Big Data Analytics

Finding Similar Items

SHYI-CHYI CHENG

Some slides credit to Prof. Tao Yang
Computer Science & Engineering Dept.
University of California Santa Barbara, USA

Outline

- Motivation
- Duplication Detection and Similarity Computing
- Shingling of duplication comparison
- Minhashing
- Locality-Sensitive Hashing (LSH)

Motivation

- **Duplicate and near-duplicate detection** is a fundamental data-mining problem in order to examine data for "similar" items
  - Copies, versions, plagiarism, spam, mirror sites
  - Over 30% of the web pages in a large crawl are exact or near duplicates of pages in the other 70%
- Duplicates consume significant resources during crawling, indexing, and search
- **Similar query suggestions**
- **Advertisement**
  - **coalition and spam detection**

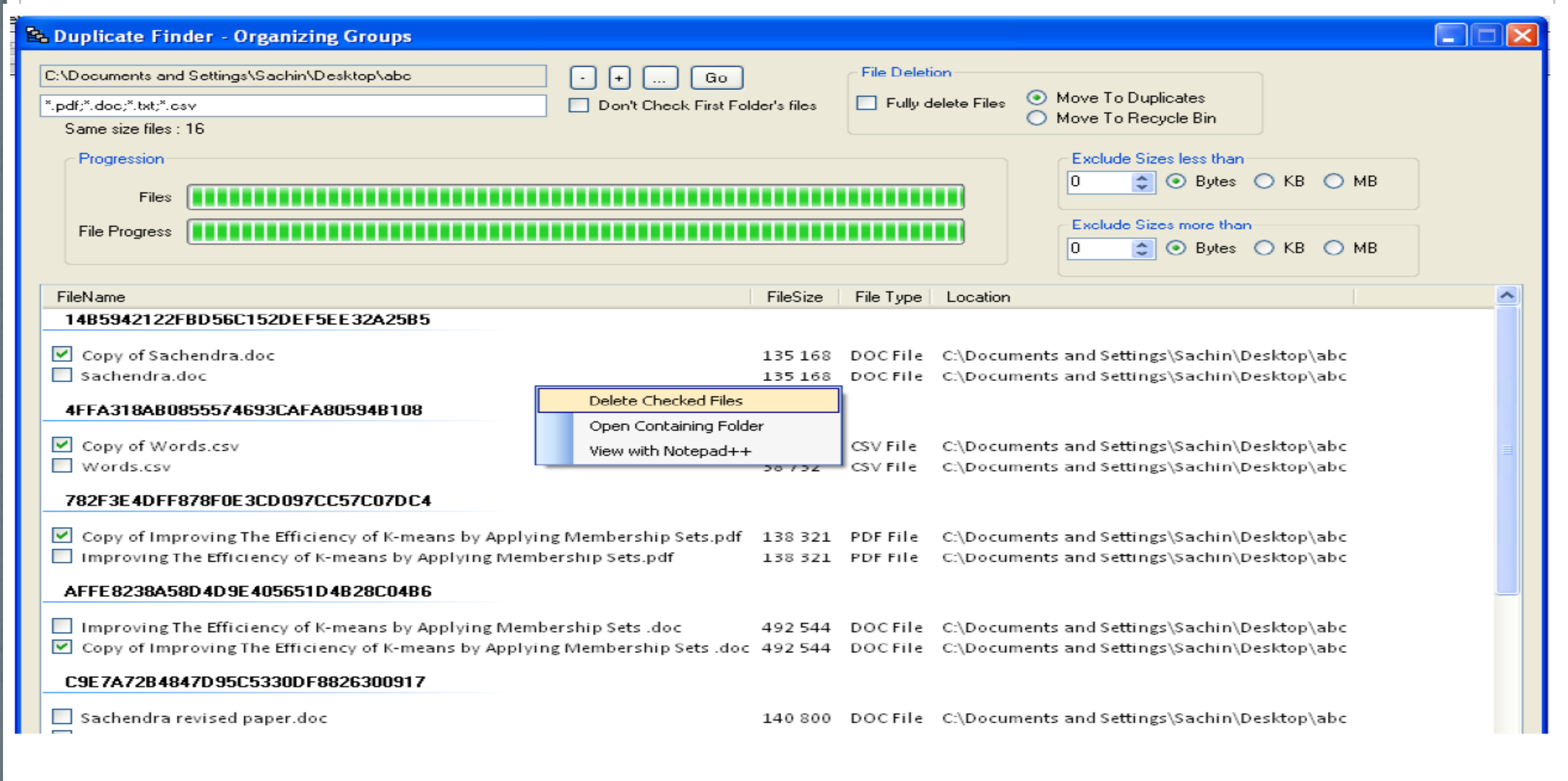
Duplicate Detection

- *Exact* duplicate detection is relatively easy
  - Content fingereprints
  - MD5, cyclic redundancy check (CRC)
- Checksum techniques
  - A checksum is a value that is computed based on the content of the document

T	r	o	p	i	c	a	l		f	i	s	h	*Sum*
54	72	6F	70	69	63	61	6C	20	66	69	73	68	508

  - Possible for files with different text to have same checksum

Exact Duplicate Detection



Near-Duplicate News Articles

MLB／柯蕭先發就是勝利保證　19勝獨步大聯盟

ETtoday.net ETtoday – 2014年9月15日 下午12:25

記者張克銘／綜合報導

道奇先發柯蕭(Clayton Kershaw)幾乎出場就是勝利的保證，今日先發面對巨人，8局飆出9次三振，加上坎普(Matt Kemp)的全壘打，最終道奇4:2擊敗巨人，柯蕭拿下獨步大聯盟的19勝。

柯蕭全場8局被敲出7支安打失掉2分，僅有1次保送、送出9次三振，拿下個人5連勝，此外自從美國時間5月17日面對響尾蛇全場失掉7分外，已經連續21場球失分都在3分以下，展現強大的壓制能力。

此外19勝也是目前大聯盟第一，防禦率1.70也是大聯盟排名第一，WHIP0.83也是大聯盟第一，依柯蕭態勢看來，柯蕭有機會自2011年後，再次挑戰20勝的記錄，更有機會連續連續兩年挑戰賽揚獎。

大聯盟勝投王 柯蕭19勝到手

民視新聞 民視 – 2014年9月15日 下午2:00

大聯盟史上身價最高的投手，道奇隊賽揚王牌柯蕭，今天在客場對宿敵巨人隊，柯蕭主投8局飆出9次三振，只失掉2分，率領道奇以4：2獲勝，柯蕭成為大聯盟本季第一個拿下19勝的投手，自責分率低到只有1.70，國聯賽揚獎幾乎是他的囊中之物，道奇在國聯西區穩居第一，封王魔術數字降至11。

