Near duplicate detection

Giorgio Gambosi

Course of Information Retrieval
CdLM in Computer Science
University of Rome Tor Vergata

Derived from slides produced by C. Manning and by H. Schütze

Applications of NDD

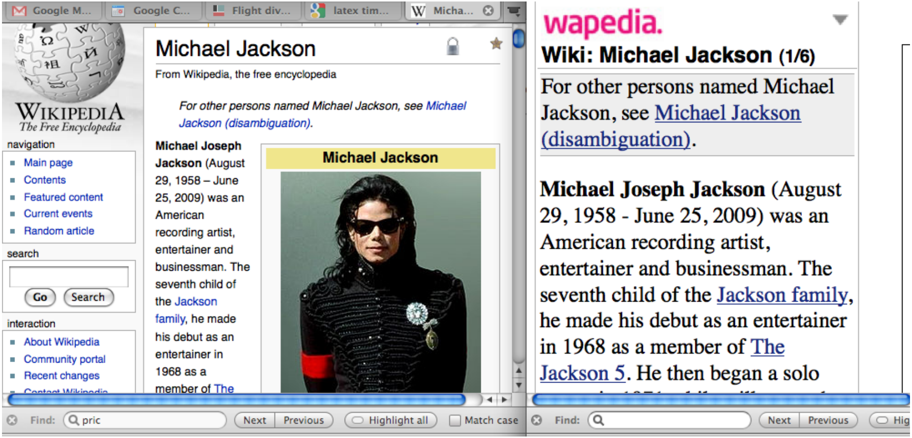
Many problems in data mining can be seen as searching in sets of similar items:

- Pages with similar words, for classification on topics.
- Topic suggestion to Twitter users with similar profiles (recommendation systems).
- Dual problem: identifying communities of users with similar interests
- Identifying same user in different contexts (e.g. social media platforms)

On the web

- The web is full of duplicated content.
- More so than many other collections
- Exact duplicates
  - Easy to eliminate
  - E.g., use hash/fingerprint
- Near-duplicates
  - Abundant on the web
  - Difficult to eliminate
- For the user, it's annoying to get a search result with near-identical documents.
- Marginal relevance is zero: even a highly relevant document becomes nonrelevant if it appears below a (near-)duplicate.
- We need to eliminate near-duplicates.

Near-duplicates: Example



Similar documents

Finding sets of documents (web pages) with much text in common:

- Mirror or quasi-mirror sites
  - Application: elimination of duplicates.
- Plagiarism, inclusion of extensive citations .
- Articles with similar content in different news sites .
  - Application: grouping articles as a “common history”.

Detecting near-duplicates

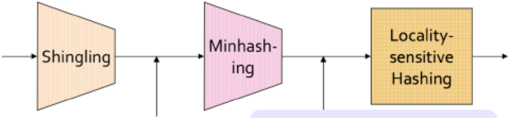
- Compute similarity with an edit-distance measure
- We want "syntactic" (as opposed to semantic) similarity.
    - True semantic similarity (similarity in content) is too difficult to compute.
- We do not consider documents near-duplicates if they have the same content, but express it with different words.
- Use similarity threshold θ to make the call "is/isn't a near-duplicate".
- E.g., two documents are near-duplicates if similarity $> \theta = 80\%$.

Three techniques useful for NDD

- Shingling: convert documents, e-mail, ecc, in sets of items.
- Minhashing: convert large sets in short sketches (or signatures), preserving similarity.
- Locality Sensitive Hashing (LSH): consider pairs of signature that could be similar with at least a given probability.

Architecture

Document



Sketch pairs to test for similarity

Shingles: Word sequences of length *k* occurring in the document

Sketches: short vectors of integers representing shingles, and preserving their similarity

Represent each document as set of shingles

Shingles are used as features to measure syntactic similarity of documents.

