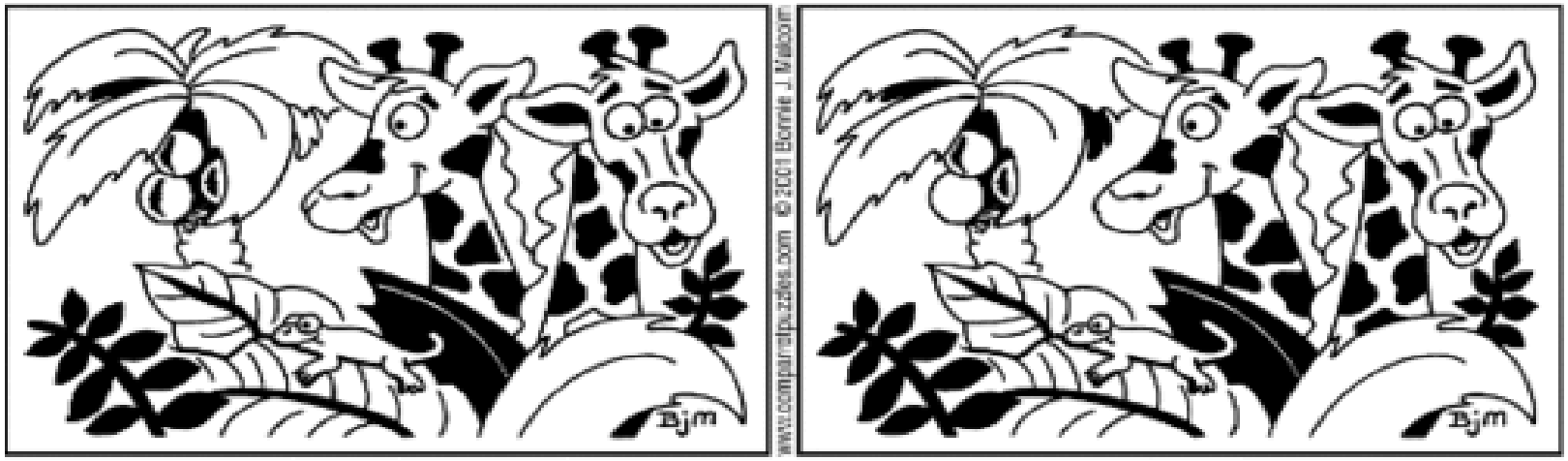
CS5344
Finding Similar Items



Motivation

- **Examine data for similar items**
- **Example: Customers who purchased similar products**
- **Express as finding sets with relatively large intersection (similar sets)**
- **Two variants of the problem**
  - [Offline] Extract all similar pairs of objects from a large collection
  - [Online] Is this object similar to something seen before?

Application: Plagiarism Detection

Vegetable Waste Treatment: Comparison and Critical Presentation of Methodologies

IOANNIS S. ARVANITOYANNIS and THEODOROS H. VARZAKAS
Critical Reviews in Food Science and Nutrition, 48:205–247 (2008)

INTRODUCTION

Fruit and vegetable wastes (FVW) are produced in large quantities in markets, and constitute a source of nuisance in municipal landfills because of their high biodegradability (Misi and Forster, 2002). In the central distribution market for food (meat, fish, fruit, and vegetables) Mercabarna (Barcelona), the total amount of wastes coming from fruit and vegetables is around 90 tons per day during 250 days per year (Mata-Alvarez et al., 1992). In India, FVW constitute about 5.6 million tons annually and currently these wastes are disposed of by dumping on the outskirts of cities (Srilatha et al., 1995). In Tunisia FVW are estimated to be 180 tons per month (Bouallagui et al., 2003).

According to Verrier et al. (1987) and Ruynal et al. (1998) the total initial solid concentration of FVW is between 8 and 18%, with a total volatile solids (VS) content of about 87% when anaerobic digestion was operated The organic fraction includes about 75% sugars and hemicellulose, 9% cellulose and 5% lignin. The easy biodegradable organic matter content of FVW (75%) with high moisture facilitates their biological treatment and shows the trend of these wastes for anaerobic digestion. However, complex vegetable processing effluent, such as olive mill wastes containing large amounts of phenolic and non-biodegradable compounds are resistant to biological degradation (Hamdi, 1996). Aerobic processes are not favored for FVW treatment because they require preliminary treatment to minimize the organic loading rate.

The aim of this work was to make a comparative and critical presentation of all vegetable waste treatment methods (both traditional and novel) in

Original Sources

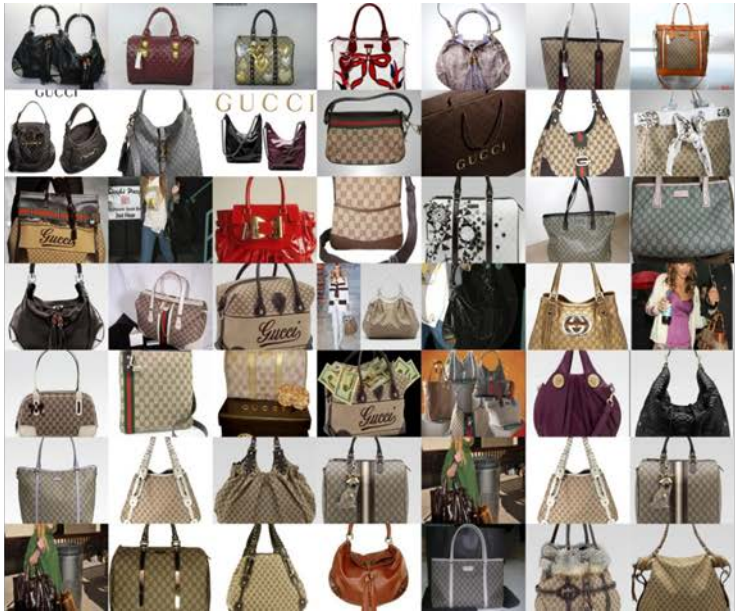
Bioreactor performance in anaerobic digestion of fruit and vegetable wastes
H. Bouallagui, Y. Touhami, R. Ben Cheikh and M. Hamdi
Process Biochemistry 40: 989-995 (2005)