前一場在自家遭到道奇，17：0大屠殺的巨人，現在又碰到大魔王ClaytonKershaw，真的是雪上加霜。

Near Duplicate Detection

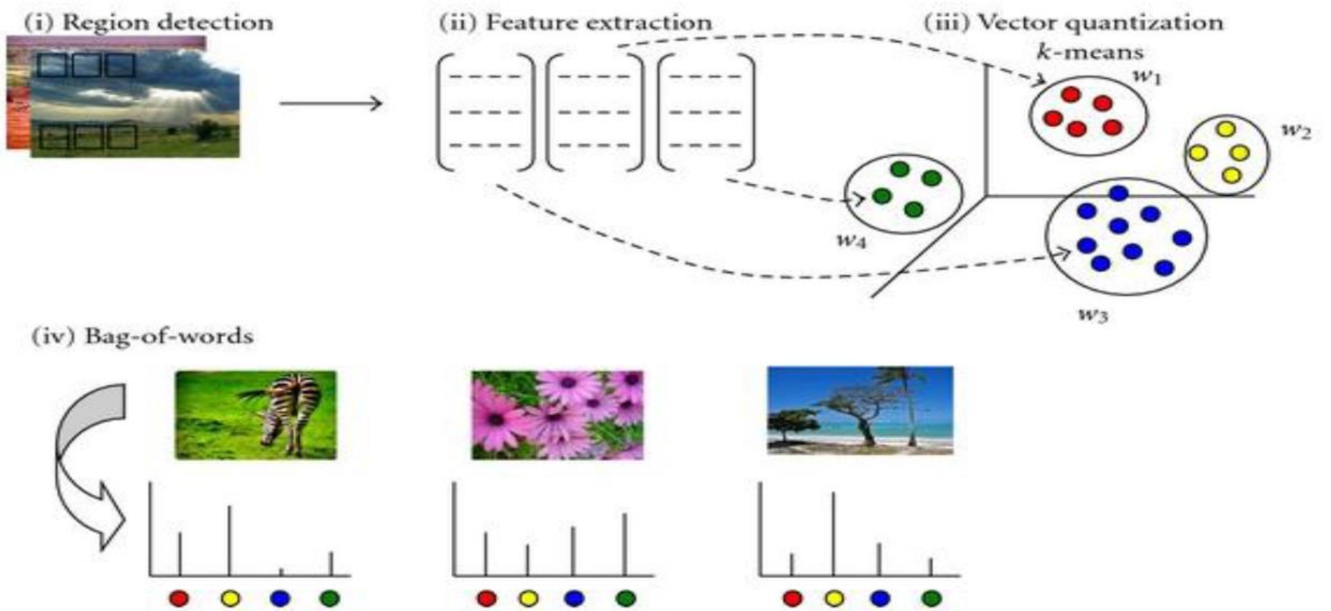
- More challenging task
  - Are web pages with same text context but different advertising or format near-duplicates?
- *Near-Duplication*: Approximate match
  - Compute syntactic similarity with an edit-distance measure
  - Use similarity threshold to detect near-duplicates
    - E.g., Similarity > 80% ➔ Documents are "near duplicates"
    - Not transitive though sometimes used transitively

Near Duplicate Detection

- Search
  - Find near-duplicates of a document
  - O(N) comparisons required
  - k-nearest-neighbor (k-NN) procedure: fast algorithms available
- Discovery
  - Find all pairs of near-duplicate documents in the collection
  - O(N^2) comparisons required
- IR techniques are effective for search scenario
- For discovery, further techniques used to generate compact representations are required

Similarity Computing

- Similarity Search
  - Vector Space
  - Bag of Words



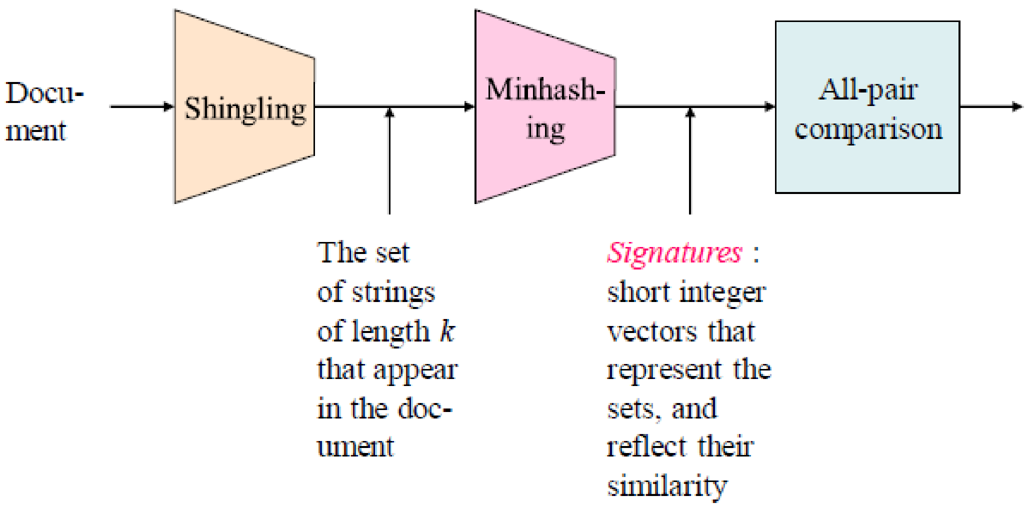
Similarity Computing

- Lexical Similarity
  - Shingling: *k*-shingles define a set of all *k* size non repeatable substrings of the document, and group them as a single object.
  - Ex., Input "*This LSH Project is good*";

    3-shingles set: {"This LSH Project", "LSH Project is", "Project is good"}.

Similarity Computing

- Shingling: **convert documents, emails, etc., to fingerprint sets.**
- Minhashing: **convert large sets to short signatures, while preserving similarity.**



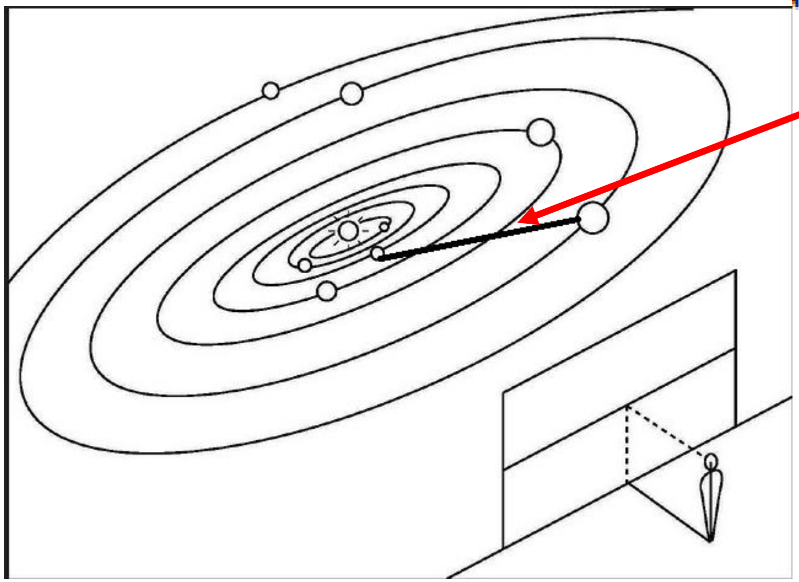
Distance Metrics

- A distance measure is a function $d(p_1, p_2)$ which takes the two arguments, some points in space as input and returns a real number, satisfying the following conditions:
  - $d(p_1, p_2) \geq 0$
  - $d(p_1, p_2) = 0$ only if $p_1 = p_2$
  - $d(p_1, p_2) = d(p_2, p_1)$
  - $d(p_1, p_2) \leq d(p_1, p_3) + d(p_3, p_2)$.