- A shingle is just a word k-gram.
- A document is represented as a set of shingles
- For $n = 5$, "*In a hole in the ground there lived a hobbit*" would be represented as this set of shingles:
  - {In a hole in the, a hole in the ground, hole in the ground there, in the ground there lived, the ground there lived a, ground there lived a hobbit }
- Similar documents will have many shingles in common

Represent each document as set of shingles

- Modifying a word affects only k shingles (the ones at distance at most k from the word)
- Moving a paragraph affects $2k$ shingles (the ones at distance at most k from the paragraph borders)
- For $n = 3$, changing "*In a hole in the ground there lived a hobbit*" to "*In a hole in the ground there was a hobbit*" only changes shingles $\{$ ground there lived, there lived a, lived a hobbit$\}$

Documents as sets of shingles

- In general, different documents should have few shingles in common, especially for higher k
- We define the similarity of two documents as the Jaccard coefficient of their shingle sets.

Recall: Jaccard coefficient

- A commonly used measure of overlap of two sets
- Let A and B be two sets: their Jaccard coefficient is defined as:

  $$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

  $(A \neq \emptyset$ or $B \neq \emptyset)$
- $J(A, A) = 1$
- $J(A, B) = 0$ if $A \cap B = 0$
- A and B don't have to be the same size.
- Always assigns a number between 0 and 1.

Jaccard coefficient: Example

- Three documents:
  d_1: "Jack London traveled to Oakland"
  d_2: "Jack London traveled to the city of Oakland"
  d_3: "Jack traveled from Oakland to London"
- Based on shingles of size 2 (2-grams or bigrams), what are the Jaccard coefficients $J(d_1, d_2)$ and $J(d_1, d_3)$?
- 1. $s(d_1)$={"Jack London", "London traveled", "traveled to", "to Oakland"}
  2. $s(d_2)$={"Jack London", "London traveled", "traveled to", "to the", "the city", "city of", "of Oakland"}
  3. $s(d_3)$={"Jack traveled", "traveled from", "from Oakland", "Oakland to", "to London'}
- there are
- $J(d_1, d_2) = 3/8 = 0.375$
- $J(d_1, d_3) = J(d_2, d_3) = 0$

Represent each document as a sketch

- The number of shingles per document is large: computing Jaccard directly from M is expensive
- To increase efficiency, we will represent documents by means of sketches, cleverly chosen subsets of their shingles.
- Let k be a predefined sketch size and let S be the overall set of shingles: document sketches are derived by means of a set of k different random permutations $\pi_1 \ldots \pi_k$ of S
- Each π_i maps a shingle to a different integer in $\{1, \ldots, |S|\}$
- The sketch of a document d is defined as:

$$\left(\min_{s \in d} \pi_1(s), \min_{s \in d} \pi_2(s), \ldots, \min_{s \in d} \pi_k(s)\right)$$

(a vector of s integers).

From sets of documents+shingles to boolean matrices

A set of documents can be represented as a boolean matrix M, where

- columns are associated to documents
- rows correspond to all shingles appearing in any document
- $M(i, j) = 1$ iff the i-th shingle appear in the j-th document
- The matrix is usually sparse

The Jaccard similarity of two documents can be derived from the corresponding columns

Four types of rows

- For any pair of columns S_1, S_2, rows can be classified in four types according to the values of the corresponding values in the matrix: each type has a different effect on numerator N and denominator D of $J(S_1, S_2)$

	S_1	S_2	effect on N	effect on D
a	1	1	increase	increase
b	1	0	same	increase
c	0	1	same	increase
d	0	0	same	same

- In fact, $J(S_1, S_2) = \dfrac{\#a}{\#a + \#b + \#c}$
- Many rows are of type d

Minhashing

Permutations of shingles correspond here to permutations of rows of M. The above considerations can be accordingly translated as follows.

- Given a row permutation π, for any document d corresponding to a column c_i in M, let us define as the Minhash of d under permutation π, denoted as $\mathrm{MH}_\pi(d)$ the index j of the first row (according to π) such that $M(j,i)=1$.
- As an extension, given a set Π_k of k permutations, for any document d corresponding to a column c_i in M, $\mathrm{MH}_{\Pi_k}(d)$ is defined as the vector of integers $(j_1,\ldots,j_k)$ such that j_r is the index of the first row (according to permutation π_r) such that $M(j_r,i)=1$.