Fruit and vegetable wastes (FVW) are produced in large quantities in markets, and constitute a source of nuisance in municipal landfills because of their high biodegradability [1 and 2]. In the central distribution market for food (meat, fish, fruit, and vegetables) Mercabarna (Barcelona), the total amount of wastes coming from fruit and vegetables is around 90 tonnes per day during 250 days per year [3]. The whole production of FVW collected from the market of Tunis (Tunisia) has been measured and estimated to be 180 tons per month [4]. In India, FVW constitute about 5.6 million tonnes annually and currently these wastes are disposed by dumping on the outskirts of cities [5].

The total initial solid concentration of FVW is between 8 and 18%, with a total volatile solids (VS) content of about 87%. The organic fraction includes about 75% sugars and hemicellulose, 9% cellulose and 5% lignin [20]. The easy biodegradable organic matter content of FVW (75%) with high moisture facilitates their biological treatment and shows the trend of these wastes for anaerobic digestion [1 and 21]. However, complex vegetable processing effluent, such as olive mill wastes containing large amounts of phenolic and non-biodegradable compounds are resistant to biological degradation [22]. Aerobic processes are not favoured for FVW treatment because they require preliminary treatment to minimise the organic loading rate [23].

- **Textual similarity - pages with similar words**
- **Plagiarizer may extract parts of a document for his own, alter a few words or order in which sentences of the original appear**

Application: Content-based Search



- **Images with similar features**

Distance Measures

- Find near-neighbors in high-dimensional space
- Define “near neighbors” as points that are a “small distance” apart
- Need to define what “distance” means for each application

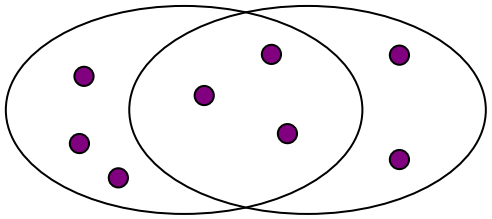
Distance Measures

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

$$sim\ (C_1, C_2) = |C_1 \cap C_2| \,/\, |C_1 \cup C_2|$$

- **Jaccard distance:**

$$d\ (C_1, C_2) = 1 - |C_1 \cap C_2| \,/\, |C_1 \cup C_2|$$



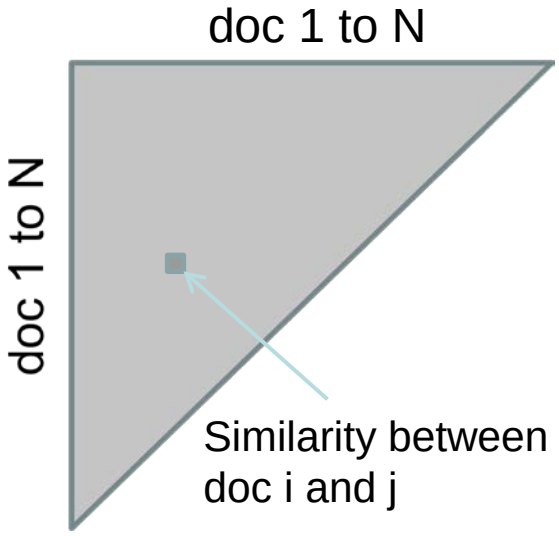
3 in intersection
8 in union
Jaccard similarity = 3/8
Jaccard distance = 5/8

Task: Find Similar Documents

- **Given a large number (N in the millions or billions) of documents, find "near duplicate" pairs**
- **Applications:**
  - Mirror websites or approximate mirrors
    - Don't want to show both in search results
  - Similar news articles at many news websites
    - Cluster articles by "same story"

Task: Find Similar Documents

- **Problems:**
  - Many small pieces of one document can appear out of order in another
  - Too many documents to compare all pairs
  - Documents are so large, or so many that they cannot fit in main memory



