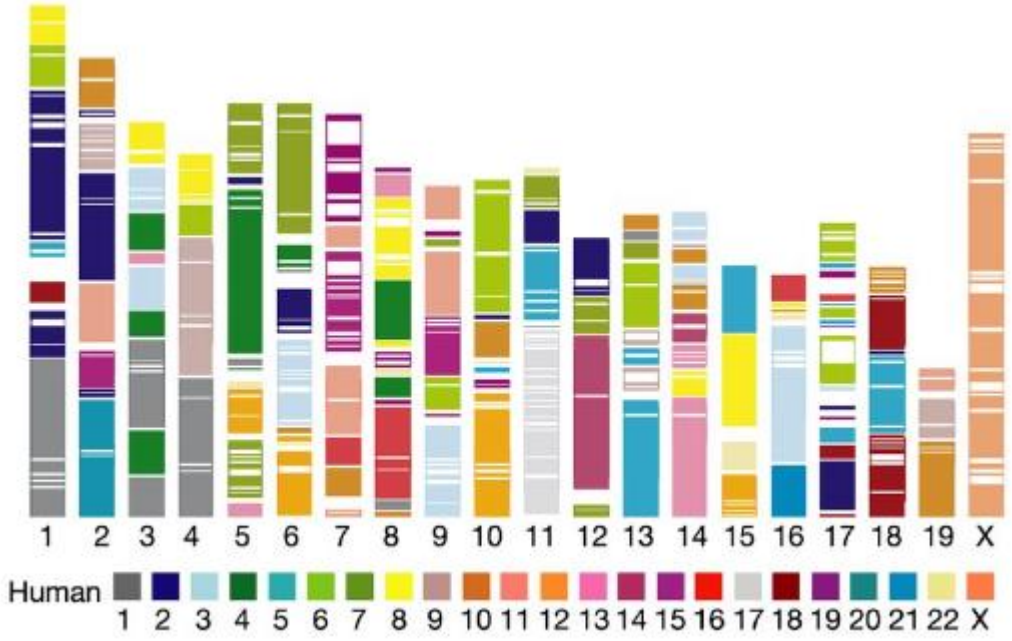
  $J(S_1, S_2) = |S_1 \cap S_2| / |S_1 \cup S_2|$.

  $D(S_1, S_2) = 1 - \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|} = \frac{|S_1 \Delta S_2|}{|S_1 \cup S_2|}$

