Lecture 3
Finding Similar Items II

Alexander Schönhuth

UNIVERSITÄT BIELEFELD
Faculty of Technology

Bielefeld University
April 13, 2023

Learning Goals Today

- Understand Minhashing
- Understand the technique of *Locality Sensitive Hashing (LSH)*

Minhashing II
–
Rapidly Computing Similarity of Sets

Minhash - Intermediate Summary / Expansion of Idea

- Computing a minhash means turning a set into one number
- For different sets, numbers agree with probability equal to their Jaccard similarity.
- Can we expand on this idea? Can we compute (ensembles of) numbers that enable us to determine their Jaccard similarity?
- Immediate idea: compute several minhashes. The fraction of times the minhashes of two sets agree equals their Jaccard similarity.
- Several sufficiently well chosen minhashes yield a *minhash signature*.

Minhash Signatures

Consider

- the m rows of the characteristic matrix
- n permutations $\{1, ..., m\} \to \{1, ..., m\}$
- the corresponding *minhash* functions
  $h_1, ..., h_n : \{0, 1\}^m \to \{1, ..., m\}$
- and a particular column $S \in \{0, 1\}^m$
  ☞ $h_i(S) \in \{1, ..., m\}$ for each $i \in \{1, ..., n\}$

Definition [Minhash Signature]
The *minhash signature* SIG_S of S given $h_1, ..., h_n$ is the array

$$[h_1(S), ..., h_n(S)] \in \{1, ..., m\}^n$$

Minhash Signatures

Definition [Minhash Signature]
The *minhash signature* SIG_S of S given $h_1, ..., h_n$ is the array

$$[h_1(S), ..., h_n(S)] \in \{1, ..., m\}^n$$

Meaning: Computing the minhash signature for a column S turns

- the binary-valued array of length m that represents S
  $\leftrightarrow S \in \{0, 1\}^m$
- into an m-valued array of length n
  $\leftrightarrow [h_1(S), ..., h_n(S)] \in \{1, ..., m\}^n$

Because $n < m$ (even $n << m$), the minhash signature is a *(much) reduced representation of a set*.

Signature Matrix

Consider a characteristic matrix, and n permutations $h_1, ..., h_n$.

Definition [Signature Matrix]

The signature matrix SIG is a matrix with n rows and as many columns as the characteristic matrix (i.e. the number of sets), where entries SIG_{ij} are defined by

$$SIG_{ij} = h_i(S_j) \tag{1}$$

where S_j refers to the j-th column in the characteristic matrix.

Signature Matrices: Facts

Let M be a signature matrix.

- Because usually $n << m$, that is n is much smaller than m, a signature matrix is much smaller than the original characteristic matrix.
- The probability that $SIG_{ij_1} = SIG_{ij_2}$ for two sets S_{j_1}, S_{j_2} equals the Jaccard similarity $\mathrm{SIM}(S_{j_1}, S_{j_2})$
- The expected number of rows where columns j_1, j_2 agree, divided by n, is $\mathrm{SIM}(S_{j_1}, S_{j_2})$.

Signature Matrices: Issues

Issue:

- For large m, it is time-consuming / storage-intense to determine permutations
$$\pi : \{1, ..., m\} \rightarrow \{1, ..., m\}$$
- Re-sorting rows relative to a permutation is even more expensive

Solution:

- Instead of permutations, use hash functions (watch the index shift!)
$$h : \{0, ..., m-1\} \rightarrow \{0, ..., m-1\}$$
- Likely, a hash function is not a bijection, so at times
  - places two rows in the same bucket
  - leaves other buckets empty
- Effects are negligible for our purposes, however

Computing Signature Matrices in Practice

- Consider n hash functions $h_i : \{0, ..., m-1\} \rightarrow \{0, ..., m-1\}, i = 1, ..., n$
- Let r and c index rows and columns in the characteristic matrix $M \in \{0, 1\}^{m \times |C|}$
- So i and c index rows and columns in the signature matrix $SIG \in \{1, ..., m\}^{n \times |C|}$

```
for each c do
    for 0 ≤ i ≤ n do
        SIG(i, c) = ∞
    end for
end for
for each row r do
    for each column c do
        if M(r, c) = 1 then
            for i=1 to n do
                SIG(i, c) =
                min(SIG(i, c), h_i(r))
            end for
        end if
    end for
end for
```