Euclidean Distance (L_2)

- The Euclidean distance between any two points x and y:

$$d([x_1, x_2, \ldots, x_n], \ [y_1, y_2, \ldots, y_n]) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$



Euclidean distance

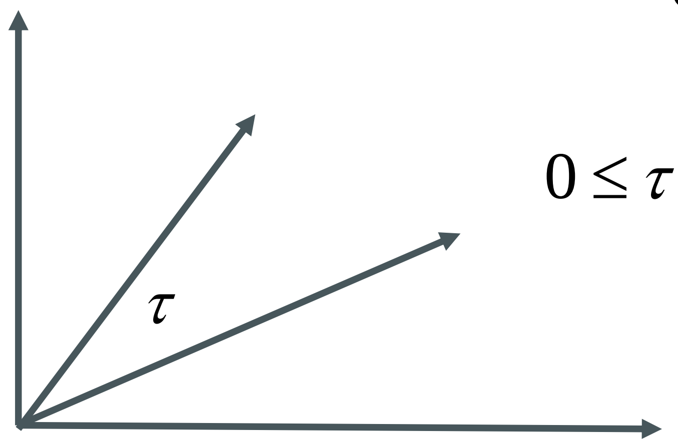
Euclidean Distance (L_r)

- A common distance measure is Manhattan distance called the L1-norm where the sum of magnitudes of the difference in each dimension is the distance between two points.

$$d([x_1, x_2, \ldots, x_n],\ [y_1, y_2, \ldots, y_n]) = (\sum_{i=1}^{n} |x_i - y_i|^r)^{1/r}$$

Cosine Distance

- The cosine distance between two points is the angle formed between their vectors.



$$\cos\tau = \frac{x \cdot y}{|x||y|}$$

$$0 \le \tau \le 180 ==> -1 \le \cos\tau \le 1$$

Hamming Distance

- The Hamming Distance is used for the Boolean vectors i.e. which contain only 0 or 1.
- The number of items in which the two items differ is the hamming distance between them.
- Can be implemented by the "AND" operator

Edit Distance

- Considering the two strings x =x1 x2…. xn and y = y1y2…. ym, the minimum number of times the insertions and deletions of single characters is done to convert x to y or y to x is the distance between them.
- Use the longest common subsequence (LCS) of x and y to compute the distance

Ex., x = abcde, y = acfdeg

→ The LCS: acde

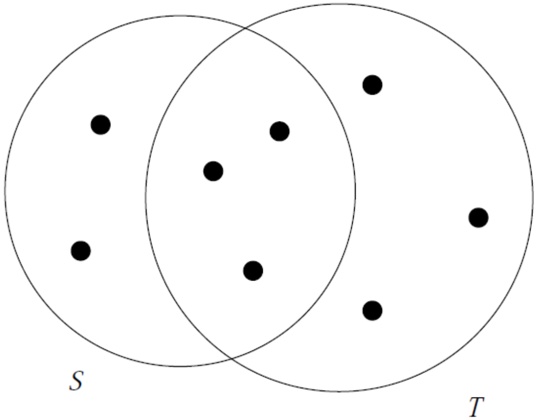
→ Edit distance: $5 + 6 - 2 \times 4 = 3$

Jaccard Distance

- The Jaccard Distance J-DIS(S1,S2), between two sets is defined as one minus Jaccard similarity between those two sets i.e.

$$\text{J-DIS}(S1, S2) = 1 - \text{J-SIM}(S1, S2).$$

- J-SIM(S,T) is $|S \cap T| / |S \cup T|$
  - ex.,



J-SIM(S, T) = 3/8

Fingerprint Generation Process for Documents

1. The document is parsed into words. Non-word content, such as punctuation, HTML tags, and additional whitespace, is removed.
2. The words are grouped into contiguous *n-grams* for some n. These are usually overlapping sequences of words, although some techniques use non-overlapping sequences.
3. Some of the n-grams are selected to represent the document.
4. The selected n-grams are hashed to improve retrieval efficiency and further reduce the size of the representation.
5. The hash values are stored, typically in an inverted index.
6. Documents are compared using overlap of fingerprints

Fingerprint Example for Web Documents

Tropical fish include fish found in tropical environments around the world, including both freshwater and salt water species.

(a) Original text

tropical fish include, fish include fish, include fish found, fish found in, found in tropical, in tropical environments, tropical environments around, environments around the, around the world, the world including, world including both, including both freshwater, both freshwater and, freshwater and salt, and salt water, salt water species

(b) 3-grams

938 664 463 822 492 798 78 969 143 236 913 908 694 553 870 779

(c) Hash values

664 492 236 908

(d) Selected hash values using *0 mod 4*

Computing Similarity with Shingles

- **Shingles (Word *k*-Grams)**

  Suppose our document D is the string abcdabd, and we pick $k = 2$. Then the set of 2-shingles for D is {ab, bc, cd, da, bd}

- Similarity measure between two docs (= sets of shingles)
  - Size_of_Intersection / Size_of_Union

J-SIM

***k* should be picked large enough that the probability of any given shingle appearing in any given document is low.**

Similarity-Preserving Summaries of Sets

- Sets of shingles are large.
  - Solution: to replace large sets by much smaller representations called "signatures."
- The important property for signatures is that we can compare the signatures of two sets and estimate the Jaccard similarity of the underlying sets from the signatures alone.

Matrix Representation of Sets

- The universal set: {a, b, c, d, e}
- S1 = {a, d}, S2 = {c}, S3 = {b, d, e}, and S4 = {a, c, d}
- The matrix representation:

$Element$	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

Minhashing

- To minhash a set represented by a column of the characteristic matrix, pick a permutation of the rows.
- The minhash value of any column is the number of the first row, in the permuted order, in which the column has a 1

Example

- Pick the order of rows beadc

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

beadc →

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

- Compute the minhashing values

  $h(S_1) = a$, $h(S_2) = c$, $h(S_3) = b$, and $h(S_4) = a$

→ $S_1 \sim S_4$?

Minhashing and Jaccard Similarity

- The probability that the minhash function for a random permutation of rows produces the same value for two sets equals the Jaccard similarity of those sets. Why?

Proof with Boolean Matrices

- Rows = elements of the universal set.
- Columns = sets.
- 1 in row *e* and column *S* if and only if *e* is a member of *S*.
- Column similarity is the Jaccard similarity of the sets of their rows with 1.
- Typical matrix is sparse.