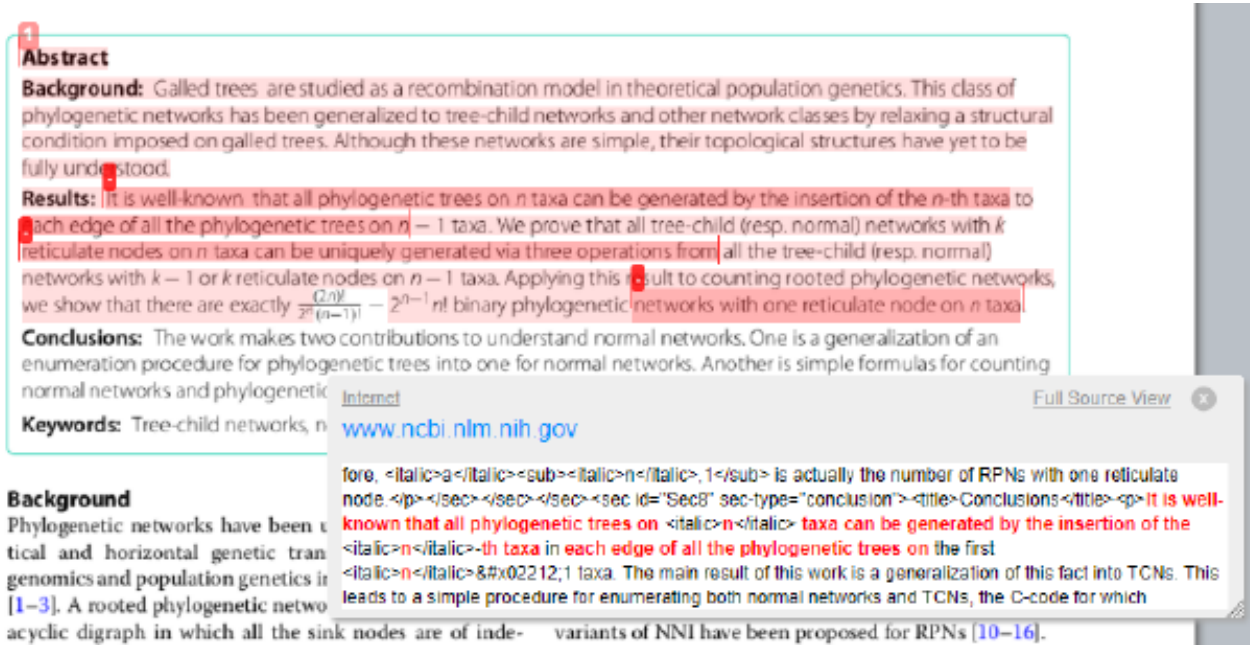


3 in intersection
8 in union
Jaccard similarity = 3/8
Jaccard distance = 5/8

- **Document Similarity and Distance**
  -- DNA sequences
  -- Text documents
  -- Webpages



iTehnticate



Part 1: Similarity Models

- **Document Similarity and Distance**
  -- Shingling: Convert documents to the sets of words
  -- Document (D) = A string (or sequence) of letters
  -- A k-shingle (or k-mer) for D is any substring of length k in D
  -- Thus, D is mapped to the set of *k*-shingles appearing in D.
  -- Similarity/distance for two documents can be defined as the Jaccard similarity/distance of the sets of *k*-shingles.

- **Example**
  D = abcdabd
  -- The set of 2-shingles for D is {ab, bc, cd, da, bd}.
  -- The set of 4-shingle for D is {abcd, bcda, cdab, dabd}

- **Edit Distance**
  -- Document = A string of letters
  -- The distance between two strings x and y is the smallest number of insertions/deletions and substitutions of single characters that will convert x to y.
  -- It is a generalization of Hamming distance
  -- Used widely in DNA sequences

kitten → sitten → sittin → sitting

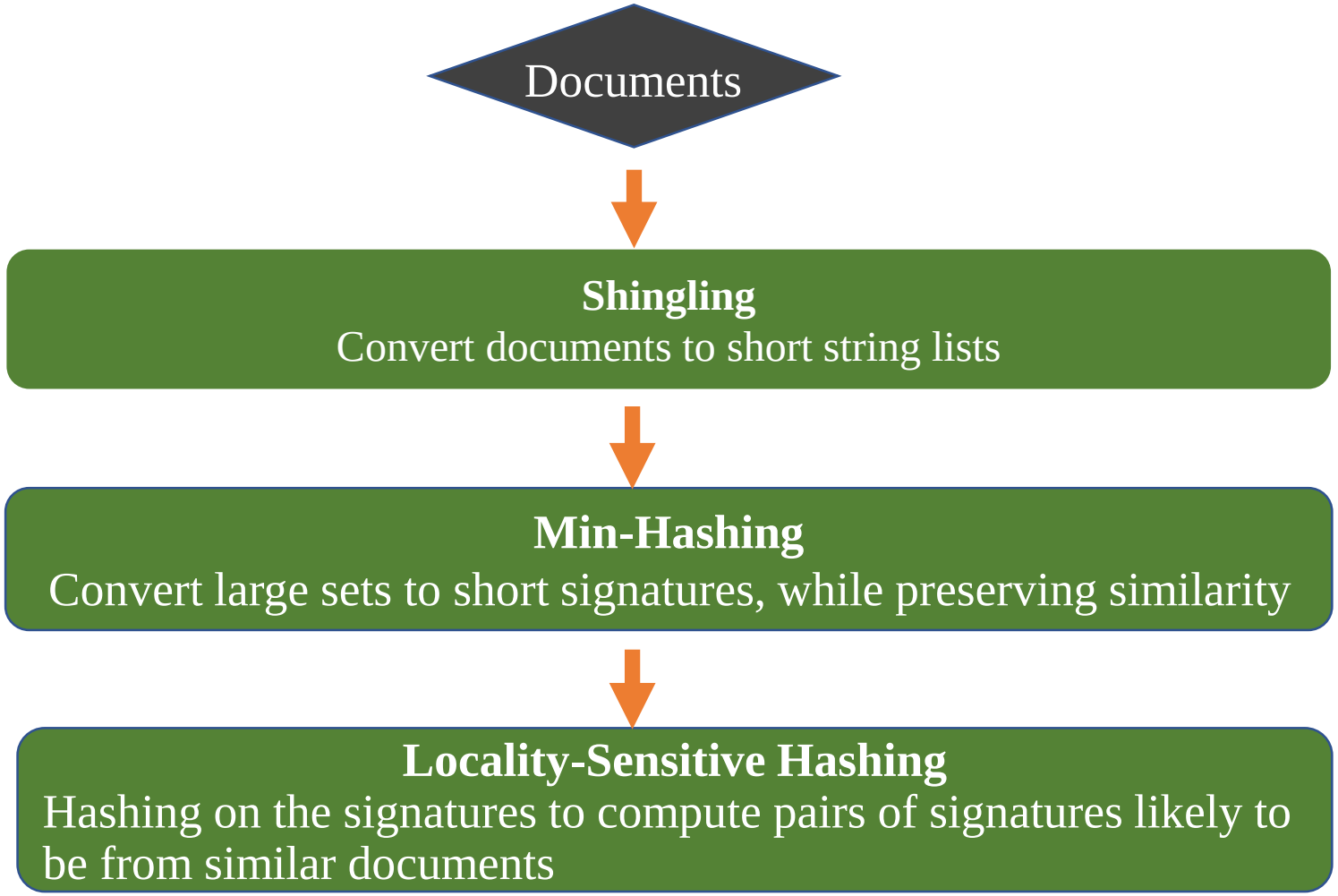
- **Distance function on high-dimensional vector spaces**
  -- Two vectors $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$
  -- L_k-distance:

$$d_k(\boldsymbol{x}, \boldsymbol{y}) = \sqrt[k]{\sum |x_i - y_i|^k}$$

  -- L_1-distance is also called Manhattan distance
  -- L_2-distance is the Euclidean distance

- **Tringle Inequality**
  -- Many metric functions used in data science don't satisfy triangle inequality
  -- Triangle inequality is important for theoretical study.
  -- (Question) Prove the Jaccard distance on sets satisfies the triangle inequality

Part 2 Find Similar Documents



Step 1. Shingling: from Documents to String Sets

D = {Shingling: New Curly Hair Style }

Set of 5-shingles S(D)={Shing, hingl, ingli, …, Style}

Variations:

Set of 3-words

Multiset (bag) of 3-shingles: {Shi, hin, ing, ngl, gli, lin, ing, …}

Shingle Size

Pick *a proper k;*
-- k = 5 for short documents e.g. emails
-- k = 10 for long documents e.g. research articles

Step 2. Min-Hashing

Representation of documents as integer k-shingles is not enough

-- Assume 1,000,000 documents and 1,000,000 comparisons/sec

-- Pairwise comparisons would take 5 days

Min-hashing approach

-- Convert large shingles to short signatures, while preserving similarity

-- Compare signatures of shingles for similarity detection

-- Signatures: short integer vectors that represent the string sets
and reflect their similarity

-- Used by AltaVista for detection of nearly-duplicate webpages

- Min-hashing is a Random process (sampling approach)

  -- **Sacrifice accuracy:** Possible to have false positives (FP) and false negatives (FN)

  -- But fast!

  -- Use a minhash function *h* to keep both FP and FF low, i.e.,

  If sim(D1, D2) is high, then *h*(D1) = *h*(D2) with high probability

  If sim(D1, D2) is low, then *h*(D1) != *h*(D2) with high probability

- Key Issue

  -- How to define such a hashing function

FP errors: two documents with low similarity have the same value

FN errors: two documents with high similarity have distinct values.