Computing Signature Matrices: Example

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

Hash functions computed for a characteristic matrix, with rows indexed from 0 to 4

Computing Signature Matrices in Practice

- Consider n hash functions $h_i : \{0, ..., m-1\} \to \{0, ..., m-1\}, i = 1, ..., n$
- Let r and c index rows and columns in the characteristic matrix $M \in \{0,1\}^{m \times |C|}$
- So c also index columns, while i indexes rows in the signature matrix $SIG \in \{1, ..., m\}^{n \times |C|}$

```
for each c do
    for 0 ≤ i ≤ n do
        SIG(i, c) = ∞
    end for
end for
for each row r do
    for each column c do
        if M(r, c) = 1 then
            for i=1 to n do
                SIG(i, c) =
                min(SIG(i, c), h_i(r))
            end for
        end if
    end for
end for
```

Computing Signature Matrices: Example

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

Hash functions computed for a characteristic matrix, with rows indexed from 0 to 4

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

Signature matrix *SIG*: after initialization

Computing Signature Matrices in Practice

- Consider n hash functions $h_i : \{0, ..., m-1\} \rightarrow \{0, ..., m-1\}, i = 1, ..., n$
- Let r and c index rows and columns in the characteristic matrix $M \in \{0,1\}^{m \times |C|}$
- So c also index columns, while i indexes rows in the signature matrix $SIG \in \{1, ..., m\}^{n \times |C|}$

```
for each c do
    for 0 ≤ i ≤ n do
        SIG(i, c) = ∞
    end for
end for
for each row r do
    for each column c do
        if M(r, c) = 1 then
            for i=1 to n do
                SIG(i, c) =
                min(SIG(i, c), h_i(r))
            end for
        end if
    end for
end for
```

Computing Signature Matrices in Practice

- Consider n hash functions $h_i : \{0, ..., m-1\} \rightarrow \{0, ..., m-1\}, i = 1, ..., n$
- Let r and c index rows and columns in the characteristic matrix $M \in \{0,1\}^{m \times |C|}$
- So c also index columns, while i indexes rows in the signature matrix $SIG \in \{1, ..., m\}^{n \times |C|}$

```
for each c do
    for 0 ≤ i ≤ n do
        SIG(i, c) = ∞
    end for
end for
for each row r do
    // Iteration 1: first row
    for each column c do
        if M(r, c) = 1 then
            for i=1 to n do
                SIG(i, c) =
                min(SIG(i, c), h_i(r))
            end for
        end if
    end for
    // End first row
end for
```

COMPUTING SIGNATURE MATRICES: EXAMPLE

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

Hash functions computed for a characteristic matrix, with rows indexed from 0 to 4

First iteration: row # 0 has 1's in S_1 and S_4, so put
$SIG_{11} = SIG_{14} = \min\{\infty, h_1(0)\} = 0 + 1 \mod 5 = 1,$
$SIG_{21} = SIG_{24} = \min\{\infty, h_2(0)\} = 3 \cdot 0 + 1 \mod 5 = 1$

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

Signature matrix after considering first row

COMPUTING SIGNATURE MATRICES IN PRACTICE

- Consider n hash functions $h_i : \{0, ..., m-1\} \rightarrow \{0, ..., m-1\}, i = 1, ..., n$
- Let r and c index rows and columns in the characteristic matrix $M \in \{0,1\}^{m \times |C|}$
- So c also index columns, while i indexes rows in the signature matrix $SIG \in \{1, ..., m\}^{n \times |C|}$

```
for each c do
    for 0 ≤ i ≤ n do
        SIG(i, c) = ∞
    end for
end for
for each row r do
    // Iteration 2: second row
    for each column c do
        if M(r, c) = 1 then
            for i=1 to n do
                SIG(i, c) =
                min(SIG(i, c), h_i(r))
            end for
        end if
    end for
    // End second row
end for
```

Computing Signature Matrices: Example

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

Hash functions computed for a characteristic matrix, with rows indexed from 0 to 4

Second iteration: row #1 has 1 in S_3, so put
$SIG_{13} = \min\{\infty, h_1(1)\} = 1 + 1 \mod 5 = 2$,
$SIG_{23} = \min\{\infty, h_2(1)\} = 3 + 1 \mod 5 = 4$.