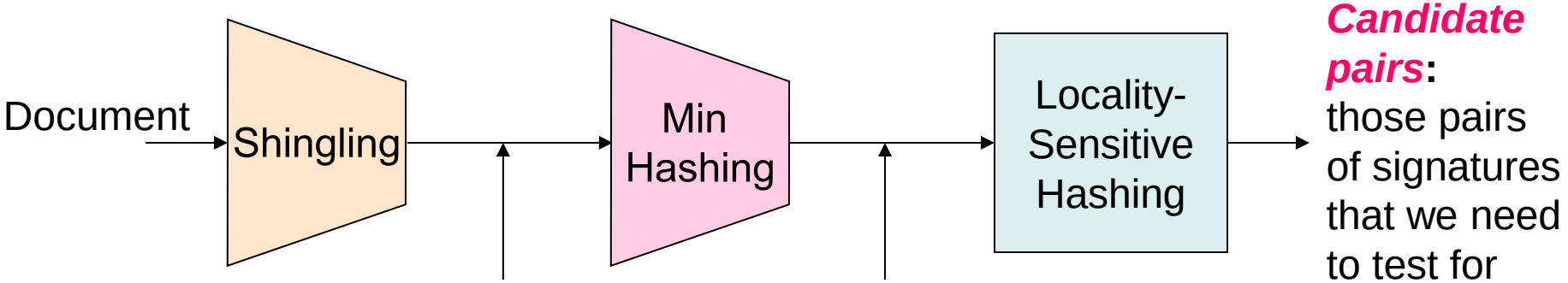
Task: Find Similar Documents

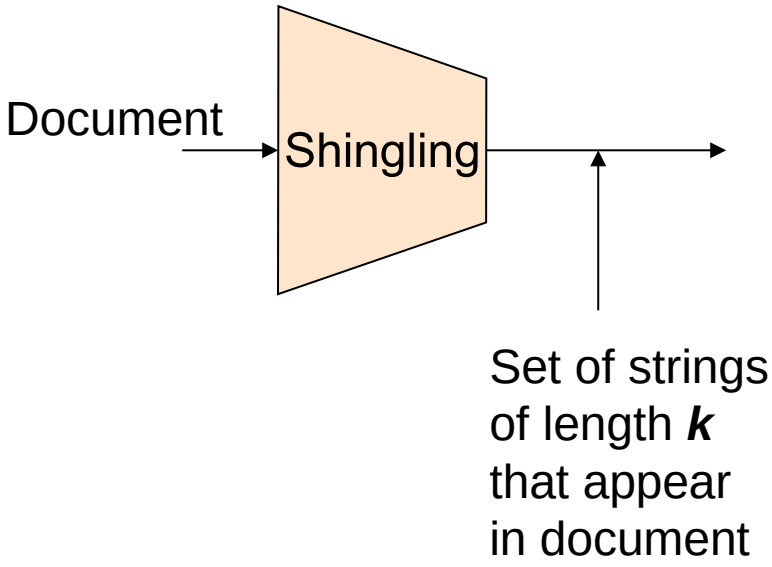
1. **Shingling - Convert documents to sets**

2. **Min-Hashing - Convert large sets to short signatures, while preserving similarity**

3. **Locality-Sensitive Hashing - Focus on pairs of signatures likely to be from similar documents**
   - **Candidate pairs!**

Overview


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



SHINGLING

Convert documents to sets

Shingling

- **Simple approaches to convert documents to sets**
  - Document = set of words appearing in document
  - Document = set of “important” words
  - Don’t work well for this application. **Why?**
- **Instead of treating each word independently, we can consider a sequence of k words (shingles)**
  - Account for ordering of words
  - More effective in terms of accuracy

Definition: Shingles

- **A *k*-shingle (or *k*-gram) for a document is a sequence of *k* tokens that appears in the document**
  - Tokens can be characters, words or some feature/object, depending on the application
- **Example: D = {This is a test}**
  - k = 3 characters
    Set of 3-shingles: S(D) = {Thi, his, is_, s_i, _is, is_, s_a, _a_, a_t, _te, tes, est}
  - k = 3 words, we have S(D) = {{This is a}, {is a test}}
- **Variation: Shingles as a bag (multiset)**
  - count a shingle as many times as it occurs in document

Shingle Size

- **Documents that have lots of shingles in common have similar text, even if the text appears in different order**

- **Caveat: You must pick *k* large enough, or most documents will have most shingles**
  - Recommended values

  k = 5 for short documents e.g. emails

  k = 10 for long documents e.g. research articles

Compressing Shingles

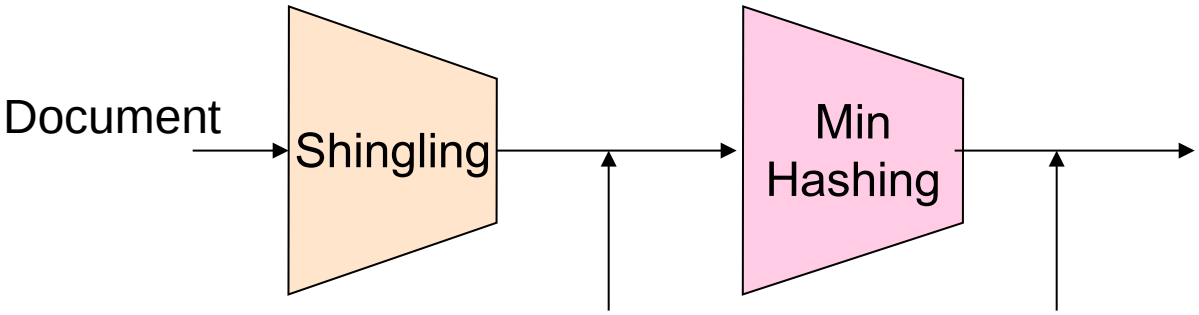
- **Instead of using substrings directly as shingles, we can hash them to (say) 4 bytes**
  - Use a hash function to map strings of length k to some number of buckets
  - Treat resulting bucket number as the shingle
- **Represent a document by the set of hash values of its *k*-shingles**
- **Example**

  k=2, Document D_1= abcab
  Set of 2-shingles: $S(D_1)$ = {ab, bc, ca}
  Hash the shingles: $h(D_1)$ = {1, 5, 7}

Motivation for Minhash/LSH

- **Represent a document as a set of k-shingles or its hash values**
  - Space requirement is still huge
- **Find near-duplicate documents among $N = 1$ million documents**
  - Compute **pairwise Jaccard similarities** for **every pair of documents**
  - **$N(N-1)/2$ ≈ 5*10^{11}** comparisons would take **5 days** at 10^5 secs/day and 10^6 comparisons/sec

Find a way to hash a document to a *single* (small size) value, and *similar* documents to the *same* value



Set of strings of length *k* that appear in document

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

MIN-HASHING

Convert large sets to short signatures, while preserving similarity

Minhash

- **Seminal algorithm for near-duplicate detection of webpages**
  - Used by AltaVista
- **Hash the *set* of document shingles (big in terms of space requirement) into a *signature* (relatively small size)**
  - Compare signatures instead of shingles