Encode shingle sets as bit vectors

- Treat a collection of shingle sets as their characteristic matrix

  -- k-shingle universal set {a, b, c, d, e} (here encode k-shingles into letters)
  -- the rows correspond to k-shingles of the universal set
  -- The columns correspond to the documents
  -- 1 is in the entry (i, j) if the i-th k-shingle is in the j-th document.

k-single	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

Computing Jaccard-based Similarity

D1={a, c, f, g}
D2={b, c, e, g}

k-Shingle	D1	D2
a	1	0
b	0	1
c	1	1
d	0	0
e	0	1
f	1	0
g	1	1

J_{00} = # rows where both elements are 0
J_{11} = # rows where both elements are 1
J_{01} = # rows where (A=0 but B=1) or (A=1 but B=0)

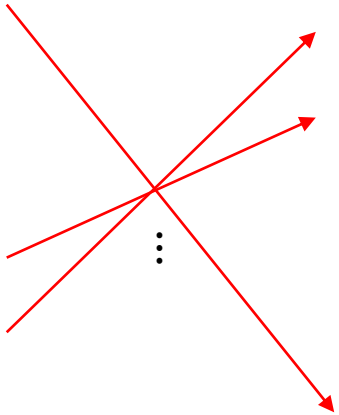
$$SIM(D1, D2) = \frac{J_{11}}{J_{01} + J_{11}} = \frac{2}{6}$$

- D1 is represented as v(D1)=1010011
- D2 becomes v(D2)=0110101
- Use the bitwise AND on v(D1) and v(D2) for J_{11}
- Use the bitwise OR to compute $J_{01} + J_{11}$

Min-hashing Algorithm

1. Start with the characteristic matrix of the shingle sets
2. Randomly permute the matrix rows
3. Min-hash value of any document is the index of the first row (in the permuted order) with a "1" in the corresponding column

	D1	D2
a	1	0
b	0	1
c	1	1
d	0	0
e	0	1
f	1	0
g	1	1



	D1	D2
e	0	1
d	0	0
b	0	1
c	1	1
f	1	0
a	1	0
g	1	1

h(D1)=4
h(D2)=1

Min-hashing algorithm maps a document to a small integer!

Theorem Probability that the minhashing alg. outputs the same value for documents C & D equals their Jaccard-based similarity, i.e. Pr[mh(C) = mh(D)] = sim(C, D)

- Row permutation is equivalent to k-shingle permutation.

Assume there are:

x red k-shingles (that appear in both C and D)
y blue k-shingles (that appear only in one document)
z black k-shingles (that don't appear in both C and D)

- $mh(C) \leq z + 1 \; or \; mh(D) \leq 1 + z$

For any j ($1 \leq j \leq 1 + z$),
Pr[mh(C)=mh(D)=j]
= Pr[the first j-1 selected k-shingles are black and the j-th one is red]
= Pr[the first non-black k-shingle appears at the j-th selection]
× Pr[the j-th k-shingle is red | the j-th k-shingle is the first non-black one]
= Pr[the first non-black k-shingle appears at the j-th selection] × $\frac{x}{x+y}$

Pr[E1 & E2]=Pr[E1] Pr[E2|E1]

k-shingle	C	D
	0	0
	0	0
	0	0
	1	1
	⋮	⋮
	1	0
	0	1

j=4

Pr[mh(C)=mh(D)]
= $\sum_{1 \leq j \leq z+1}$ Pr[mh(C)=mh(D)=j]
$= \frac{x}{x+y} \sum_{1 \leq j \leq z+1}$ Pr[the first non−green k−shingle appears at the j−th selection]= $\frac{x}{x+y}$

False positive/negative errors

- **Idea of MinHash**: Use a function to partition efficiently the document into disjoint clusters so that similar documents are likely found in the same cluster.

- **Issues**
  -- False positive (FP) errors: map the documents with low similarity into a cluster
  -- False negative (FN) errors: Distribute the documents with high similarity into different clusters
  -- Solution for FN: Apply MinHash multiple (k) times to obtain a Minhash vector, called MinHash signatures; Document similarity is estimated using these MinHash signatures.

