Processamento e Recuperação de Informação

Efficient Similarity Search

Departamento de Engenharia Informática
Instituto Superior Técnico

1º Semestre
2018/2019

Outline

1. Finding Similar Items
2. Shingles
3. Minhashing
4. Locality-sensitive hashing

Bibliography

Jure Leskovec, Anand Rajaraman, and Jeff Ullman, Mining of Massive Datasets, Chapter 3

High Dimensional Data

Many real-world problems

- Web Search and Text Mining
  - Billions of documents, millions of terms
- Product Recommendations
  - Millions of customers, millions of products
- Scene Completion, other graphics problems
  - Image features
- Online Advertising, Behavioral Analysis
  - Customer actions (e.g., websites visited, searches, ...)

A Common Metaphor

Many problems can be expressed as finding similar sets.

Find near-neighbors in high-dimensional space.

Examples:

- Pages with similar words
  - For duplicate detection, classification by topic
- Customers who purchased similar products
  - NetFlix users with similar tastes in movies
- Products with similar customer sets
- Images with similar features
- Users who visited the similar websites

Distance Measures

We formally define near neighbors as points that are a small distance apart.

For each use case, we need to define what distance means.

Two major classes of distance measures:

- A Euclidean distance is based on the locations of points in such a space
- A Non-Euclidean distance is based on properties of points, but not their location in a space
  - Cosine similarity, Jaccard similarity coefficient, ...

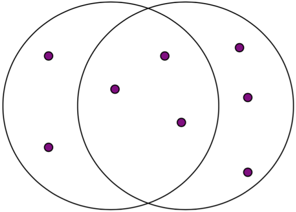
Jaccard Similarity

The Jaccard Similarity of two sets is the size of their intersection over the size of their union.

$$Sim(C_1, C_2) = \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|}$$

The Jaccard Distance between sets is 1 minus their Jaccard similarity.

$$d(C_1, C_2) = 1 - \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|}$$



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

Outline

1 Finding Similar Items

2 Shingles

3 Minhashing

4 Locality-sensitive hashing

Finding Similar Documents

Goal:

Given a large number (N in the millions or billions) of text documents, find pairs that are near duplicates

Applications:

- Mirror websites, or approximate mirrors
  - Don't want to show both in a search
- Similar news articles at many news sites
  - Cluster articles by same story

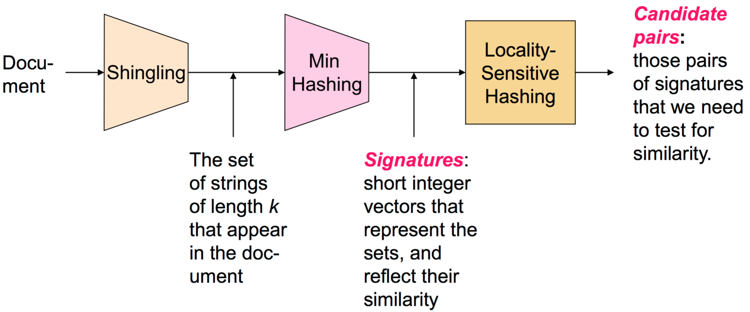
Problems:

- 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 memory

Three Essencial Steps

1. Shingling: Convert documents, emails, etc., to sets;
2. Minhashing: Convert large sets to short signatures, while preserving similarity;
   - Depends on the distance metric;
3. Locality-sensitive hashing: Focus on pairs of signatures likely to be from similar documents.