Minhashing

- The sketch vector $\mathrm{MH}_{\Pi_k}(d)$ can be interpreted as a signature of d
- Signatures can be visualized as columns in a new matrix M', where columns correspond to documents while rows correspond to hash functions. The values in column c_i are then defined as $\mathrm{MH}_{\Pi_k}(d_i)$, where d_i is the document corresponding to c_i

Minhashing example

Shingle/document matrix M

d_1	d_2	d_3	d_4
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

Minhashing example

Permutations

1
3
7
6
2
5
4

M

d_1	d_2	d_3	d_4
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'

S_1	S_2	S_3	S_4
1	2	1	2

Minhashing example

Permutations

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

M

d_1	d_2	d_3	d_4
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'

S_1	S_2	S_3	S_4
2	1	4	1
1	2	1	2

Minhashing example

Permutations

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

M

d_1	d_2	d_3	d_4
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'

S_1	S_2	S_3	S_4
2	1	2	1
2	1	4	1
1	2	1	2

Detecting near-duplicates from sketches

Assume a single permutation π. Check is performed as follows:

- If $\text{MH}_{\pi(d_1)} = \text{MH}_{\pi(d_2)}$ then d_1 and d_2 probably are near-duplicates.
- If $\text{MH}_{\pi(d_1)} \neq \text{MH}_{\pi(d_2)}$ then d_1 and d_2 are probably not near-duplicates.

Detecting near-duplicates from sketches

Why does it work? Let us estimate the probability that, by randomly choosing π, we get $\mathrm{MH}_\pi(d_1) = \mathrm{MH}_\pi(d_2)$.

- $\mathrm{MH}_\pi(d_1)$ can be, with equal probability, any shingle occurring in d_1 (that is, each item in c_1 with value 1, they are $\#a + \#b$); the same for $\mathrm{MH}_\pi(d_2)$ (that is, any item in c_2 with value 1, they are $\#a + \#c$)
- the number of possible pairs $(\mathrm{MH}_\pi(d_1), \mathrm{MH}_\pi(d_2))$ (that is of pairs of rows with values 1 in c_1 and c_2) is $(\#a + \#b)(\#a + \#c) - \#b\#c$
- the number of possible pairs with $\mathrm{MH}_\pi(d_1) = \mathrm{MH}_\pi(d_2)$ (that is of rows with values 1 both in c_1 and in c_2) is $\#a^2$
- the probability that $\mathrm{MH}_\pi(d_1) = \mathrm{MH}_\pi(d_2)$ is then given by

$$p_h(d_1, d_2) = \frac{\#a^2}{(\#a + \#b)(\#a + \#c) - \#b\#c} = \frac{\#a}{\#a + \#b + \#c}$$

Detecting near-duplicates from sketches

- But

$$\frac{\#a^2}{(\#a+\#b)(\#a+\#c)-\#b\#c}=\frac{\#a}{\#a+\#b+\#c}$$

  is the Jaccard coefficient $J(d_1,d_2)$, that is our similarity measure between d_1 and d_2. So, estimating $p_\pi(d_1,d_2)$ corresponds to estimating the similarity between d_1 and d_2
- How can we get a good estimate of $p_\pi(d_1,d_2)$ more efficiently than computing $J(d_1,d_2)$ (which implies taking into account all their shingles?)

Detecting near-duplicates from sketches

- Observe that $p_\pi(d_1, d_2)$ is independent from the particular hash function h applied (our only requirement is that h induces a permutation of the matrix rows, which we assume true with high probability): by randomly selecting h and applying it to (d_1, d_2) we know that the probability that the event $\text{MH}_\pi(d_1) = \text{MH}_\pi(d_2)$ occurs is $p_\pi(d_1, d_2) = p(d_1, d_2)$.
- Selecting π and observing whether $\text{MH}_\pi(d_1) = \text{MH}_\pi(d_2)$ can be seen as sampling a stone from an urn containing $\#a$ red stones and $\#b + \#c$ black stones and checking whether the sampled stone is red