	D1	D2	D3	D4
a	1	0	1	0
b	0	1	0	1
c	1	1	0	1
d	0	0	0	1
e	0	1	0	1
f	1	0	1	0
g	1	1	1	0

Three Permutations

P1 4376125

P2 4213675

P3 1376254

Signatures

	D1	D2	D3	D4
P1	2	2	3	1
P2	3	2	3	1
P3	1	2	1	2

Apply **P1**

	D1	D2	D3	D4
d	0	0	0	1
c	1	1	0	1
g	1	1	1	0
f	1	0	1	0
a	1	0	1	0
b	0	1	0	1
e	0	1	0	1

Apply **P2**

	D1	D2	D3	D4
d	0	0	0	1
b	0	1	0	1
a	1	0	1	0
c	1	1	0	1
f	1	0	1	0
g	1	1	1	0
e	0	1	0	1

Apply **P3**

	D1	D2	D3	D4
a	1	0	1	0
c	0	1	0	1
g	1	0	1	0
f	1	1	0	1
b	1	0	1	0
e	1	1	1	0
d	0	1	0	1

	D1	D2	D3	D4
a	1	0	1	0
b	0	1	0	1
c	1	1	0	1
d	0	0	0	1
e	0	1	0	1
f	1	0	1	0
g	1	1	1	0

Signatures

	D1	D2	D3	D4
P1	2	2	3	1
P2	3	2	3	1
P3	1	2	1	2

Original (New) Similarity

	D1	D2	D3	D4
D1		$\frac{1}{3}\left(\frac{1}{3}\right)$	$\frac{3}{4}\left(\frac{2}{3}\right)$	$\frac{1}{7}\left(\frac{0}{3}\right)$
D2			$\frac{1}{6}\left(\frac{2}{3}\right)$	$\frac{3}{6}\left(\frac{1}{3}\right)$
D3				$\frac{0}{7}\left(\frac{2}{3}\right)$

- Pr [D1 and D2 have the same minhash value] = [Jaccard-based similarity sim(D1, D2)]
- The expected portion of rows in which the signatures of D1 and D2 have the same value is their similarity.

- Issues: take too much time if the signatures are derived from 200 random permutations.

Efficient simulation of Min-Hash Signatures

1. Use a random hash functions h that maps rows to as many many buckets (i.e., values) as rows.
2. Update the minimum hash values SIG(D, h) for each D as follows:

   -- Initialization: set SIG(D, h) to ∞ for h

   -- for r=1, 2, …, R:

   if entry(r, D)=1, reset SIG(D, h) to min(h(r), SIG(D, h))

Use random hashing
To replace minhashing function

Row	S_1	S_2	S_3	S_4	Hash function h_1	Hash function h_2
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

Initialization step:

	S_1	S_2	S_3	S_4
h_1	∞	∞	∞	∞

Row 0

$Entry(0, S_1) = Entry(0, S_4) = 1$

$SIG(S_1, h_1) = Min(h_1(0), \infty)$=1;

$SIG(S_4, h_1) = Min(h_1(0), \infty)$=1;

	S_1	S_2	S_3	S_4
h_1	1	∞	∞	1

Row 1

$Entry(1, S_3) = 1$

$SIG(S_3, h_1) = Min(h_1(1), \infty)$=2;

	S_1	S_2	S_3	S_4
h_1	1	∞	2	1

Exercise: Estimate the signature values for hash function

h(x)=2x+4 (mod 5)

for S1 to S4.

Row 2

$Entry(2, S_2) = Entry(2, S_4) = 1$

$SIG(S_2, h_1) = Min(h_1(2), \infty)$=3;

$SIG(S_4, h_1) = Min(h_1(2), 1)$=1;

	S_1	S_2	S_3	S_4
h_1	1	3	2	1

Row 3

$Entry(3, S_1) = Entry(3, S_3) = Entry(3, S_4) = 1$

	S_1	S_2	S_3	S_4
h_1	1	3	2	1

Row 4

$Entry(4, S_3) = 1$

$SIG(S_3, h_1) = Min(h_1(4), 2)$=0;

	S_1	S_2	S_3	S_4
h_1	1	3	0	1

Row	S_1	S_2	S_3	S_4	h_1
0	1	0	0	1	1
1	0	0	1	0	2
2	0	1	0	1	3
3	1	0	1	1	4
4	0	0	1	0	0

Step 3: Locality-Sensitive Hashing for Documents

- Short signatures output from Step 2 preserve document similarity well.

	D1	D2	D3	D4	…	D999999	D1000000
h_1	3	4	3	4	…	3	2
h_2	4	1	1	1	…	4	2
h_3	2	1	2	1	…	3	3
⋮	⋮	⋮	⋮	⋮		⋮	⋮
h_{247}	3	3	5	2	…	1	4
h_{248}	3	2	1	2	…	1	4

- For 1 million documents, the above table takes 1G memory. However, there are $(10^6 - 1)10^6/2$ possible pairs of documents and computing all similarities takes 6 days on a laptop.