	S_1	S_2	S_3	S_4
h_1	1	∞	2	1
h_2	1	∞	4	1

Signature matrix M after considering second row

Computing Signature Matrices in Practice

- Consider n hash functions $h_i : \{0, ..., m-1\} \rightarrow \{0, ..., m-1\}, i = 1, ..., n$
- Let r and c index rows and columns in the characteristic matrix $M \in \{0, 1\}^{m \times |C|}$
- So c also index columns, while i indexes rows in the signature matrix $SIG \in \{1, ..., m\}^{n \times |C|}$

```
for each c do
    for 0 ≤ i ≤ n do
        SIG(i, c) = ∞
    end for
end for
for each row r do
    // Iteration 3: third row
    for each column c do
        if M(r, c) = 1 then
            for i=1 to n do
                SIG(i, c) =
                min(SIG(i, c), h_i(r))
            end for
        end if
    end for
    // End third row
end for
```

Computing Signature Matrices: Example

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

Hash functions computed for a characteristic matrix, with rows indexed from 0 to 4

Third iteration: row # 2 has 1's in S_2 and S_4, so put
$SIG_{12} = \min\{\infty, h_1(2)\} = 2+1 \mod 5 = 3$,
$SIG_{14} = \min\{SIG_{14}, h_1(2)\} = \min(1, 2+1 \mod 5 = 3) = 1$,
$SIG_{22} = \min\{\infty, h_2(2)\} = 6+1 \mod 5 = 2$,
$SIG_{24} = \min\{SIG_{24}, h_2(2)\} = \min(1, 6+1 \mod 5 = 2) = 1$

Computing Signature Matrices: Example

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

Hash functions computed for a characteristic matrix, with rows indexed from 0 to 4

	S_1	S_2	S_3	S_4
h_1	1	3	2	1
h_2	1	2	4	1

Signature matrix after considering third row

Computing Signature Matrices in Practice

- Consider n hash functions $h_i : \{0, ..., m-1\} \to \{0, ..., m-1\}, i = 1, ..., n$
- Let r and c index rows and columns in the characteristic matrix $M \in \{0,1\}^{m \times |C|}$
- So c also index columns, while i indexes rows in the signature matrix $SIG \in \{1, ..., m\}^{n \times |C|}$

```
for each c do
    for 0 ≤ i ≤ n do
        SIG(i, c) = ∞
    end for
end for
for each row r do
    // Iteration 4: fourth row
    for each column c do
        if M(r, c) = 1 then
            for i=1 to n do
                SIG(i, c) =
                    min(SIG(i, c), h_i(r))
            end for
        end if
    end for
    // End fourth row
end for
```

Computing Signature Matrices: Example

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

Hash functions computed for a characteristic matrix, with rows indexed from 0 to 4

Fourth iteration: SIG_{11} stays 1, SIG_{21} changes to 0, SIG_{13} stays 2, SIG_{23} changes to 0, SIG_{14} stays 1, SIG_{24} changes to 0

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

Signature matrix after considering fourth row

Computing Signature Matrices in Practice

- Consider n hash functions $h_i : \{0, ..., m-1\} \rightarrow \{0, ..., m-1\}, i = 1, ..., n$
- Let r and c index rows and columns in the characteristic matrix $M \in \{0,1\}^{m \times |C|}$
- So c also index columns, while i indexes rows in the signature matrix $SIG \in \{1, ..., m\}^{n \times |C|}$

```
for each c do
    for 0 ≤ i ≤ n do
        SIG(i, c) = ∞
    end for
end for
for each row r do
    // Iteration 5: fifth (final) row
    for each column c do
        if M(r, c) = 1 then
            for i=1 to n do
                SIG(i, c) =
                min(SIG(i, c), h_i(r))
            end for
        end if
    end for
    // End fifth (final) row
end for
```