C_1	C_2	
0	1	*
1	0	*
1	1	* *
0	0	
1	1	* *
0	1	*

J-SIM $(C_1, C_2) = 2/5 = 0.4$

J-SIM(S,T) is $|S \cap T| / |S \cup T|$

Proof with Boolean Matrices

- For columns C_i, C_j, four types of rows

	C_i	C_j
A	1	1
B	1	0
C	0	1
D	0	0

- Overload notation: A = # of rows of type A
- Claim

$$\text{J-SIM}(C_i, C_j) = \frac{A}{A+B+C}$$

Proof with Boolean Matrices

- For columns C_i, C_j, four types of rows

	C_i	C_j
A	1	1
B	1	0
C	0	1
D	0	0

- Overload notation: A = # of rows of type A
- Claim

$$\text{J-SIM}(C_i, C_j) = \frac{A}{A+B+C}$$

Approximated Representation with Sketching

- Computing exact set intersection of shingles between all pairs of documents is expensive
  - Approximate using a subset of shingles (called sketch vectors)
  - Create a sketch vector using minhashing.
    - For doc d, $\text{sketch}_d[i]$ is computed as follows:

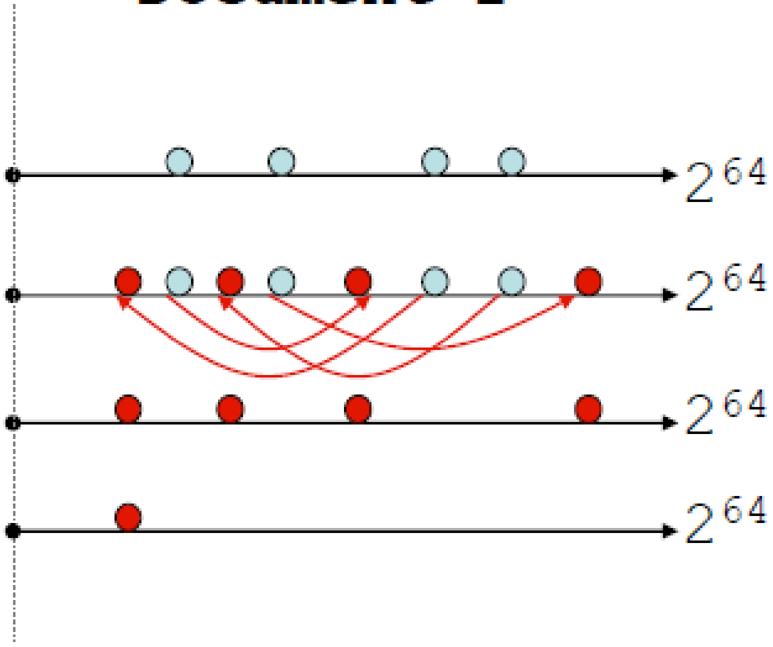
      Let f map all shingles in the universe to $0..2^m$

      Let π_i be a specific random permutation on $0..2^m$

      Pick MIN $\pi_i(f(s))$ over all shingles s in this document d
  - Documents which share more than t (say 80%) in sketch vector's elements are *similar*

Computing Sketch[i] for Doc1

Document 1

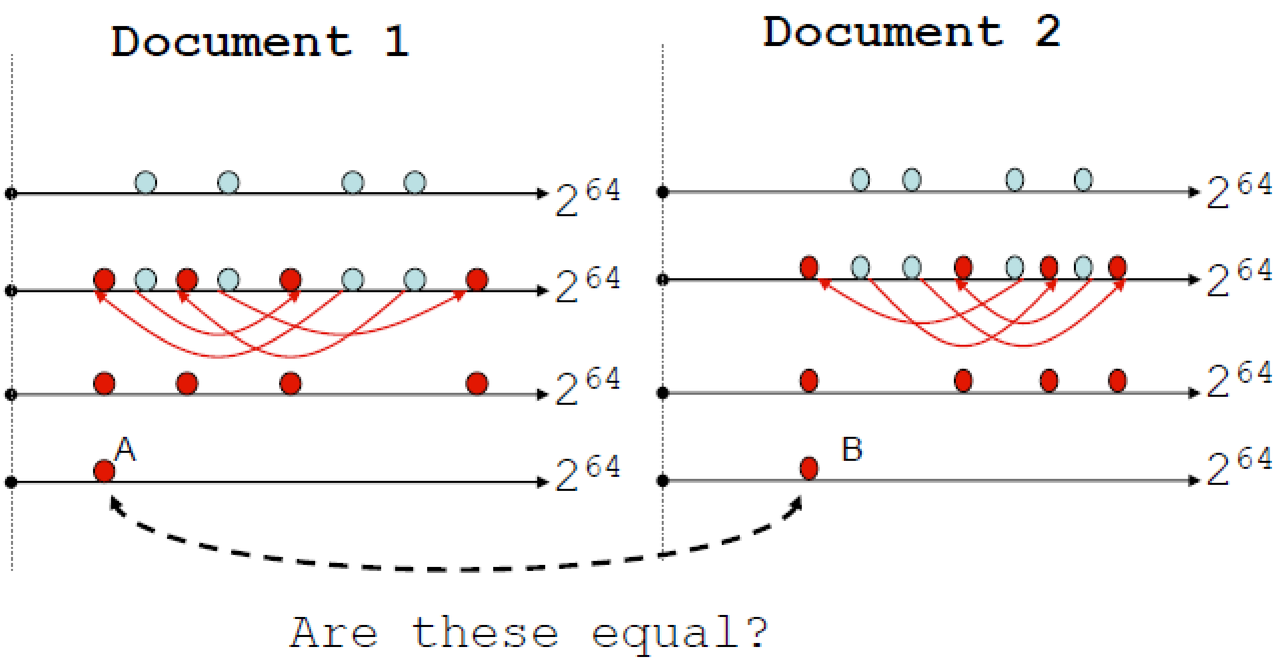


Start with 64 bit shingles

Permute on the number line with π_i

Pick the min value

Test if Doc1.Sketch[i] = Doc2.Sketch[i]



Test for 200 random permutations: $\pi_1, \pi_2, \ldots \pi_{200}$

Computing Minhash Signatures for Big Data

- It is not feasible to permute a large characteristic matrix explicitly
  - Solution:
    - to simulate the effect of a random permutation by a random hash function that maps row numbers to as many buckets as there are rows.
    - To pick n randomly chosen hash functions $h_1, h_2, \ldots, h_n$ on the rows

Computing Minhash Signatures for Big Data

- Let SIG(i, c) be the element of the signature matrix for the ith hash function and column c
- Initially, set SIG(i, c) to ∞ for all i and c.
- For each row *r* do

1. Compute $h_1(r), h_2(r), \ldots, h_n(r)$.
2. For each column c do the following:
   (a) If c has 0 in row r, do nothing.
   (b) However, if c has 1 in row r, then for each $i = 1, 2, \ldots, n$ set $\mathrm{SIG}(i, c)$ to the smaller of the current value of $\mathrm{SIG}(i, c)$ and $h_i(r)$.

Example

Input

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

Initial signature matrix

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

Hash row 0

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

Example

Hash row 1

	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

Hash row 2

	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

Hash row 3

	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

Hash row 4

	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

Example

Final signature matrix

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

J-SIM(S1, S4) ~ 1.0 (2/3 actually)

J-SIM(S1, S2) ~ 0.0 (0 actually)

J-SIM(S1, S3) ~ 0.5 (1/4 actually)