MinHash

- **Important: Similarities of signatures and similarities of shingles MUST BE related**
  - Not every hashing function is applicable
  - Need one that satisfies the following:
    - if $sim(D_1,D_2)$ is high, then with high prob. $h(D_1) = h(D_2)$
    - if $sim(D_1,D_2)$ is low, then with high prob. $h(D_1) \neq h(D_2)$
  - Possible to have false positives, and false negatives!
- **Minhashing turns out to be one such function for Jaccard similarity**

Encode Sets as Bit Vectors

- **Many similarity problems can be formalized as finding subsets with significant intersection**
- **Encode sets using 0/1 (bit, boolean) vectors**
  - One dimension per element in the universal set
- **Interpret set intersection as bitwise AND, and set union as bitwise OR**

- **Example: C_1 = 10111, C_2 = 10011**
  - Size of intersection **= 3**
  - Size of union **= 4**
  - **Jaccard similarity** (not distance) **= 3/4**
  - **Distance: $d(C_1,C_2)$ = 1 – (Jaccard similarity) = 1/4**

From Sets to Boolean Matrices

- **Rows = elements (shingles)**
- **Columns = sets (documents)**
  - **Each document is a column**
  - 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 corresponding sets (rows with value *1)*
- **Example: sim(C_1 ,C_2) = ?**
  - Size of intersection = 3; size of union = 7 Jaccard similarity (not distance) = 3/7
  - **d(C_1,C_2) = 1 – (Jaccard similarity) = 4/7**

Documents

Shingles

1	1	1	0
1	1	0	1
0	1	0	1
0	1	0	1
1	0	0	1
1	1	1	0
1	0	1	0

From Sets to Boolean Matrices

Sets:

A = {e_1, e_3, e_7}

B = {e_3, e_5, e_7}

Can be equivalently expressed as matrices:

Shingle	A	B
e_1	1	0
e_2	0	0
e_3	1	1
e_4	0	0
e_5	0	1
e_6	0	0
e_7	1	1

Let:

M_{00} = # rows where both elements are 0

M_{11} = # rows where both elements are 1

M_{01} = # rows where A=0, B=1

M_{10} = # rows where A=1, B=0

$$\mathrm{J}(A,B) = \frac{M_{11}}{M_{01} + M_{10} + M_{11}}$$

Minhashing

- **Start with the matrix representation of the set**
- **Randomly permute the rows of the matrix**
- **Minhash value of any column is the first row (in the permuted order) with a “1”**

Input matrix

Row	Shingle	A	B
1	e_1	1	0
2	e_2	0	0
3	e_3	1	1
4	e_4	0	0
5	e_5	0	1
6	e_6	0	0
7	e_7	1	1

Permuted matrix

Shingle	A	B
e_6	0	0
e_2	0	0
e_5	0	1
e_3	1	1
e_7	1	1
e_4	0	0
e_1	1	0

h(A) = 4

h(B) = 3

Minhash and Jaccard Similarity

- **Probability that minhash function for a random permutation of rows produces the same value for two sets = Jaccard similarity of the sets**

$$P[h(A) = h(B)] = \mathrm{J}(A, B) \qquad \frac{M_{11}}{M_{01} + M_{10} + M_{11}}$$

Shingle	A	B
e_6	0	0
e_2	0	0
e_5	0	1
e_3	1	1
e_7	1	1
e_4	0	0
e_1	1	0

M_{00} = # rows where both elements are 0

M_{11} = # rows where both elements are 1

M_{01} = # rows where A=0, B=1

M_{10} = # rows where A=1, B=0

The Min-Hash Property

0	0
0	0
1	**1**
0	0
0	1
1	0

Choose a random permutation π

Claim: $\Pr[h_\pi(C_1) = h_\pi(C_2)] = sim(C_1, C_2)$

Why?

Let **X** be a doc (set of shingles), $y \in X$ is a shingle

Then $\Pr[\pi(y) = \min(\pi(X))] = 1/|X|$

Equally likely that any $y \in X$ is mapped to the ***min*** element

Let ***y*** be such that $\pi(y) = \min(\pi(C_1 \cup C_2))$

Then either: $\pi(y) = \min(\pi(C_1))$ if $y \in C_1$ **or**

$\pi(y) = \min(\pi(C_2))$ if $y \in C_2$

*One of the two cols had to have 1 at position **y***

So probability that **both** are true is the probability $y \in C_1 \cap C_2$

$\Pr[\min(\pi(C_1))=\min(\pi(C_2))] = |C_1 \cap C_2|/|C_1 \cup C_2| = sim(C_1, C_2)$

MinHash – False positive/negative

- **Instead of comparing sets, we now compare only 1 hash value!**
- **False positive?**
  - False positive can be easily dealt with by doing an additional layer of checking (treat minhash as a filtering mechanism)
- **False negative?**
- **High error rate! Can we do better?**
  - Yes. Generalize to multiple hash functions

Minhash Signatures

Comparison between two sets (original matrix) becomes comparison between two columns of minhash values (signature matrix)

Similarity between two signatures is the fraction of hash values in which they agree

Input

1	0	1	0
1	0	0	1
0	1	0	1
0	1	0	1
0	1	0	1
1	0	1	0
1	0	1	0

Permutations

3	4	1
4	2	3
7	1	7
6	3	6
1	6	2
2	7	5
5	5	4

Minhash signatures

2	1	2	1

	1-3	2-4	1-2
Col/Col	0.75	0.75	0
Sig/Sig	0.67	1.00	0

Similarities

Implementation of MinHash

- **Permutations are expensive**
  - Incur space and random access (if data cannot fit into memory)
- **Simulate the effect of a random permutation**
  - Use a random hash function that maps row numbers to as many buckets as there are rows
  - Only need to keep track of minimum hash values