The Big Picture



Outline

Processamento e Recuperação de Informação

Finding Similar Items

Shingles

Minhashing

Locality-sensitive hashing

1. Finding Similar Items
2. Shingles
3. Minhashing
4. Locality-sensitive hashing

Documents as High Dimensional Data

Step 1:
Shingling: Convert documents, emails, etc., to sets

- Simple approaches...
  - Document = set of words appearing in document
  - Document = set of *important* words
- ...don't work well for this application!
  - Need to account for ordering of words
- A different way: Shingles

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 something else, depending on application
  - Assume tokens = characters for next examples

Example: $k = 2$; $D_1 = abcab$

Set of 2-shingles: $S(D_1) = \{ab, bc, ca\}$

Option: Shingles as a bag (i.e., multi-set), counting ab twice

- Represent a doc by the set of hash values of its k-shingles

Compressing Shingles

- To compress long shingles, we can hash them into a convenient/efficient representation (e.g. 4 bytes)
- Represent a doc by the set of hash values of its k-shingles
- Idea: Two docs could (rarely) appear to have shingles in common, when in fact only the hash-values were shared

Example: $k = 2$; $D_1 = abcab$

Set of 2-shingles: $S(D_1) = \{ab, bc, ca\}$

Hash the shingles: $h(D_1) = \{1, 5, 7\}$

Similarity Metric for Shingles

- Document D_1 = set of k-shingles $C_1 = S(D_1)$
- Equivalently, each document is a $0/1$ vector in the space of k-shingles
  - Each unique shingle is a dimension
  - Vectors are very sparse
- A natural similarity measure is the Jaccard similarity:

$$Sim(C_1, C_2) = \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|}$$

Working Assumption

- Documents that have lots of shingles in common have similar text, even if the text appears in different order
- Careful: You must pick k large enough, or most documents will have most shingles
  - $k = 5$ is OK for short documents
  - $k = 10$ is better for long documents

Motivation for Minhash/LSH

Processamento e Recuperação de Informação

Finding Similar Items

Shingles

Minhashing

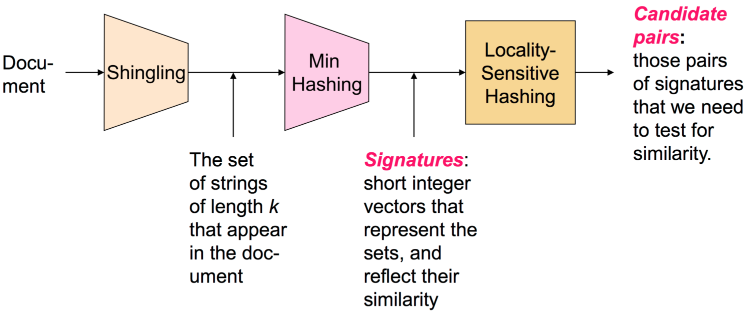
Locality-sensitive hashing

- Suppose we need to find near-duplicate documents among a large collection with $N = 1$ million documents
- Naively, we'd have to compute pairwise Jaccard similarites for every pair of documents
  - i.e, $\frac{N \times (N-1)}{2} \approx 5 \times 10^{11}$ comparisons
  - At 10^5 secs/day and 10^6 comparisons/sec, it would take 5 days to compute all pairwaise Jaccard similarites
- For $N = 10$ million, it takes more than a year...

Outline

1. Finding Similar Items
2. Shingles
3. Minhashing
4. Locality-sensitive hashing

The Big Picture



Encoding Sets as Bit Vectors

Many similarity problems can be formalized as finding subsets hat have 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) $= 3/4$
- $d(C_1, C_2) = 1 -$ (Jaccard similarity) $= 1/4$

From Sets to Boolean Matrices

Finding Similar Items

Shingles

Minhashing

Locality-sensitive hashing

- 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

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

Jaccard of Columns

Each document is a column:

Example: $C1 = 1100011$; $C2 = 0110010$

- Size of intersection $= 2$;
- size of union $= 5$,
- Jaccard similarity $= 2/5$
- $d(C_1, C_2) = 1 -$ (Jaccard similarity) $= 3/5$

We might not really represent the data by a Boolean matrix

- Sparse matrices can be represented by the list of places with non-zero values

shingles

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

documents

Finding Similar Columns

So far:

- Documents represented as sets of shingles
- Represent sets as boolean vectors in a matrix

Next Goal: Find similar columns

1. Signatures of columns: small summaries of columns
2. Examine pairs of signatures to find similar columns
   - Essential that similarities of signatures and columns are related
3. Optional: check that columns with similar signatures are really similar

Warnings:

- Comparing all pairs may take too much time: job for LSH
- These methods can produce false negatives, and even false positives (if the optional check is not made)

Signatures of Columns

Key idea:

Hash each column C to a small signature $h(C)$, such that:

1. $h(C)$ is small enough that the signature(s) fit(s) in RAM
2. $sim(C_1, C_2)$ is the same as the similarity of signatures $h(C_1)$ and $h(C_2)$

Goal:

Find a hash function $h()$ such that:

- if $sim(C_1, C_2)$ is high, then with high prob. $h(C_1) = h(C_2)$
- if $sim(C_1, C_2)$ is low, then with high prob. $h(C_1) \neq h(C_2)$

Hash documents into buckets, and expect that most pairs of near duplicate documents hash into the same bucket

Min-Hashing (1)

Goal:

Find a hash function $h()$ such that:

- if $sim(C_1, C_2)$ is high, then with high prob. $h(C_1) = h(C_2)$
- if $sim(C_1, C_2)$ is low, then with high prob. $h(C_1) \neq h(C_2)$

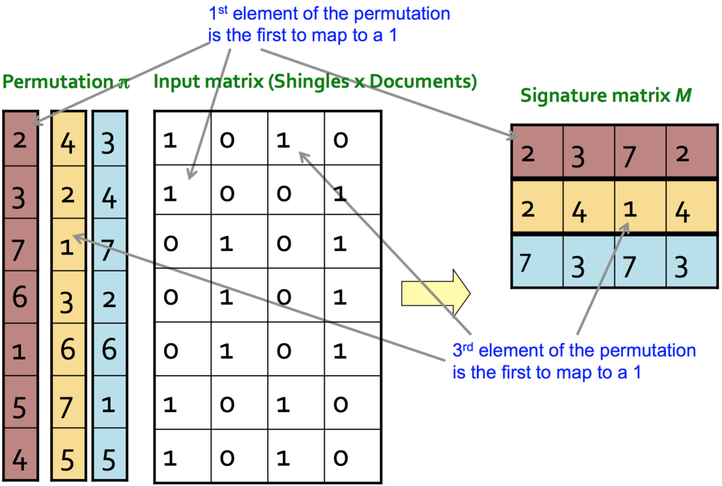
Clearly, the hash function depends on the similarity metric:

- Not all similarity metrics have a suitable hash function
- There is a suitable hash function for Jaccard similarity: Min-hashing

Min-Hashing (2)

- Imagine the rows of the boolean matrix permuted under random permutation π
- Define a hash function $h_\pi(C)$ = the number of the first (in the permuted order π) row in which column C has value 1:
$$h_\pi(C) = \min_\pi(C)$$
- Use several (e.g., 100) independent hash functions (i.e., permutations) to create a signature of a column
- Implementation trick
  - Ordering under hash function gives a random row permutation
  - For each column C and hash-function π keep a *slot* for the min-hash value

Min-Hashing (3)



Surprising Property

Under a random permutation π,
$Pr[h_\pi(C_1) = h_\pi(C_2)] = sim(C_1, C_2)$

Sketch of proof:

- Let X be a set of shingles, and let $x \in X$
- Then: $Pr[\pi(y) = \min(\pi(X))] = \frac{1}{|X|}$
  - It is equally likely that any $y \in X$ is mapped to the min element
- Let x be s.t. $\pi(x) = \min(\pi(C_1 \cup C_2))$
  - Then either $\pi(x) = \min(\pi(C_1))$ if $x \in C_1$, or $\pi(x) = \min(\pi(C_2))$ if $x \in C2$
  - So the probability that both are true is the probability $x \in C_1 \cap C_2$
- $Pr[\min(\pi(C_1)) = \min(\pi(C_2))] = \frac{|C1 \cap C2|}{|C1 \cup C2|} = sim(C_1, C_2)$

Similarity for Signatures

- We know $Pr[h_\pi(C_1) = h_\pi(C_2)] = sim(C_1, C_2)$
- Now generalize to multiple hash functions
- The similarity of two signatures is the fraction of the hash functions in which they agree
- Note: Because of the minhash property, the similarity of columns is the same as the expected similarity of their signatures