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

Locality-Sensitive Hashing

All-Pair Comparison is expensive

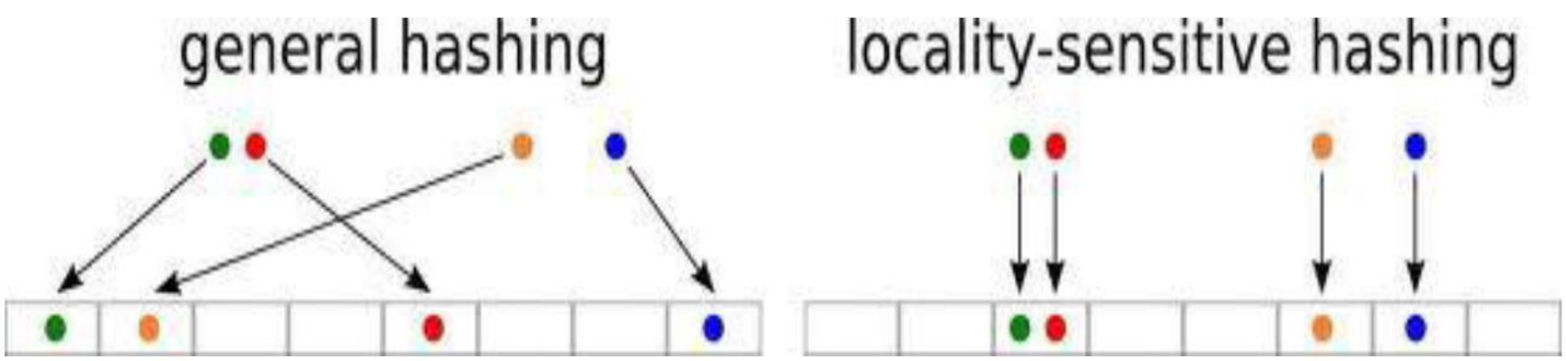
- The performance of minhash still limited by the large number (C_2^n)of pairs of documents
  - n= 1,000,000, #pairs ~ half a trillion ($5\text{x}10^{11}$)
  - If it takes a microsecond to compute the similarity of two signatures, then it takes almost six days to compute all the similarities on that laptop.
- Solutions
  - Parallel programming
  - Locality-sensitive hashing (LSH) or near-neighbor search (NNR)

Key-Issues of Similarity Search in Big Data

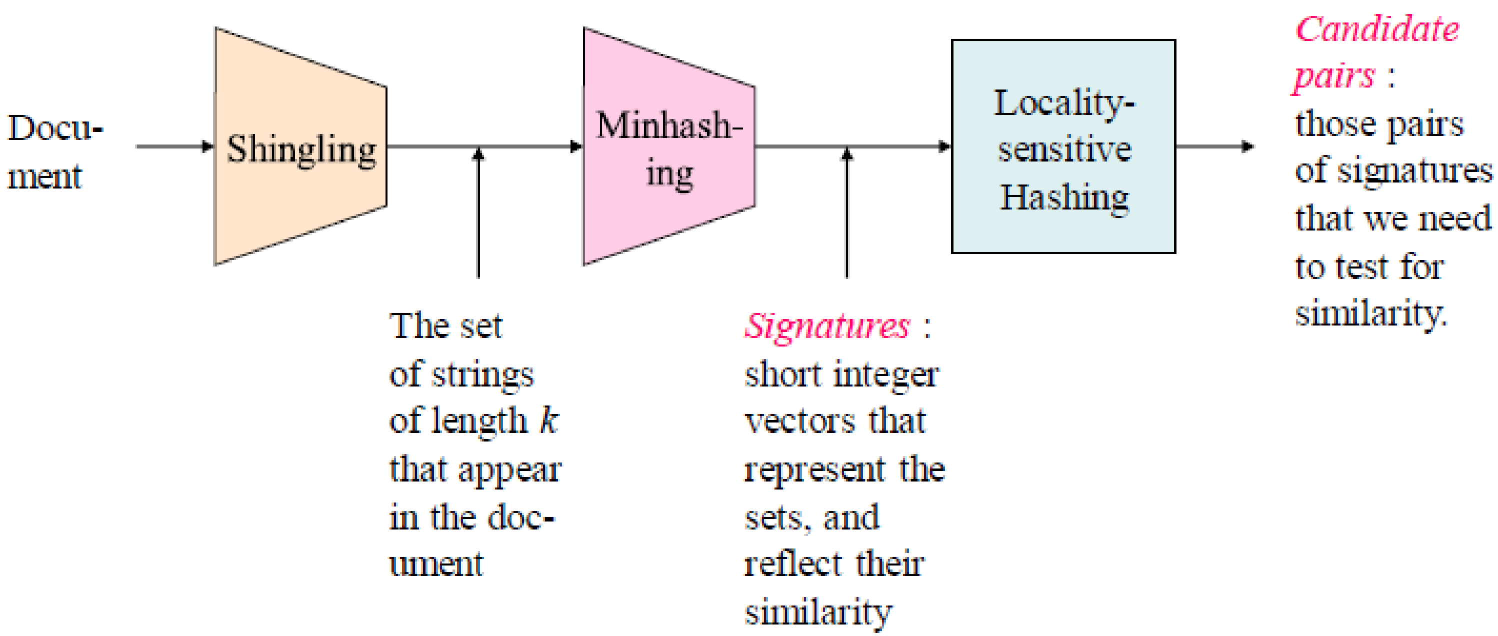
- **Execution time complexity of high dimensional data**
- **Space complexity**
- **Accuracy**
  - Too many false positives and false negatives

General Hash vs. LSH

- In general hashing, closed (near) items may be hashed in different locations after hashing
- Locality Sensitive Hashing items maintain their closeness even after hashing (mapping)
  - *candidate pairs* are those pairs which hash to the same bucket