Naïve Idea

- To reduce time complexity, we apply hashing function again

	D1	D2	D3	D4	…	D999999	D1000000
h_1	3	4	3	4	…	3	2
h_2	1	1	1	1	…	4	2
h_3	2	1	2	1	…	3	3
⋮	⋮	⋮	⋮	⋮		⋮	⋮
h_{249}	4	3	5	3	…	1	4
h_{250}	3	2	3	2	…	3	4

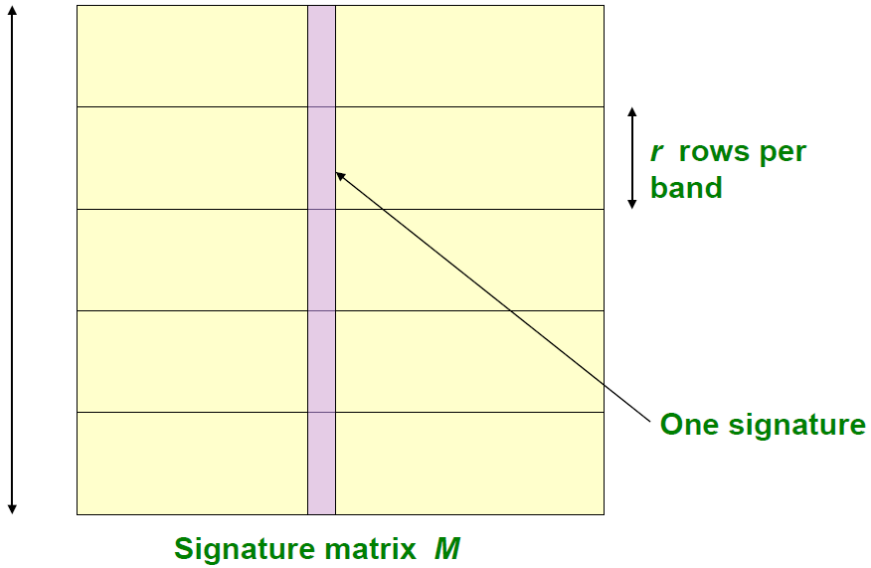
- **High FN errors!** two signatures with a few different components can be mapped into distinct buckets

- To reduce FN errors, we divide signature into bands;
  map two documents into the same bucket if they have the same value
  in one of the bands.

	D1	D2	D3	D4	…	D999999	D1000000	
h_1	3	4	3	4	…	3	2	
h_2	1	1	1	1	…	4	2	Band 1
h_3	2	1	2	1	…	3	3	
⋮	⋮	⋮	⋮	⋮		⋮	⋮	
h_{238}	4	2	4	4	…	1	4	
h_{239}	4	3	5	5	…	1	4	Band 80
h_{240}	3	2	3	3	…	3	4	

Partition *M* into *b* Bands

- Divide the signature matrix M into b bands of r rows
- For each band, hash the portion of each column to a hash table with k buckets. Make *k* as large as possible such that different sub-signatures are likely mapped into distinct buckets.
- Candidate column pairs are those that are hashed to the the same bucket for ≥ 1 band
- Pick b and r to catch many similar pairs, but few non-similar pairs