2	1
3	1

Row	C1	C2
1	1	0
2	0	1
3	1	1
4	1	0
5	0	1

2	4
3	1
4	3
5	5
1	2

$h(x) = x \bmod 5 + 1$

$g(x) = (2x+1) \bmod 5 + 1$

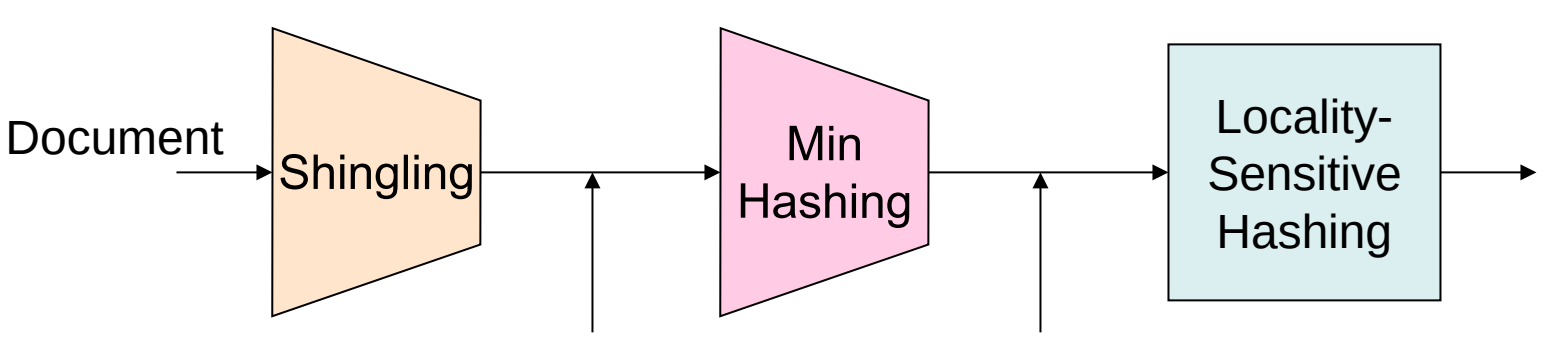
Implementation of Minhash

Row	C_1	C_2
1	1	0
2	0	1
3	1	1
4	1	0
5	0	1

$h(x) = x \bmod 5 + 1$
$g(x) = (2x+1) \bmod 5 + 1$

	Sig1	Sig2
h(1) = 2	2	∞
g(1) = 4	4	∞
h(2) = 3	2	3
g(2) = 1	4	1
h(3) = 4	2	3
g(3) = 3	3	1
h(4) = 5	2	3
g(4) = 5	3	1
h(5) = 1	2	1
g(5) = 2	3	1

- **Initialization: set signatures to ∞**
- **Apply all hash functions on each row**
- **If column value (of the source matrix) is 1, keep the minimum value**
- **Otherwise, do nothing**



Candidate pairs: those pairs of signatures that we need to test for similarity

Set of strings of length *k* that appear In document

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

Locality Sensitive Hashing

Focus on pairs of signatures likely to be from similar documents

Locality Sensitive Hashing (LSH)

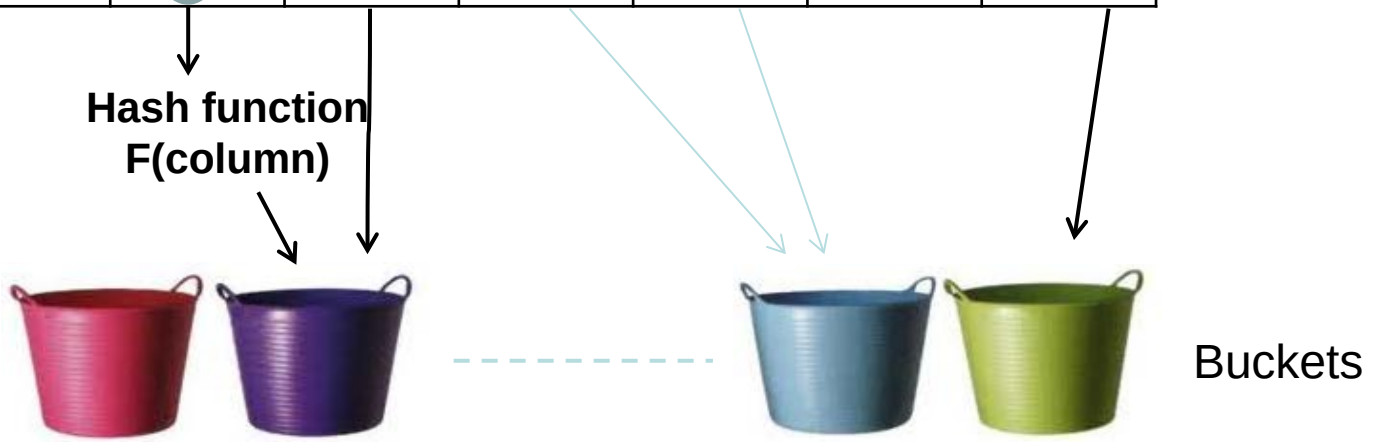
- **Given N documents, derive k minhash signatures for each document**

Minhash Signature	D_1	D_2	D_3	D_4	….	D_N
1	3	3	2	2		3
2	7	7	5	5		7
	⋮	⋮				
k-1	2	2	2	2		2
k	1	1	1	1		2

Locality Sensitive Hashing (LSH)

- **Hash columns of signature matrix to buckets**

minhash	D_1	D_2	D_3	D_4	….	D_N
1	3	3	2	2		3
2	7	7	5	5		7
k-1	2	2	2	2		2
k	1	1	1	1		1



Locality Sensitive Hashing (LSH)

- **Documents that fall into the same buckets are likely to be similar – candidate pair**
- **Finding all pairs within a bucket is computationally cheaper**
  - Declare all pairs within a bucket to be “matching” OR
  - Perform pair-wise comparisons for those documents that fall into the same bucket
    - Much smaller than pair-wise over all documents