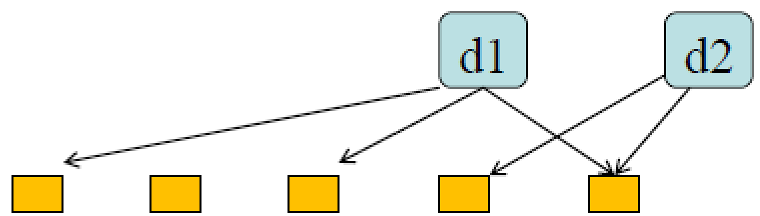


The Big Picture



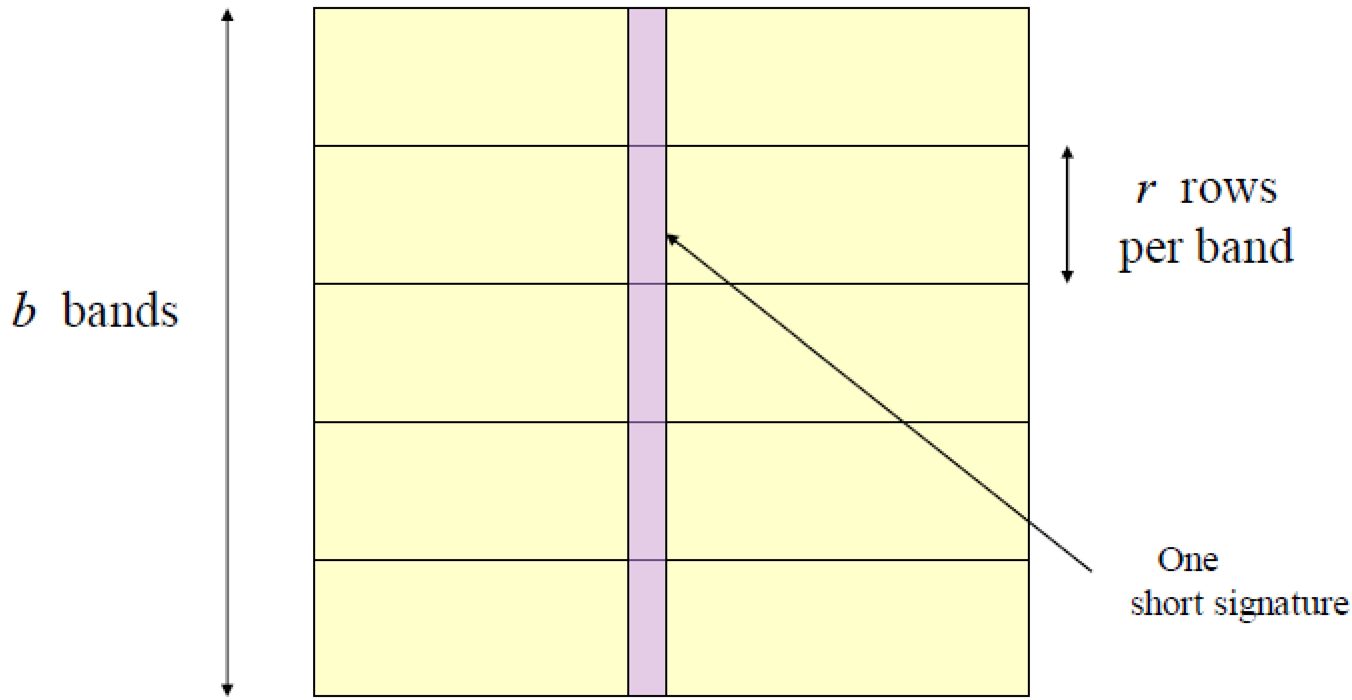
Locality-Sensitive Hashing

- General idea: Use a function f(x,y) that tells whether or not *x* and *y* is a *candidate pair*: a pair of elements whose similarity must be evaluated.
- Map a document to many buckets



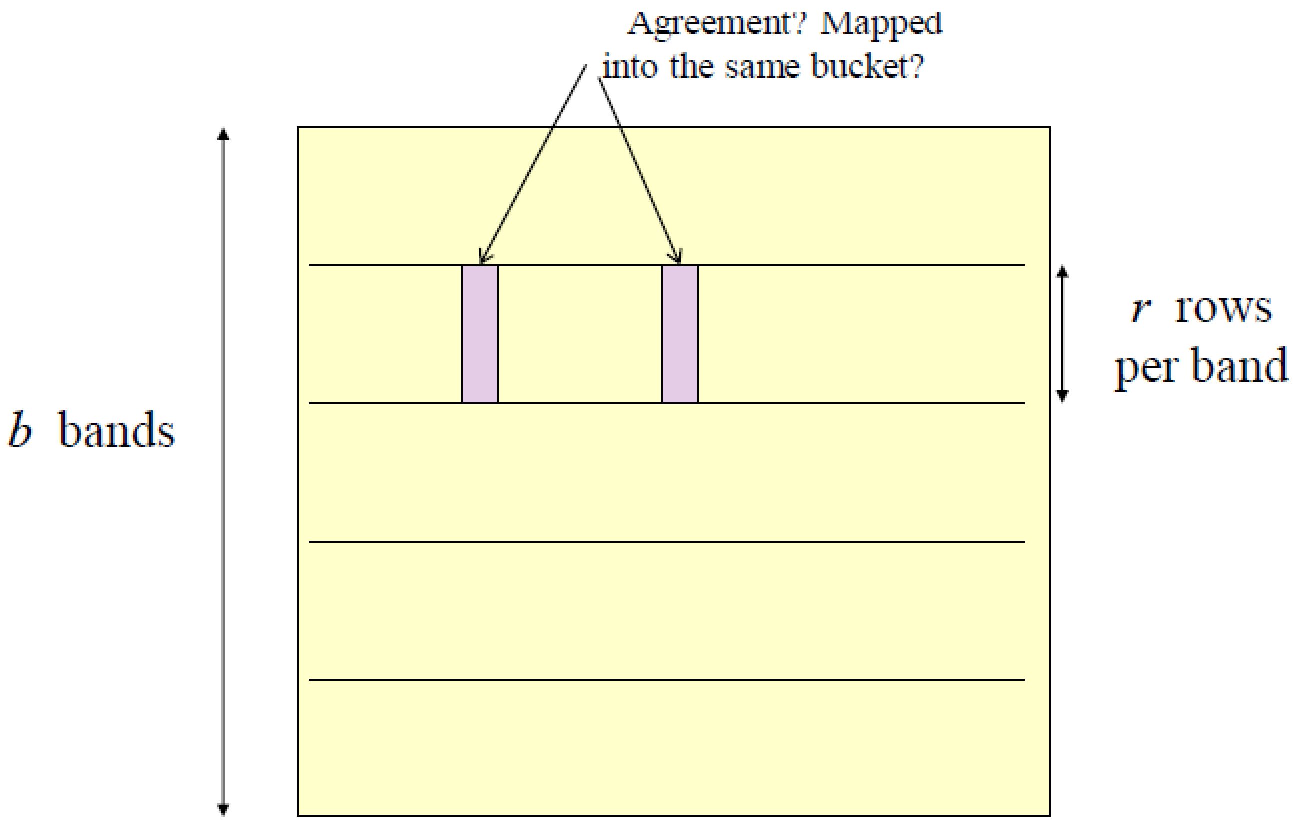
- Make elements of the same bucket candidate pairs.