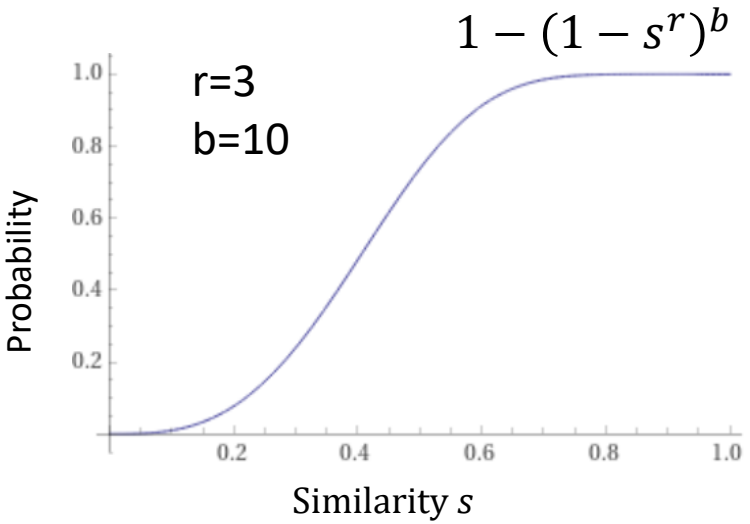


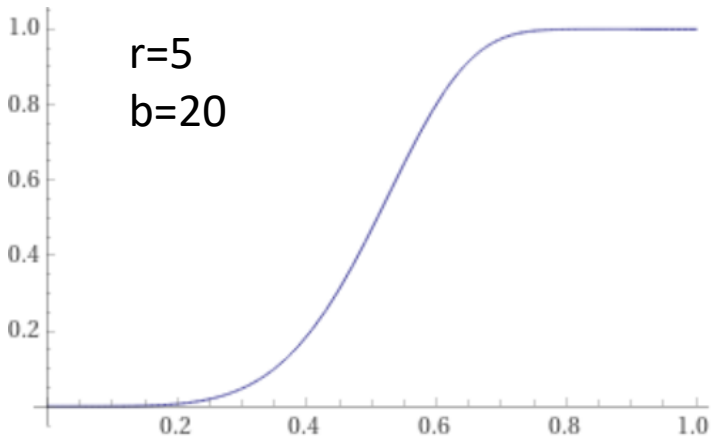
Theorem Suppose the signatures are divided into b bands of r row each.
For any two documents with the Jaccard similarity s,

Pr[their signatures agree in all the rows in at least one band]= $1 - (1 - s^r)^b$

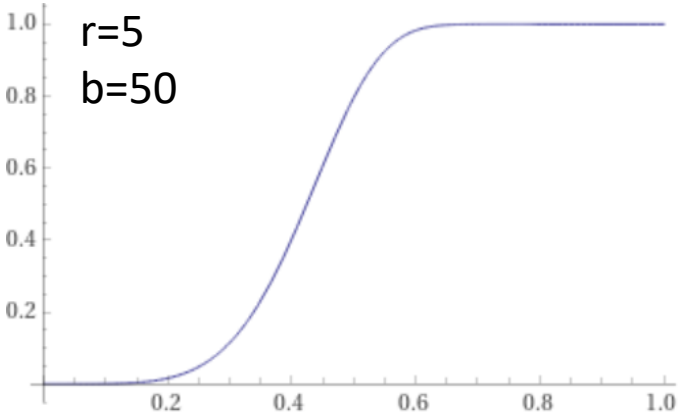
Proof.

1. The probability that two signatures agree in all rows of one particular band is s^r.
2. The probability that two signatures disagree in at least one row of a particular band is 1-s^r.
3. The probability that two signatures disagree in every band is $(1 - s^r)^b$.
4. The probability that two signatures agree in at least one band, and therefore become a candidate pair, is $1 - (1 - s^r)^b$.





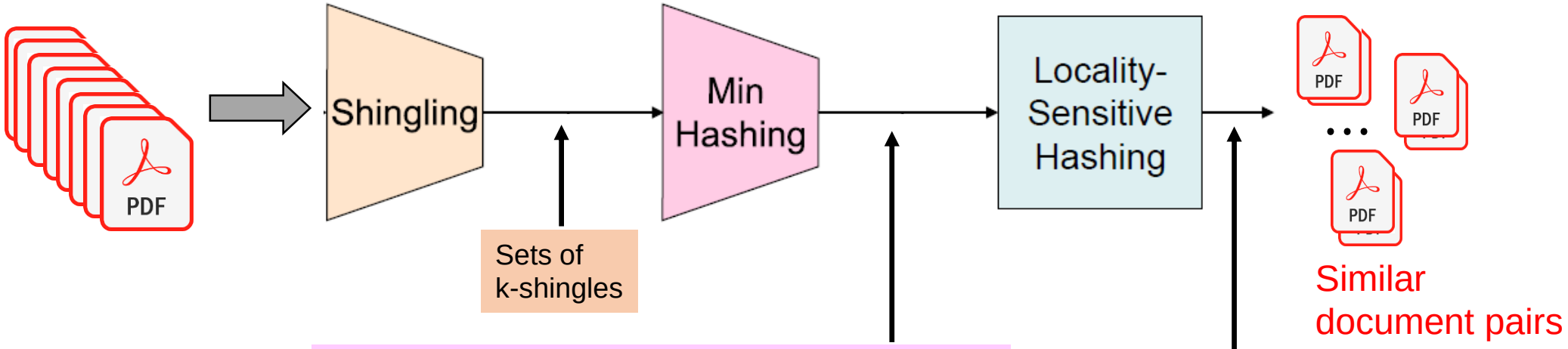
s	$1-(1-s^r)^b$
.2	.006
.3	.047
.4	.186
.5	.470
.6	.802
.7	.975
.8	.9996