LSH for Minhash Signatures

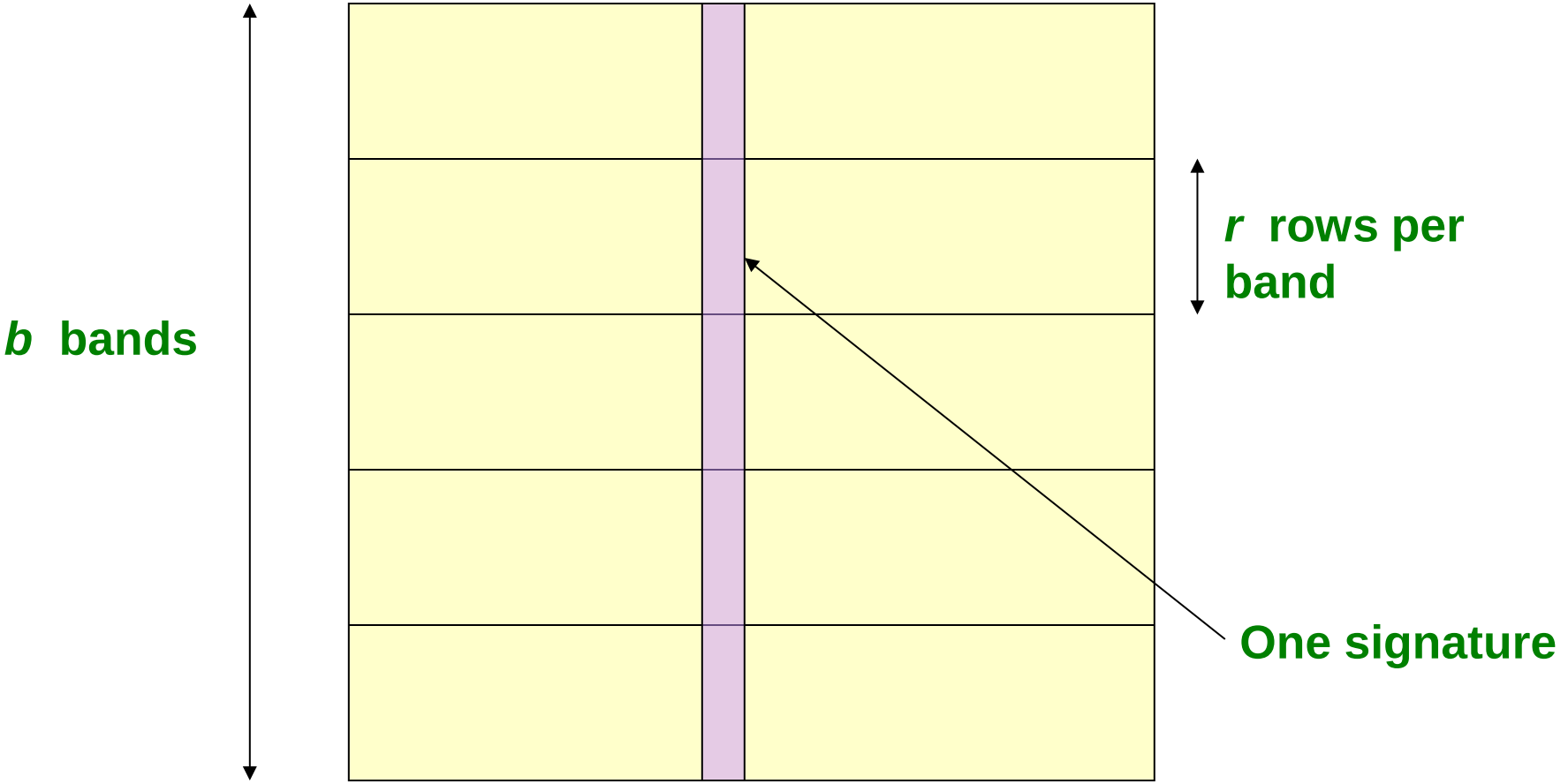
- **More false positive (as a result of LSH) possible**
  - Why? Because of collision (subject to hash function and number of buckets), need a refinement step
- **Many false negatives likely**
  - Why? Because only one hash function on an entire column of signature
- **Apply hash function on the column multiple times, each on a partition of the column**
  - Similar columns will be hashed to the same bucket (with high probability)
  - Candidate pairs are those that hash at least once to the same bucket

LSH for Minhash – Intuition

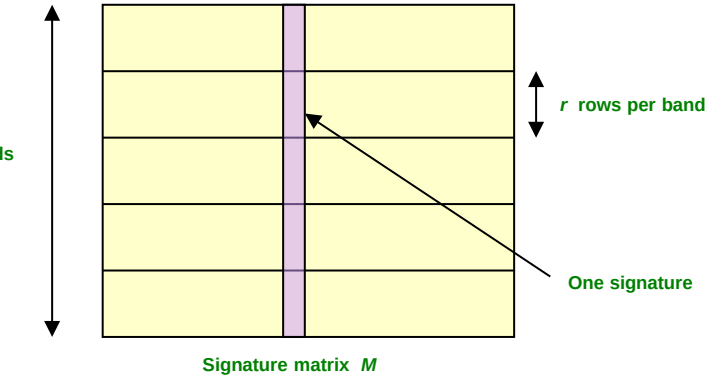
C1	C2
3	3
6	1
9	9
4	4
2	2
8	8
4	4
8	3
1	1
1	1
4	4
2	2

- 12 minhash functions – likely to be similar documents since they agree on 10 out of 12 minhash values
- Hash on all 12 minhash values → likely to be in different buckets → not similar
- Partition 12 minhash values into two - still not similar
- Partition 12 minhash values into four - potentially similar (>= 50% similar)