Another view of LSH: Produce signature with bands

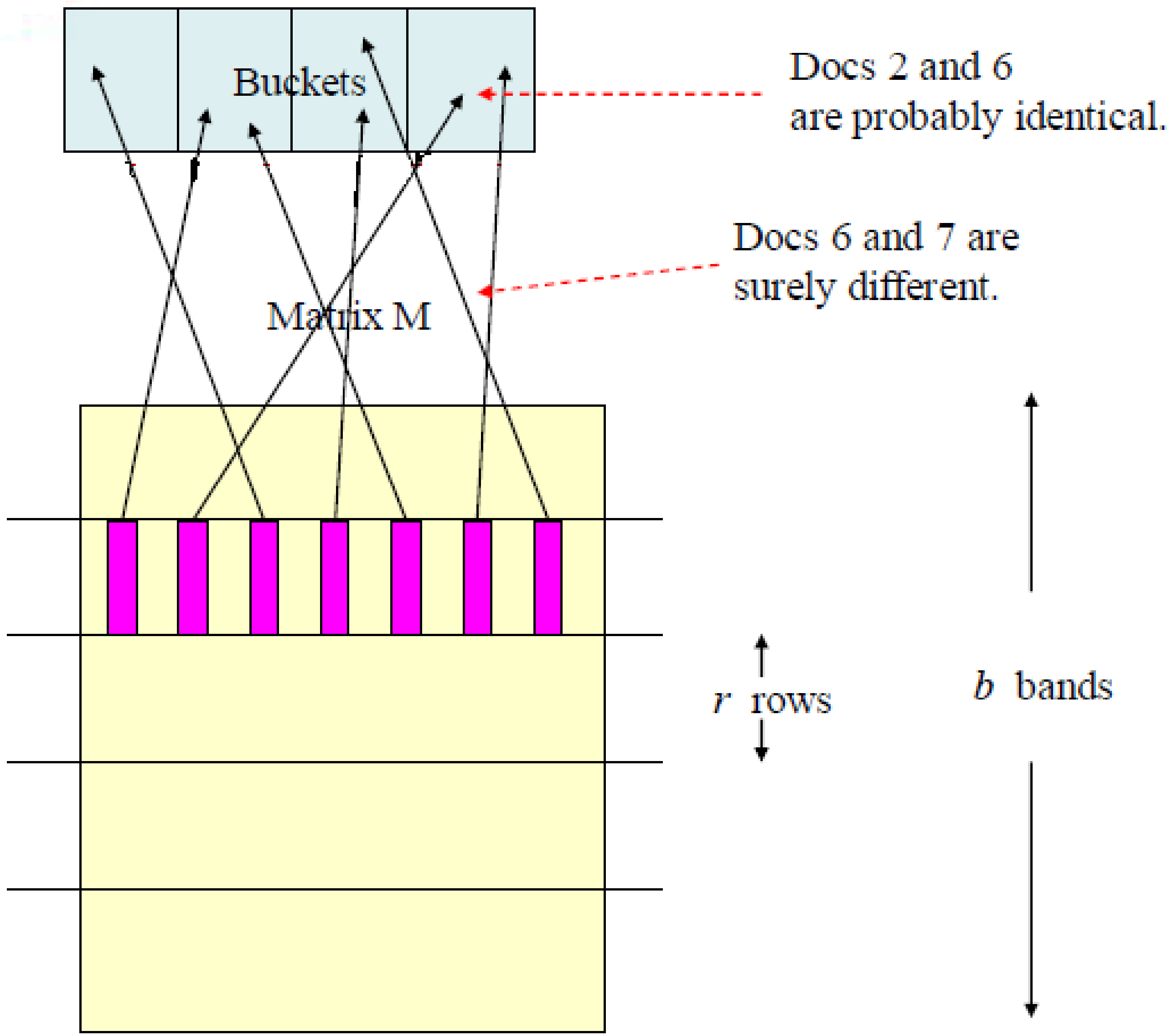


Signature Matrix

Signature agreement of each pair at each band



Signature Matrix


Buckets
Docs 2 and 6
are probably identical.
Docs 6 and 7 are
surely different.
Matrix M
r rows
b bands

Signature generation and bucket comparison

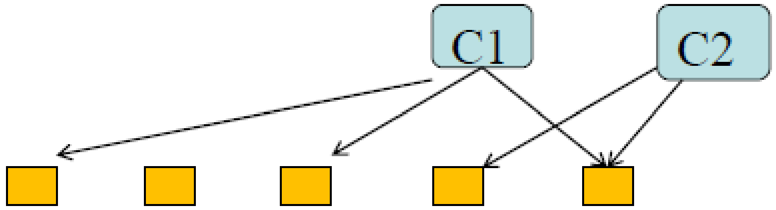
- Create *b* bands for each document
  - Signature of doc X and Y in the same band agrees → a candidate pair
  - Use *r* minhash values (*r* rows) for each band
- Tune *b* and *r* to catch most similar pairs, but few nonsimilar pairs.
  - Similar pairs hashed to the same bucket reduces the false positives
  - Dis-similar pairs hashed to different buckets reduces the false negatives

Analysis of Banding Technique

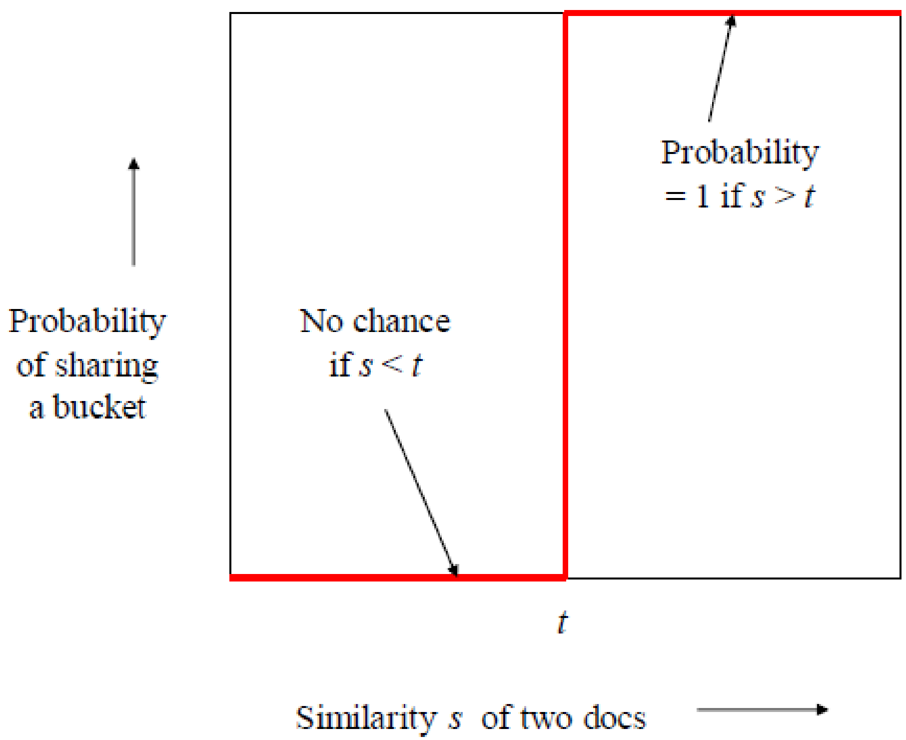
- Probability the minhash signatures of C_1, C_2 agree in one row: s
  - Threshold of two similar documents
- Probability C_1, C_2 identical in one band: s^r
- Probability C_1, C_2 do not agree at least one row of a band: $1\text{-}s^r$
- Probability C_1, C_2 do not agree in all bands: $(1\text{-}s^r)^b$
  - False negative probability
- Probability C_1, C_2 agree one of these bands: $1\text{-}(1\text{-}s^r)^b$
  - Probability that we find such a pair.

Example

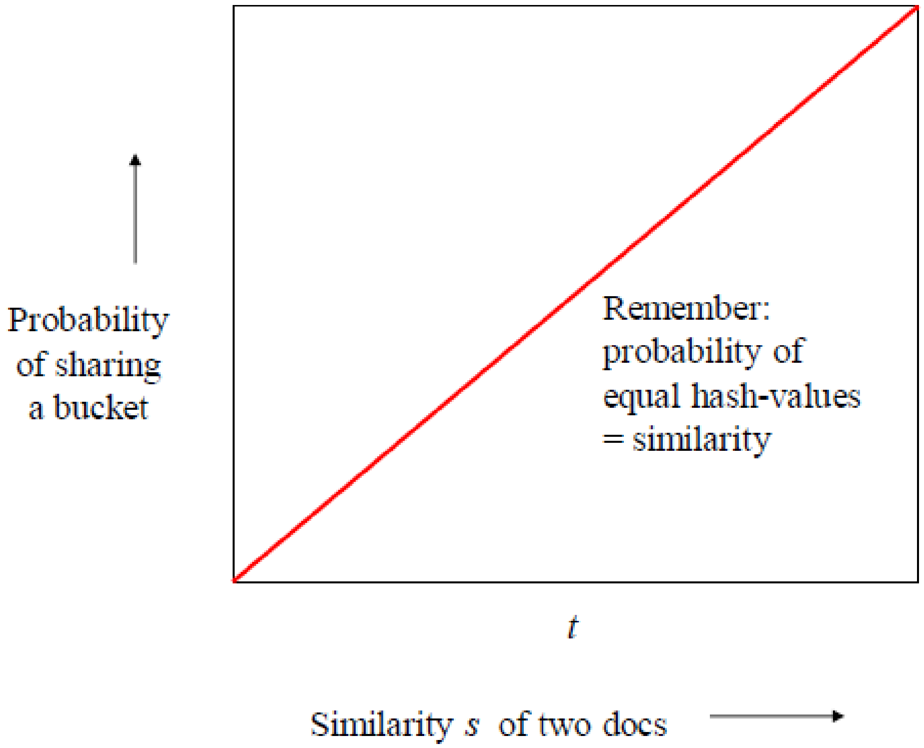
- Suppose C_1, C_2 are 80% Similar
- Choose 20 bands of 5 signatures/band.
- Probability C_1, C_2 identical in one particular band: $(0.8)^5 = 0.328$.
- Probability C_1, C_2 are *not* similar in any of the 20 bands: $(1-0.328)^{20} = .00035$.
    - i.e., about 1/3000th of the 80%-similar column pairs are false negatives.