Detecting near-duplicates from sketches

- Performing a random sample of k independent permutations $\pi_1, \ldots, \pi_k$ and observing whether $\mathrm{MH}_{\pi_i}(d_1) = \mathrm{MH}_{\pi_i}(d_2)$ for each π_i corresponds to sampling k stones from the urn (with replacement) and checking how many sampled stoned are red
- This is a sequence of Bernoulli trials with probability $p(d_1, d_2)$. In this case, the number of red stones (functions such that $\mathrm{MH}_{\pi_i}(d_1) = \mathrm{MH}_{\pi_i}(d_2)$) is distributed according to a binomial distribution

$$p(\mathrm{MH}_{\pi_i}(d_1) = \mathrm{MH}_{\pi_i}(d_2) \text{ for exactly } r \text{ functions}) = \binom{r}{k} p(d_1, d_2)^r (1 - p(d_1, d_2))^{k-r}$$

which has mean $kp(d_1, d_2)$

Detecting near-duplicates from sketches

- $J(d_1, d_2)$ can be estimated by estimating $p(d_1, d_2)$ from the sample of size k provided by the set functions Π_k.
- by standard statistics, an unbiased estimator of p is $\hat{p} = \frac{r}{k}$, where r is the number of functions $h \in \Pi_k$ such that $\mathrm{MH}_\pi(d_1) = \mathrm{MH}_\pi(d_2)$
- the corresponding standard error is given by the sample standard deviation $\hat{s} = \sqrt{\frac{\hat{p}(1-\hat{p})}{k}}$: this makes it possible to a define confidence interval on $J(d_1, d_2)$ at any given confidence level θ as $[\hat{p} - Z_\theta \hat{s}, \hat{p} + Z_\theta \hat{s}]$, where Z_θ is the Z-score at probability θ (number of standard deviation from the man of a gaussian such that the tail probability is $1 - \theta$)
- the precision of the estimation improves as k increases

Random hash functions as permutations

Sketches can be efficiently computed by means of random hash functions.

- We can map shingles in S to integers by fingerprinting, that is by applying a given hash function h which maps any sequence of unigrams to a sequence of (say) m bytes, that is to an integer interval $0..2^m - 1$
- For suitably large m, with high probability there is no collision between pairs of shingles in S, that is $h(s_1) \neq h(s_2)$ for all $s_1, s_2 \in S$.
- Then, for suitably large m, h defines a permutation of shingles with high probability

Implementing Minhashing

- Let k be the number of hash functions.
- To each column d_j (document) and function h_i, a slot $s_{i,j}$ is associated.
- Iteratively compute, for each $r = 0, \dots$ up to the number of rows minus 1, all values $h_i(r)$
- At the end of the k-th iteration, $s_{i,j}$ stores the minimum value $\min r$, for all $0 \leq r \leq k-1$ and $M(j, h_i(r)) = 1$
- That is, $s_{i,j}$ stores the minimum index, in the permutation of rows induced by h_i, of a row with value 1 in correspondence to document d_j (the index of the first shingle of d_j)
- This is the current MinHash (for all considered shingles) of document d_j when function h_i is applied

At the end, $s_{i,j}$ will store MinHash for d_j and h_i.

Example

i	d_1	d_2
0	1	1
1	0	0
2	1	1
3	1	0
4	0	1

$$h_1(x) = x \bmod 5$$
$$h_2(x) = (2x+1) \bmod 5$$

h_1	d_1	d_2
0	1	1
1	0	0
2	1	1
3	1	0
4	0	1

h_2	d_1	d_2
0	0	0
1	1	0
2	1	1
3	1	1
4	0	1

$\min(h_1(d_1))=0=0=\min(h_1(d_2))$

$\min(h_2(d_1))=1\neq 2=\min(h_2(d_2))$

$$\hat{J}(d_1, d_2) = \frac{1}{2} = .5$$

$$J(d_1, d_2) = \frac{2}{5} = .4$$

Example

i	d_1	d_2
0	1	1
1	0	0
2	1	1
3	1	0
4	0	1

	$M(h_i(r),1)$	$M(h_i(r),2)$	$s_{1,i}$	$s_{2,i}$
h_1			∞	∞
h_2			∞	∞
$h_1(0) = 0$	1	1	0	0
$h_2(0) = 1$	0	0	∞	∞
$h_1(1) = 1$	0	0	0	0
$h_2(1) = 3$	1	0	1	∞
$h_1(2) = 2$	1	1	0	0
$h_2(2) = 0$	1	1	1	2
$h_1(3) = 3$	1	0	0	0
$h_2(3) = 2$	1	1	1	2
$h_1(4) = 4$	0	1	0	0
$h_2(4) = 4$	0	1	1	2