Computing Signature Matrices: Example

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

Hash functions computed for a characteristic matrix, with rows indexed from 0 to 4

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

Signature matrix after considering fifth row: final signature matrix

Computing Signature Matrices: Example

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

Signature matrix after considering fifth row: final signature matrix

- *Estimates* for Jaccard similarity: $\text{SIM}(S_1, S_3) = \frac{1}{2}, \text{SIM}(S_1, S_4) = 1$
- *True* Jaccard similarities: $\text{SIM}(S_1, S_3) = \frac{1}{4}, \text{SIM}(S_1, S_4) = \frac{2}{3}$
- Estimates will be better when raising number of hash functions that is increasing number of rows of the signature matrix

Minhashing

–

Speeding Up Computations

SPEEDING UP MINHASHING: BASIC IDEA

- Minhashing is time-consuming, because iterating through all m rows of M necessary, and m is large (huge!)
- *Thought experiment:*
  - Recall: minhash is first row in permuted order with a 1
  - Consider permutations $\pi : \{1, ..., \bar{m}\} \rightarrow \{1, ..., \bar{m}\}$ for $\bar{m} < m$
  - Consider only examining the first $\bar{m}$ of the permuted rows
  - Speed up of a factor of $\frac{m}{\bar{m}}$

Speeding Up Minhashing: Basic Idea II

- Minhashing is about *estimates*
- Minhashing on subsets of the real sets may provide good estimates already?
- How do estimates behave more concretely?

Speeding Up Minhashing: Basic Idea III

- Continue thought experiment...
- Consider computing signature matrices by only examining $\bar{m} < m$ rows in the characteristic matrix, and using permutations $\pi : \{1, ..., \bar{m}\} \rightarrow \{1, ..., \bar{m}\}$
- By the way: the chosen $\bar{m}$ rows need not be the first $\bar{m}$ rows
- For each column where no 1 shows, keep ∞ as symbol in the signature matrix *SIG*

Speeding Up Minhashing: Issues I

- There may be columns where all first $\bar{m}$ rows contain zeroes
- Using the algorithm discussed previously, we will have ∞ symbols in the signature matrix

	S_1	S_2	S_3	S_4
h_1	1	∞	2	1
h_2	1	∞	4	1

Signature matrix M with ∞ remaining (not referring to example from slide before)

Speeding Up Minhashing: Issues II

- *Situation:* Much faster to compute SIG, but $SIG(i, c) = \infty$ in some places (how many? is this bad?)
- How to deal with that? Can we nevertheless work with only $\bar{m} < m$ rows and compute sufficiently accurate estimates for the Jaccard similarity of two columns?

Speeding Up Minhashing: Practice I

Situation:

- Compute Jaccard similarities for pairs of columns, while possibly
- $SIG(i, c) = \infty$ for some (i, c)
- *Algorithm for estimating Jaccard similarity:*
  - Row by row, by iterative updates,
  - Maintain count of rows a where columns agree
  - Maintain count of rows d where columns disagree
  - Estimate SIM as $\frac{a}{a+d}$

Speeding Up Minhashing: Practice II

- Maintain count of rows a where columns agree
- Maintain count of rows d where columns disagree
- Estimate SIM as $\frac{a}{a+d}$

Three cases:

1. *Both columns do not contain ∞ in row:* update counts as usual (either $a \rightarrow a+1$ or $d \rightarrow d+1$)
2. *Only one column has ∞ in row:*
   - Let two columns be c_1, c_2, and $SIG(i, c_1) = \infty$, but $SIG(i, c_2) \neq \infty$:
   - It follows that $SIG(i, c_1) > SIG(i, c_2)$
   - So increase count of disagreeing rows by one ($d \rightarrow d+1$)
3. *Both columns have ∞ in a row:* unclear, skip updating counts

SPEEDING UP MINHASHING: PRACTICE III

Summary: One determines $\frac{a}{a+d}$ as estimate for $SIM(c_1, c_2)$

- (*) Counts rely on less rows than before
- (**) However, since each permutation only refers to $\bar{m} < m$ rows, we can afford more permutations
- (*) makes counts less reliable, while (**) compensates for it
- Can we control the corresponding trade-off to our favour?