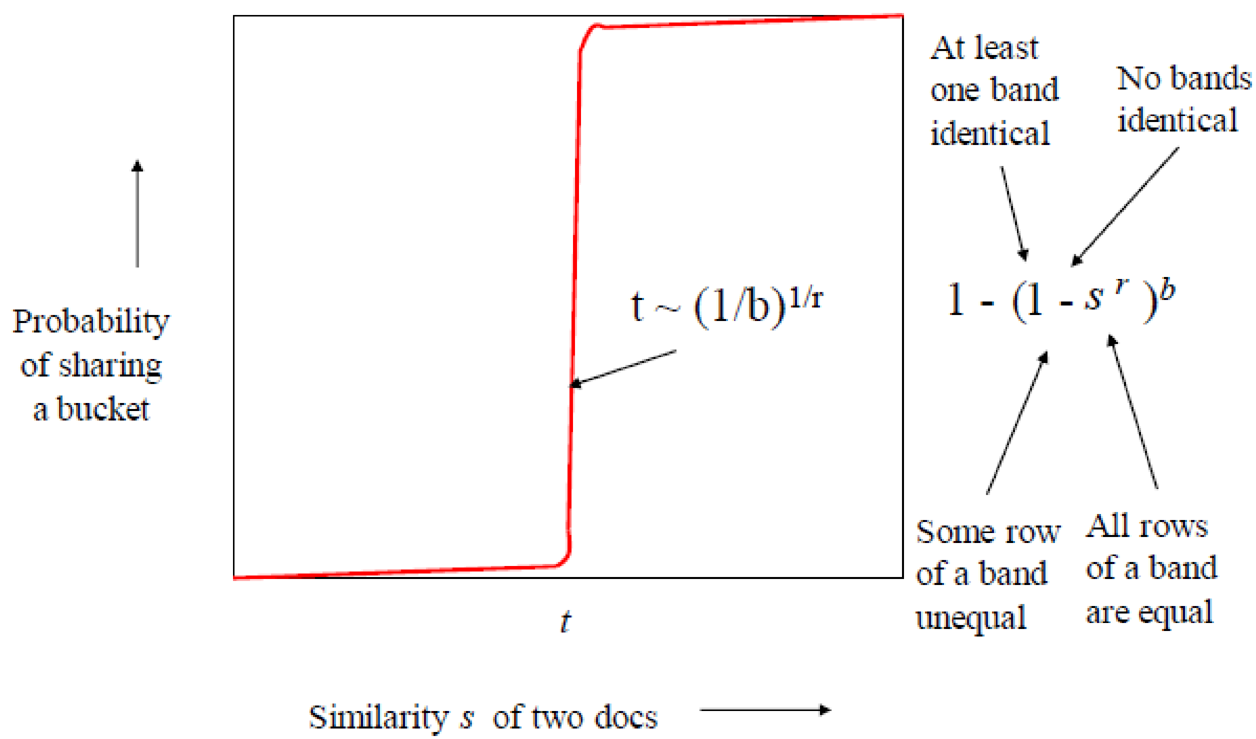
Analysis of LSH – What We Want



What One Band Gives You



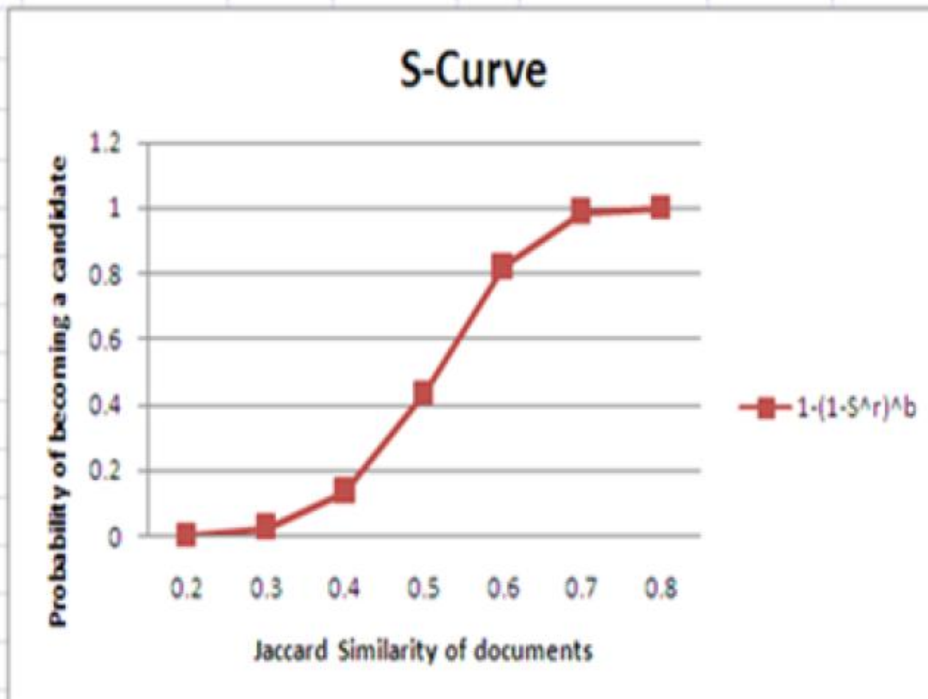
What *b* Bands of *r* Rows Gives You



Example: $b = 36$; $r = 6$

- Probability of a similar pair to share a bucket

s	1-(1-S^r)^b	Rows-Per Ban	Bands
0.2	0.0023	6	36
0.3	0.02591		
0.4	0.13736		
0.5	0.43274		
0.6	0.82094		
0.7	0.98896		
0.8	0.99998		



LSH Summary

- **Get almost all pairs with similar signatures, but eliminate most pairs that do not have similar signatures.**
  - Check that candidate pairs really do have similar signatures.
- LSH involves tradeoff
  - Pick the number of minhashes, the number of bands, and the number of rows per band to balance false positives/negatives.
  - Example: if we had only 15 bands of 5 rows, the number of false positives would go down, but the number of false negatives would go up.

Algorithm for Similarity Search

Input: Set of files

Output: Most similar files grouped together

Step1: Select all files

Step2: Convert text files into shingles set

Step3: Store shingles into bloom filters for each file, and union of all shingles set into a large array.

Step4: Create characteristic matrix

Step5: Create signature matrix

Step6: Divide signature matrix into of n rows into b band and in every band r rows

Step7: Hash the columns of signature matrix in a large bucket by using different hash function for different band Use of bloom filter for storing the shingles improves the search time

Any Question?