final sketches

Exercise

	d_1	d_2	d_3
s_1	0	1	1
s_2	1	0	1
s_3	0	1	0
s_4	1	0	0

$$h(x) = 5x + 5 \mod 4$$
$$g(x) = (3x + 1) \mod 4$$

Estimate $\hat{J}(d_1, d_2)$, $\hat{J}(d_1, d_3)$, $\hat{J}(d_2, d_3)$

Efficient near-duplicate detection

- We have an extremely efficient method for estimating similarity for a single pair of documents
- But we still have to estimate $O(N^2)$ values where N is the number of documents: still intractable
- However, often we need to derive all pairs whose similarity is above a given threshold
- One solution: locality sensitive hashing (LSH)

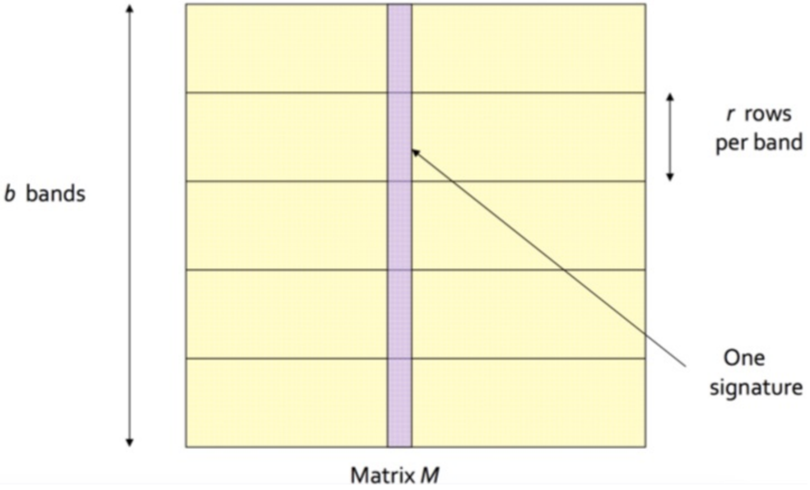
Candidate pairs

- pick a similarity threshold s, $0 \leq s \leq 1$
- goal: find pairs of documents with Jaccard similarity at least s
- columns i and j are a candidate pair if their signatures agree in at least a fraction s of their rows
- we expect pairs of documents to have the same similarity as their signatures

Locality-Sensitive Hashing (LSH) for signatures

- Idea: Hash columns of signatures matrix M' to a predefined set of buckets in such a way that similar columns are likely to be hashed to the same bucket, with high probability
- A pair of columns hashed to the same bucket is a candidate pair for similarity, to be verified more accurately
- False positives (dissimilar pairs hashed to same bucket); false negatives (similar pairs hashed to different buckets)

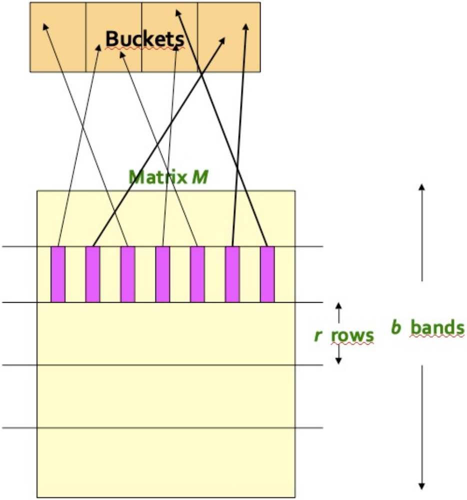
Partition in bands



Partition in bands

- Divide the signature matrix M into b bands, each of r rows.
- For each band B_i, a hash function h_i is defined which maps vectors of r integers to k buckets, with k large enough
- We could use the same hash functions for all bands, but different bucket arrays
- A pair of columns is a candidate pair if they are hashed to the same bucket for at least 1 band
- Tune b (and correspondingly r) to catch most similar pairs, but few not similar ones.