SPEEDING UP MINHASHING: THEORY I

- Let T be the set of elements of the universal set that correspond to the initial $\bar{m}$ rows in the characteristic matrix.
- When executing the above algorithm on only these $\bar{m}$ rows, we determine

$$\frac{|S_1 \cap S_2 \cap T|}{|(S_1 \cup S_2) \cap T|} \tag{2}$$

  as an estimate for the true Jaccard similarity $\frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$.
- If T is chosen randomly, the expected value of (2) is the Jaccard similarity $\mathrm{SIM}(S_1, S_2)$
- But: there may be some disturbing variation to this estimate

Speeding up Minhashing: Practice IV

- Divide m rows into $\frac{m}{\bar{m}}$ blocks of $\bar{m}$ rows each
- For each hash function $h : \{0, ..., \bar{m} - 1\} \to \{0, ..., \bar{m} - 1\}$, compute minhash values for each block of $\bar{m}$ rows
- Yields $\frac{m}{\bar{m}}$ minhash values for a single hash function, instead of just one
- *Extreme:* If $\frac{m}{\bar{m}}$ is large enough, only one hash function may be necessary
- *Possible advantage:*
  - Type X rows are distributed across blocks of $\bar{m}$ rows
  - Type Y rows are distributed across blocks of $\bar{m}$ rows
  - Using all m rows balances out irregularities across blocks

Speeding up Minhashing: Example

S_1	S_2	S_3
0	0	0
0	0	0
0	0	1
0	1	1
1	1	1
1	1	0
1	0	0
0	0	0

Characteristic matrix for three sets S_1, S_2, S_3. $m = 8, \bar{m} = 4$.

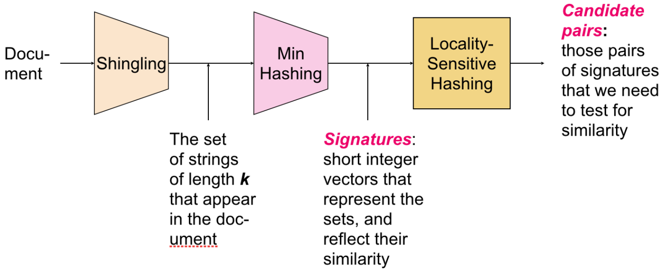
- *Truth:* $\text{SIM}(S_1, S_2) = \frac{1}{2}, \text{SIM}(S_1, S_3) = \frac{1}{5}, \text{SIM}(S_2, S_3) = \frac{1}{2}$
- *Estimate for first four rows:* $\text{SIM}(S_1, S_2) = 0$
- *Estimate for last four rows:* $\text{SIM}(S_1, S_2) = \frac{2}{3}$ on average across randomly picked hash functions
- *Overall estimate (expected across randomly picked hash functions):* $\text{SIM}(S_1, S_2) = \frac{1}{3}$, Ok estimate for two hash functions

Current Status

–

Summary

SUMMARY OF CURRENT STATUS



From `mmds.org`

- *Shingling:* turning text files into sets ☞ Done!
- *Minhashing:* computing similarity for large sets ☞ Done!
- *Locality Sensitive Hashing:* avoids $O(N^2)$ comparisons by determining candidate pairs ☞ Coming next!