Example

Permutation π

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

Input matrix (Shingles x Documents)

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

Signature matrix *M*

2	3	7	2
2	4	1	4
7	3	7	3

Similarities:

	1-3	2-4	1-2	3-4
Col/Col	0.75	0.75	0	0
Sig/Sig	0.33	0.67	0	0

Minhash Signatures

- Pick 100 random permutations of the rows
- Think of $sig(C)$ as a column vector
- Let $sig(C)[i] =$ according to the i-th permutation, the index of the first row that has a 1 in column C

$$sig(C)[i] = \min(\pi_i(C))$$

- Note: The sketch (signature) of document C is small – $\tilde{1}00$ bytes!
- We achieved the goal of compressing long bit vectors into short signatures

Outline

1. Finding Similar Items
2. Shingles
3. Minhashing
4. Locality-sensitive hashing

The Big Picture

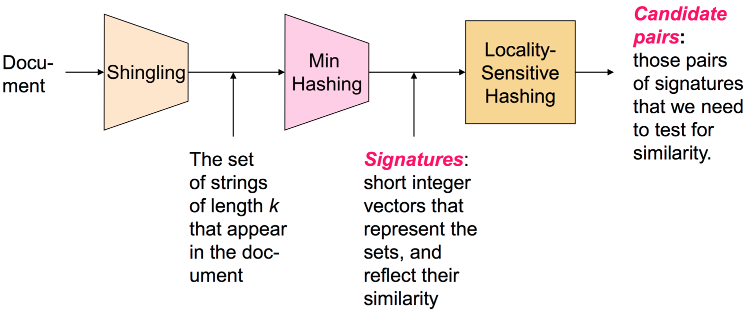
Processamento e Recuperação de Informação

Finding Similar Items

Shingles

Minhashing

Locality-sensitive hashing



LSH : General Intuition

Goal:

Find documents with Jaccard similarity at least s (for some similarity threshold, e.g., $s = 0.8$)

Locality-Sensitive Hashing (LSH)

- Use a function $f(x, y)$ that tells whether x and y is a candidate pair, i.e. a pair of elements whose similarity must be evaluated
- For minhash matrices:
  - Hash columns of signature matrix M to many buckets
  - Each pair of documents that hashes into the same bucket is a candidate pair

Candidates from Minhash

- Pick a similarity threshold $0 < s < 1$
- Columns x and y of M are a candidate pair if their signatures agree on at least fraction s of their rows:

$$M(i, x) = M(i, y) \text{ for at least fraction } s \text{ values of } i$$

- We expect documents x and y to have the same similarity as their signatures

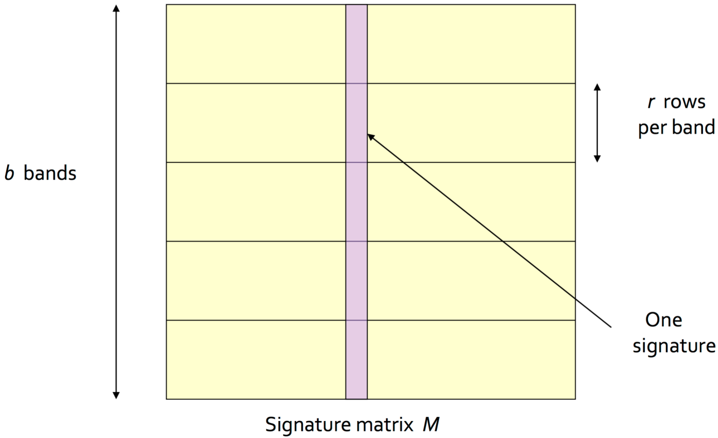
LSH for Minhash Signatures

- Big idea: Hash columns of signature matrix M several times
- Arrange that (only) similar columns are likely to hash to the same bucket, with high probability
- Candidate pairs are those that hash to the same bucket