Band hashing



- Columns 2 and 6 are probably identical (candidate pair)
- Columns 6 and 7 are different (wrt to this band, they could be declared candidate pairs by hashing the other bands)

Example

- Assume we have 10^5 columns (documents).
- Each signature is a vector of length 100.
- Each signature element is an integer 4 bytes long.
- Then all signatures are 40MB long.
- The naive approach requires $10^5 \times (10^5 - 1) \times .5 \simeq 5 \times 10^9$ pairs of signatures to be compared: could take months
- Let us apply LSH: choose, for example, $b = 20$, $r = 5$

False negatives

Assume we wish all document pairs with similarity at least .8

- Let columns C_1, C_2 be signatures of similar documents: that is, they have equal values in at least a .8 fraction of their rows
- The probability that columns C_1, C_2 collide in a given band is then $(0.8)^5 = 0.328$.
- The probability that C_1, C_2 do not collide in any of the 20 bands is then $(1 - 0.328)^{20} \simeq 0.00035$.
    - that is, there is a chance of 1 over about 3000 that two 0.8 similar columns do not collide anywhere, and are declared not similar (false negative)
    - we would find 99.965% pairs of truly similar documents: very few false negatives

False positives

- Assume columns C_1, C_2 are signatures of not similar documents: they have equal values in a .3 fraction of their rows
- The probability that columns C_1, C_2 collide in a given band is then $(0.3)^5 = 0.00243$.
- The probability that C_1, C_2 collide in at least one of the 20 bands is then $1 - (1 - 0.00243)^{20} \simeq 0.0474$.
  - that is, approximately 4.74% pairs of docs with similarity 0.3% end up becoming candidate pairs (false positive)
  - they will be checked more precisely and it will turn out they are not similar (at .8 threshold)

Collision probability in a band

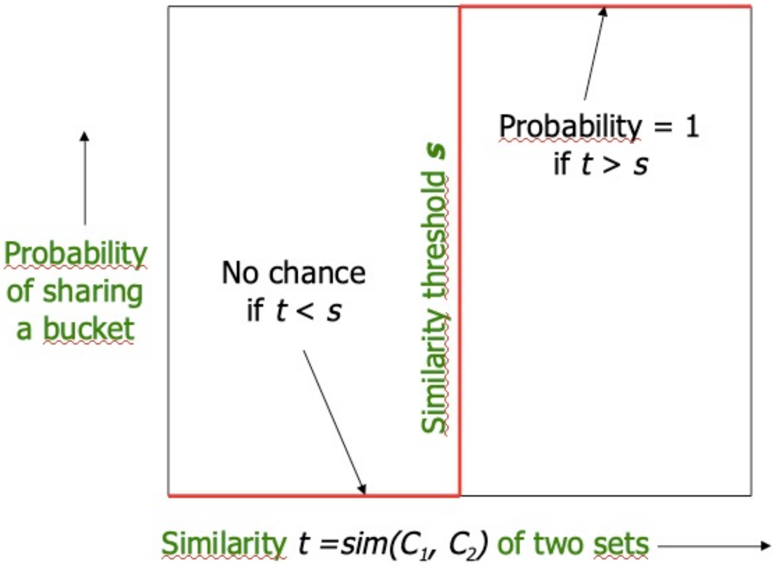
- The probability that two given columns C_1, C_2 have equal rows in a certain band is s^r
- The probability that two given columns C_1, C_2 differ in at least one row in a certain band is $1 - s^r$
- The probability that two given columns C_1, C_2 differ in at least one row in all bands is $(1 - s^r)^b$
- The probability that two given columns C_1, C_2 have equal rows in at least one band (they are a candidate pair) is $1 - (1 - s^r)^b$

La **probabilità** che **una data banda** due colonne con indice di similarità s abbiano **tutte** le r righe **uguali** $\Rightarrow$ s^r