Current Status: Issues Still Remaining

- Minhashing enabled to compute similarity between two sets very fast
- Shingling enabled to turn documents into sets such that minhashing could be applied
- But if number of items N is too large, $O(N^2)$ similarity computations are infeasible, even using minhashing
- *Idea:* Browse through items and determine *candidate pairs*:
  - Number of candidate pairs is much smaller than $O(N^2)$
  - One performs minhashing only for candidate pairs
  - Candidate pairs can be determined with a very fast procedure
- *Solution: Locality Sensitive Hashing (a.k.a. Near Neighbor Search)*

Locality Sensitive Hashing

Signature Matrix: Reminder

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

Signature matrix *SIG* for two permutations (hash functions) h_1, h_2, and four sets S_1, S_2, S_3, S_4

- Figure:
  - Size of universal set: $m = 5$
  - Number of hash functions: $n = 2$
  - Number of sets: $N = 4$
- Originally: each set is from $\{0, 1\}^m$ (a bitvector of length m)
- Now: each set is from $\{0, ..., m - 1\}^n$
- Much reduced representation, because $n << m$

Locality Sensitive Hashing: Idea

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

Signature matrix *SIG* for two permutations (hash functions) h_1, h_2, and four sets S_1, S_2, S_3, S_4

Idea:

- Hash columns in *SIG* using several hash functions into buckets
- *Candidate pair:* Pair of columns hashed to same bucket by any function

Runtime:

- Hashing all columns is $O(N)$ (much faster than $O(N^2)$)
- Examining buckets requires little time

LOCALITY SENSITIVE HASHING: CHALLENGE

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

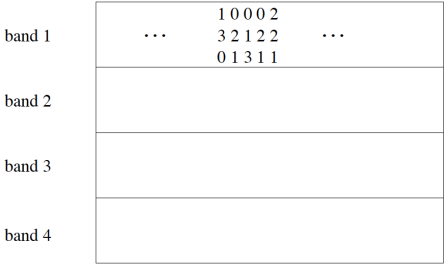
Signature matrix *SIG* for two permutations (hash functions) h_1, h_2, and four sets S_1, S_2, S_3, S_4

Challenge:

- Hash similar columns to same buckets
- Hash dissimilar columns to different buckets

How to design hash functions?

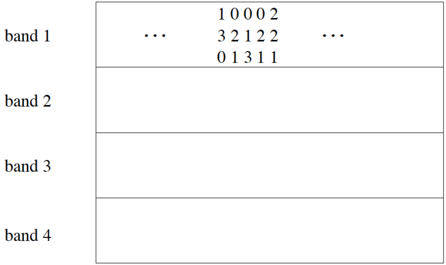
Locality Sensitive Hashing: Banding Technique



Signature matrix divided into $b = 4$ bands of $r = 3$ rows each

- Divide rows of signature matrix into b bands of r rows each
- For each band, a hash function hashes r integers to buckets
- Number of buckets is large to avoid collisions
- *Candidate pair:* a pair of columns hashed to the same bucket, in any band

Banding Technique: Example



Signature matrix divided into $b = 4$ bands of $r = 3$ rows each

- The columns showing $[0, 2, 1]$ in band 1 are declared a candidate pair
- Other pairs of columns no candidate pairs because of first band
  - apart from collisions occurring ☞ designed to happen very rarely
- Columns hashed to same bucket in another band ☞ candidate pairs

Banding Technique: Theorem

Let *SIG* be a signature matrix grouped into

- b bands of
- r rows each

and consider

- a pair of columns of Jaccard similarity s

Theorem [LSH Candidate Pair]:
The probability that the pair of columns becomes a candidate pair is

$$1 - (1 - s^r)^b \tag{3}$$

Banding Technique: Proof of Theorem

Proof.

Consider a pair of columns whose sets have Jaccard similarity s.