Partition *M* into *b* Bands

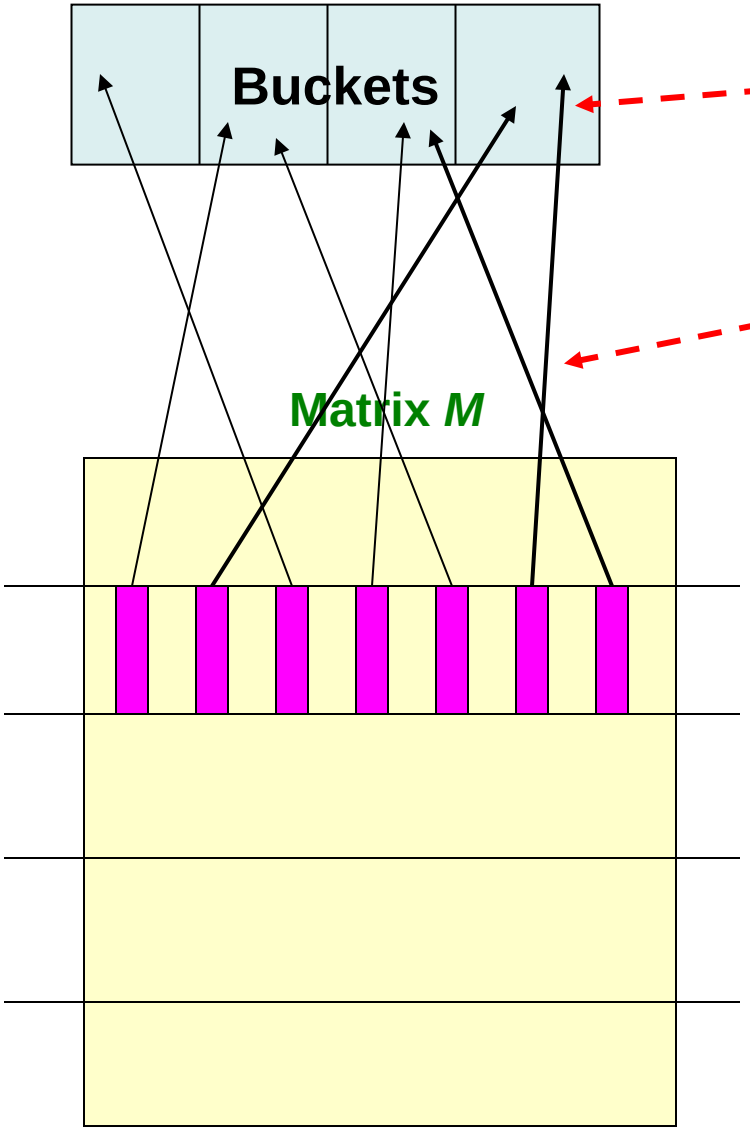


Partition *M* into *b* Bands



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

Hashing Bands



Columns 2 and 6 are probably identical (candidate pair)

Columns 6 and 7 are surely different.

r **rows**

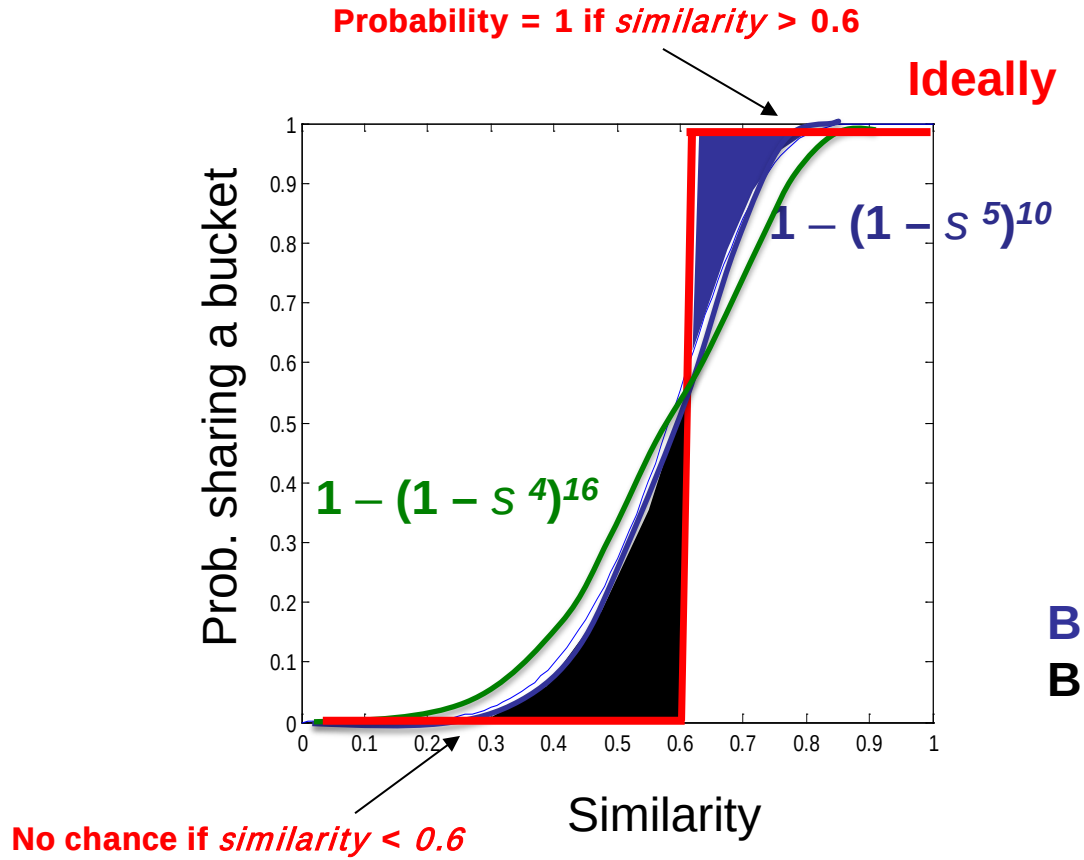
b **bands**

Analysis of LSH

- ***b* bands, *r* rows per band**
- **Columns C_1 and C_2 have Jaccard similarity *s***
  - Prob. that they have same minhash value
- **Pick any band (*r* rows)**
  - Prob. that all rows in band equal = s^r
  - Prob. that some row in band unequal = $1 - s^r$
- **Prob. that no band identical = $(1 - s^r)^b$**
- **Prob. that at least 1 band identical = $1 - (1 - s^r)^b$**
- ***How to choose r and b?***