For fix r, b. P(s)<<s if s<0.4;
P(s)>> s if s>0.6

The steepness of the S-curve reflects how effectively we can avoid false positives and false negatives for large r and b.

Summary of Finding Similar Documents



Document signatures

Documents D are mapped to short integer vectors h(D) such that the expected fraction of components with same values in h(D1) and h(D2) equals roughly the Jaccard similarity S(D1, D2).

Balance trade-off between specificity and sensitivity

Select proper r and b to divide signatures into b bands of r components each, and only measure the similarity of a pair of signatures if they are identical in at least one band.

To reduce false positives, increase r; To reduce false negative, increase b.

Part 3 The Theory of Locality-Sensitive Functions

For similar document detection, we use magic min-hash functions, which have the following properties:

1. Min-hash functions must make close pairs be candidate pairs more likely than distant pairs.
2. Min-hash functions must be statistically independent.
3. Min-hash must be efficient, in two ways:
   -- Must be able to identify candidate pairs in time much less than the time it takes to look at all pairs.
   -- Must be combinable to build functions that are better at avoiding FN/FP errors, and the combined functions must also take time that is much less than the number of pairs.

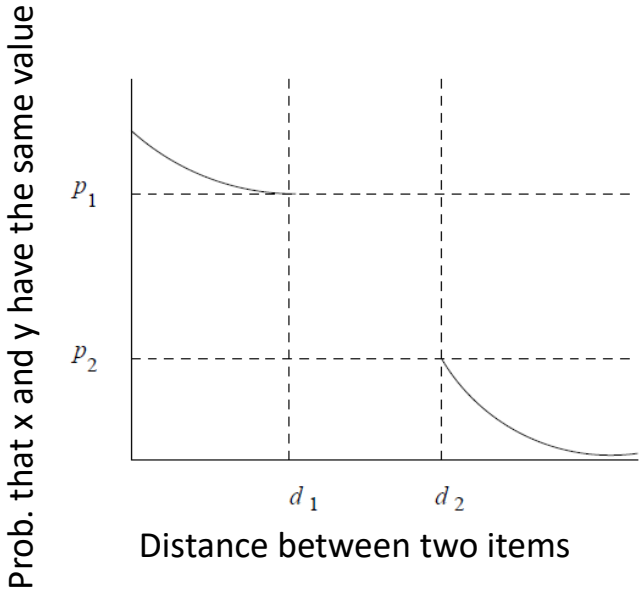
The family of minhash functions are a $(d_1, d_2, 1-d_1, 1-d_2)$-sensitive family for any d_1 and d_2 such that $0 \leq d_1 < d_2 \leq 1$.

Locality-Sensitive Functions

Let d_1 and d_2 be two distance values according to a distance measure d, such that $d_1 < d_2$.

A family F of functions is said to be (d_1, d_2, p_1, p_2)-sensitive if for every f in F:

-- If $d(x,y) \leq d_1$, then $\Pr[f(x) = f(y)] \geq p_1, p_1 > 0$.
-- If $d(x,y) \geq d_2$, then $\Pr[f(x) = f(y)] \leq p_2,\ p_2 > 0$.



Theorem There are (d_1, d_2, p_1, p_2)-sensitive function families for the Euclidean distance

Summary

- Jaccard Similarity
- Shingling
- Min-hashing/Min-hash Signatures
- Locality-Sensitive Hashing for Signatures
- Distance Measures
- Locality-Sensitive Hashing families for distance measures