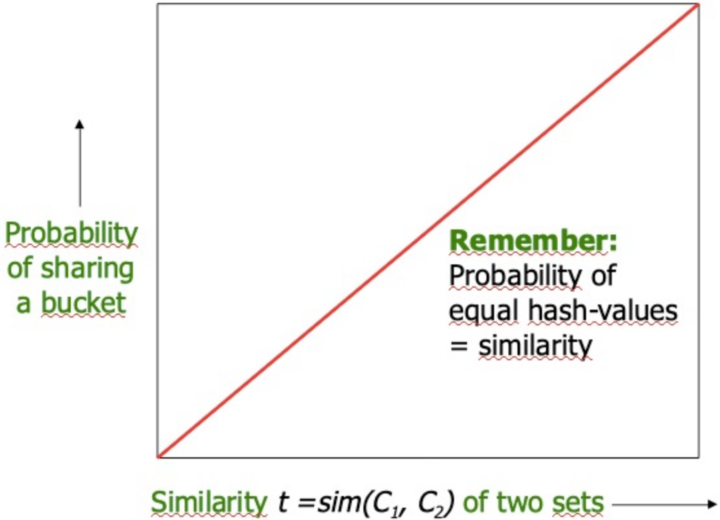
LSH Involves a Tradeof

- Pick
  - The number of MinHashes (rows of M')
  - The number of bands b
  - The number of rows r 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

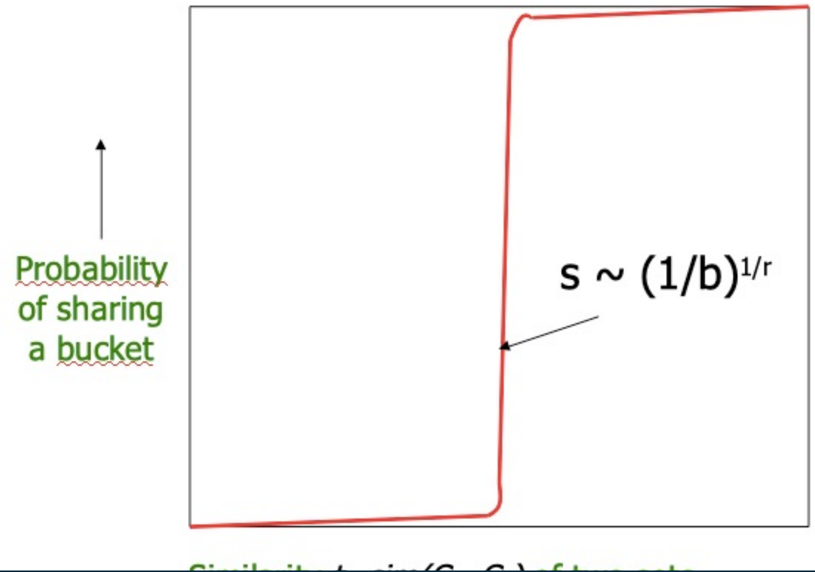
What we want


Probability of sharing a bucket
No chance if $t < s$
Similarity threshold s
Probability = 1 if $t > s$
Similarity $t = sim(C_1, C_2)$ of two sets

What we get with 1 row



What we get with b bands, r rows



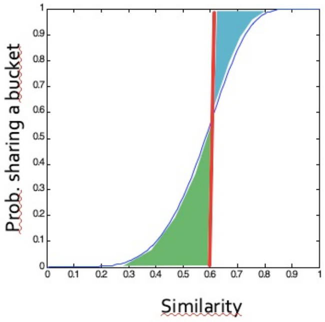
Example: $b = 20$, $r = 5$

- Similarity threshold s
- Probability that at least 1 band is identical (collision)

s	$1-(1-s^r)^b$
.2	.006
.3	.047
.4	.186
.5	.47
.6	.802
.7	.975
.8	.9996

Picking the S-curve

- Picking r and b to get the best S-curve
- 50 hash-functions ($r = 5$, $b = 10$)



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

LSH summary

- Tune M, b, r 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