Picking *r* and *b*: The S-curve

- **Function $1 - (1 - s^{r})^{b}$ is a S-curve regardless of r and b**
- **Minimize false positives and false negatives**



Blue area: False Negative
Black area: False Positive

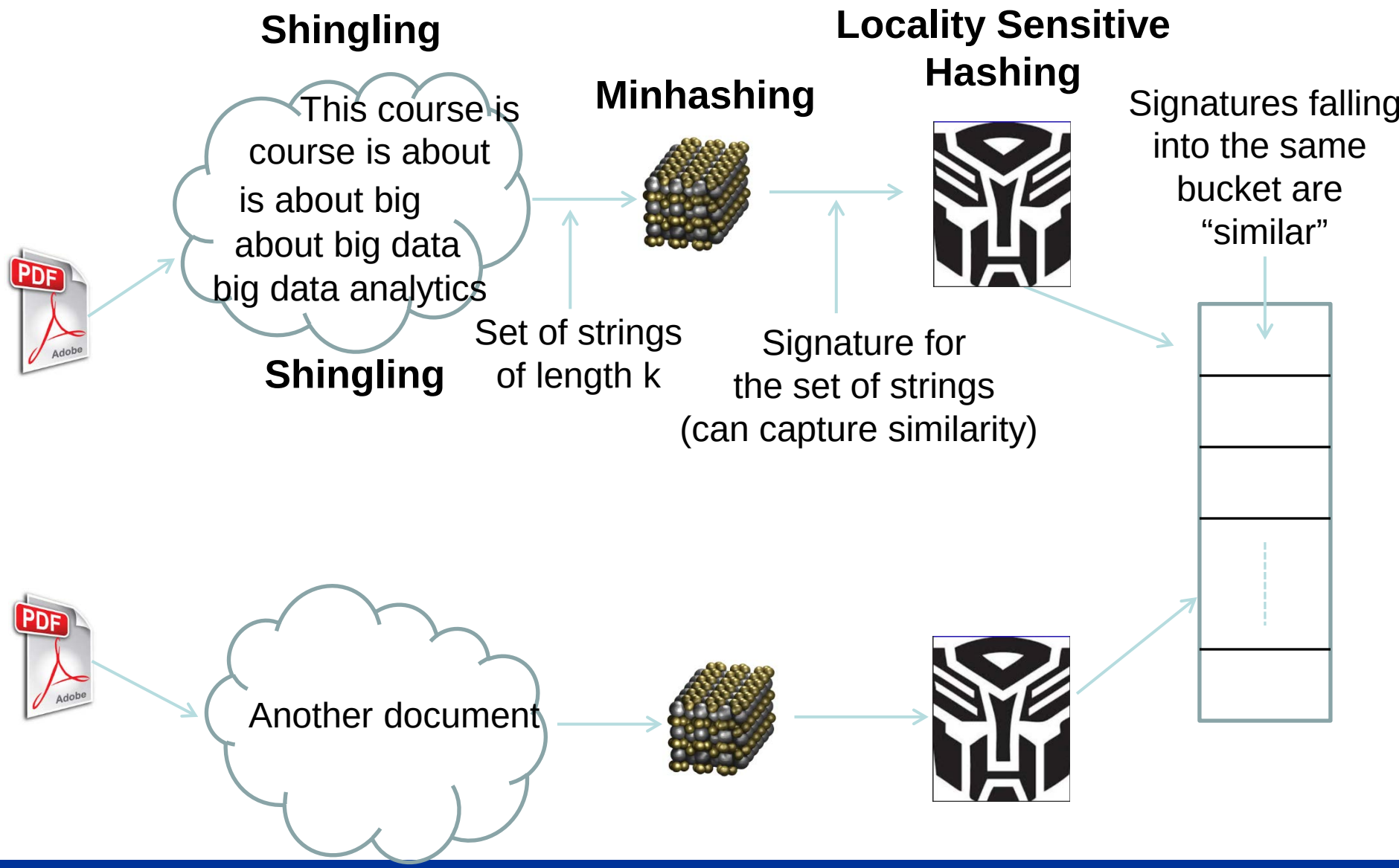
LSH Summary

- **Tune *b and r* to get almost all pairs with similar signatures, but eliminate most pairs that do not have similar signatures**
- **Need to check that candidate pairs have similar signatures**
- **Need further refinement to find really similar documents**

Combining the Techniques

- **Shingling: Convert documents to sets**
  - Use hashing to assign each shingle an ID
- **Min-Hashing: Convert large sets to short signatures, while preserving similarity**
  - Use **similarity preserving hashing** to generate signatures with property $\mathbf{Pr[}h_\pi(C_1) = h_\pi(C_2)\mathbf{]} = sim(C_1, C_2)$
  - Use hashing to get around generating random permutations
- **Locality-Sensitive Hashing: Focus on pairs of signatures likely to be from similar documents**
  - Use hashing to find **candidate pairs**

Summary


Shingling
Locality Sensitive Hashing
Minhashing
This course is
course is about
is about big
about big data
big data analytics
Signatures falling into the same bucket are "similar"
Shingling
Set of strings of length k
Signature for the set of strings (can capture similarity)
PDF
Adobe
Another document