- Given any row, by Theorem "Minhash and Jaccard Similarity" of last lecture, they agree in that row with probability s

Because minhash values are independent of each other, the probability to

- agree in all rows of one band is s^r
- disagree in at least one of the rows in a band $1 - s^r$
- disagree in at least one row in each band is $(1 - s^r)^b$
- agree in all rows for at least one band is $1 - (1 - s^r)^b$

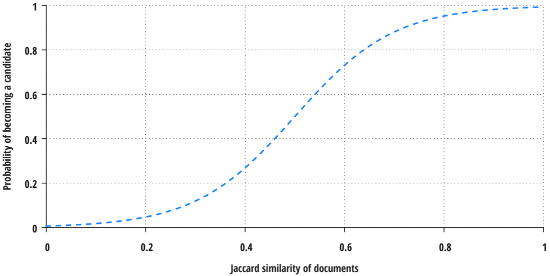
□

Banding Technique: The S-Curve

Definition: [S-Curve]

For given b and r, the *S-curve* is defined by the prescription

$$s \mapsto 1 - (1 - s^r)^b \qquad (4)$$



Exemplary S-curve

Banding Technique: The S-Curve

Definition: [S-Curve]
For given b and r, the *S-curve* is defined by the prescription

$$s \mapsto 1 - (1 - s^r)^b \tag{5}$$

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

Table: Values for S-curve with $b = 20$ and $r = 5$

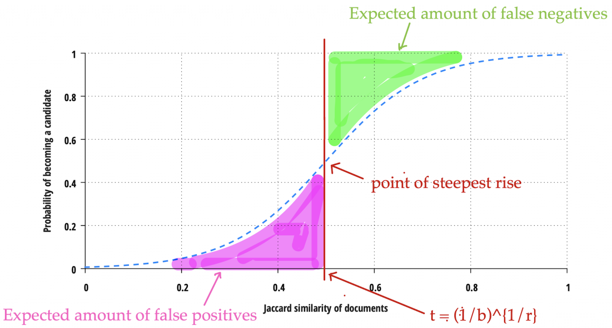
Locality Sensitive Hashing: Guidelines

- One needs to determine b, r where $br = n$
- One needs to determine threshold t:
  - $s \geq t$: candidate pair
  - $s < t$: no candidate pair
- t corresponds with point of steepest rise on S-curve: approximately $(1/b)^{(1/r)}$

Motivation:

- *False Positive:* dissimilar pair hashing to the same bucket
- *False Negative:* similar pair never hashing to the same bucket
- *Motivation:* limit both false positives and negatives

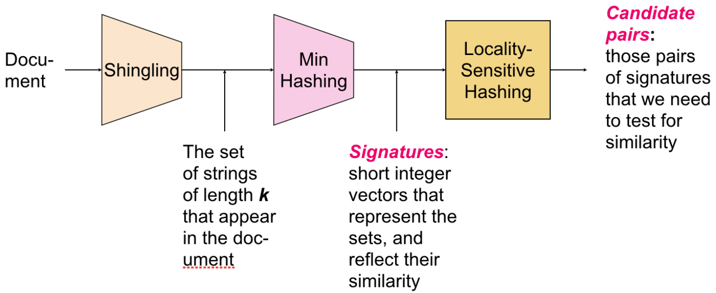
LSH: False Negatives / Positives



Exemplary S-curve

- Pick threshold t, number of bands b and rows r
- Avoiding false negatives: have $t \approx (1/b)^{1/r}$ low
- Avoiding false positives, or enhancing speed: have $t \approx (1/b)^{1/r}$ large

Finding Similar Documents: Overall Workflow



From `mmds.org`

- Shingling: Done!
- Minhashing: Done!
- Locality-Sensitive Hashing: Done!

Finding Similar Documents: Summary

1. *Shingling:*
   - Pick k and determine k-shingles for each document
   - Sort shingles, document is bitvector over universe of shingles
2. *Minhashing:*
   - Pick n hash functions
   - Compute minhash signatures as per earlier algorithm
3. *Locality Sensitive Hashing:*
   - Pick number of bands b and rows r
   - Watch $t \approx (1/b^{1/r}$ ☞ avoid false negatives/positives
   - Determine candidate pairs by applying the banding technique
4. Return to signatures of candidate pairs and determine whether fraction of components where they agree is at least t

Materials / Outlook

- See *Mining of Massive Datasets*, chapter 3.3–3.4
- See `http://www.mmds.org/` for further resources
- Next lecture: Presentation by mindsquare & “Finding Similar Items III”
  - See *Mining of Massive Datasets* 3.5–3.7

Example / Illustration