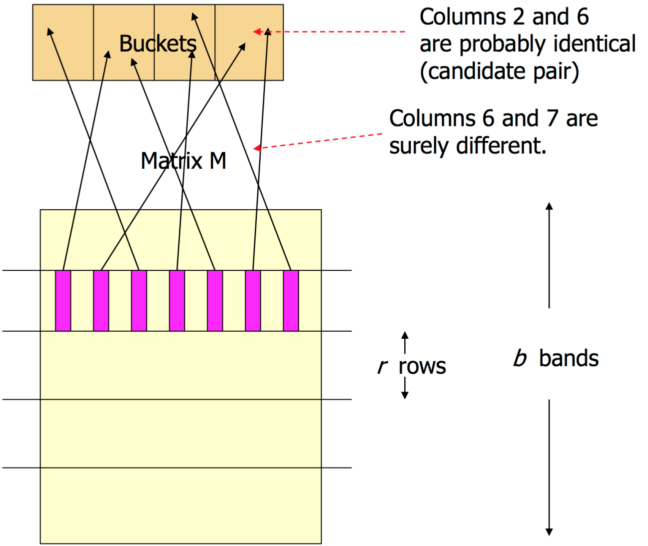
Partition M into Bands (1)

- 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 or more
- Tune b and r to catch most similar pairs, but few non-similar pairs

Partition M into Bands (2)



Hashing 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 of Bands (1)

Assume the following case:

- Suppose 100,000 columns of M (100k docs)
- Signatures of 100 integers (rows)
- Therefore, signatures take 40Mb
- Choose 20 bands of 5 integers/band

Goal: Find pairs of documents that are at least $s = 80\%$ similar

Example of Bands (2)

Assume: C_1, C_2 are 80% similar

- Since s=80% we want C_1, C_2 to hash to at least one common bucket (at least one band is identical)
- Probability C_1, C_2 identical in one particular band: $(0.8)^5 = 0.328$
- Probability C_1, C_2 are not similar in all of the 20 bands: $(1 - 0.328)^{20} = 0.00035$
    - i.e., about 1/3000th of the 80%-similar column pairs are false negatives
    - We would find 99.965% pairs of truly similar documents

Example of Bands (3)

Assume: C_1, C_2 are 30% similar

- Since s=80% we want C_1, C_2 to hash to at NO common buckets (all bands should be different)
- Probability C_1, C_2 identical in one particular band: $(0.3)^5 = 0.00243$
- Probability C_1, C_2 identical in at least 1 of 20 bands: $1 - (1 - 0.00243)^{20} = 0.0474$
  - In other words, approximately 4.74% pairs of docs with similarity 30% end up becoming candidate pairs – false positives

LSH Involves a Tradeoff

Pick parameters to balance false positives/negatives:

- number of minhashes (rows of M)
- number of bands b, and
- number of rows r per band

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

Analysis of LSH - What we want

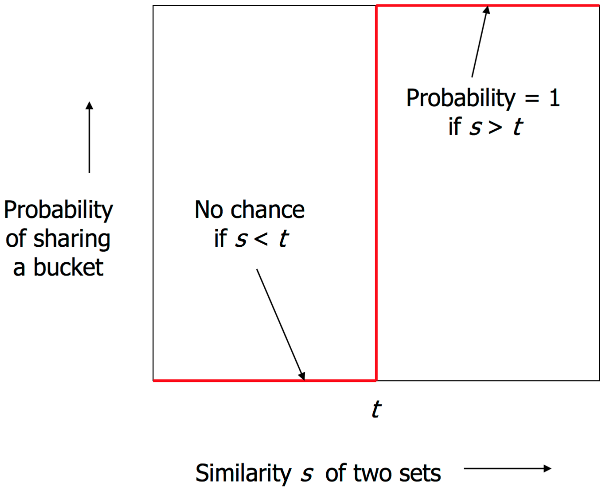
Processamento e Recuperação de Informação

Finding Similar Items

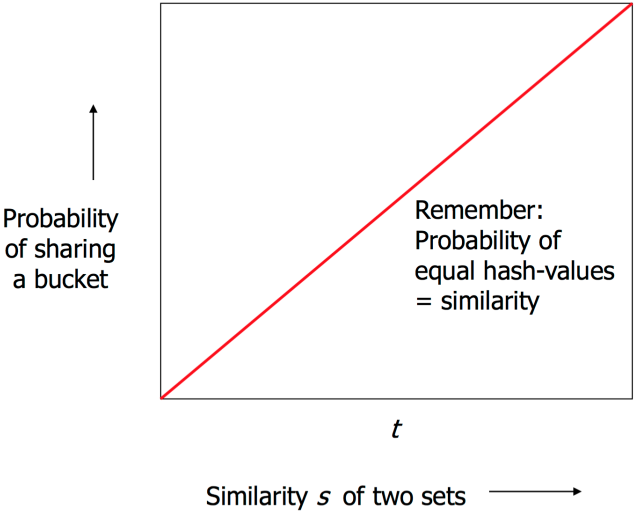
Shingles

Minhashing

Locality-sensitive hashing



Analysis of LSH - One band with one row



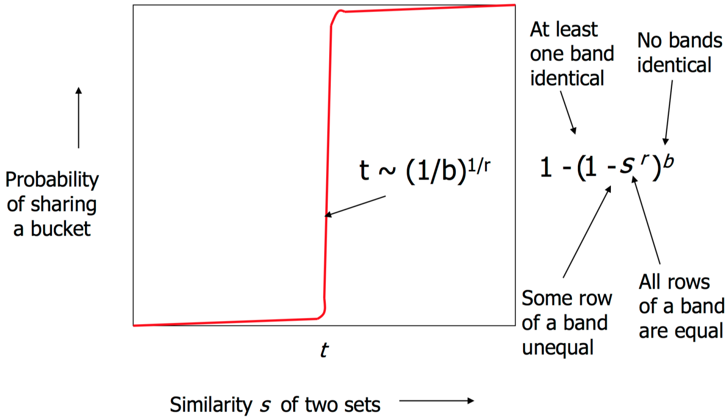
Analysis of LSH - b bands with b rows/band (1)

Columns C_1 and C_2 have similarity s

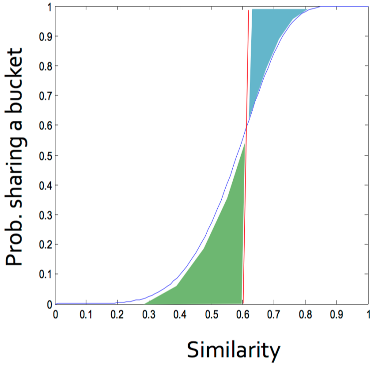
Pick any band (r rows)

- Probability that all rows in band equal $= s^r$
- Probability that some row in band unequal $= 1 - s^r$
- Probability that no band identical $= (1 - s^r)^b$
- Probability that at least 1 band identical $= 1 - (1 - s^r)^b$

Analysis of LSH - b bands With r rows/band (2)



False Positives vs. False Negatives

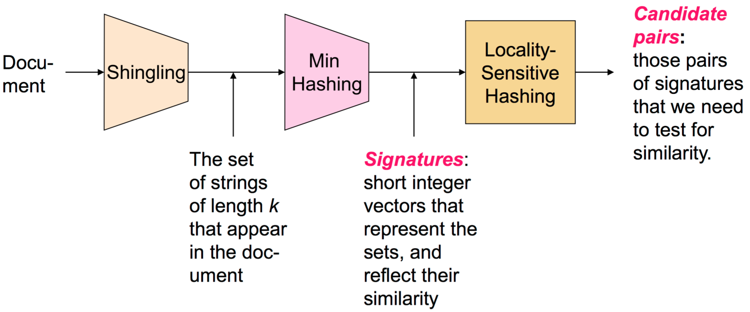


Blue area: False Negative rate
Green area: False Positive rate

LSH Summary

- Tune to get almost all pairs with similar signatures, but eliminate most pairs that do not have similar signatures
- Check in main memory that candidate pairs really do have similar signatures
- *Optional:* In another pass through data, check that the remaining candidate pairs really represent similar documents

The Big Picture



Questions?