Shingles, MinHash, LSH

Tomasz Kuśmierczyk: tomaszku@ntnu.no
2017-02-24

What’s obligatory?

Slides marked with star (*) are for people that want a deeper understanding of the topic, and are not curriculum for the exam.

Additional materials:

Book ch. 3 (only some parts apply)

http://infolab.stanford.edu/~ullman/mmds/ch3.pdf

Slides from the book's authors:

http://www.mmds.org/mmds/v2.1/ch03-lsh.pdf

Context

- Many real-world applications require **finding “near duplicate”/”similar” pairs:**
  - Pages with similar words
  - Customers who purchased similar products
  - Products with similar customer sets
  - Images with similar features
  - Users who visited similar websites

Today’s Setting

- Goal: Given a large number (in the millions or billions) of documents, **find “near duplicate” pairs**

http://www.mmds.org/mmds/v2.1/ch03-lsh.pdf

Today's Setting

- Goal: Given a large number (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 sites
    - Cluster articles by "same story"

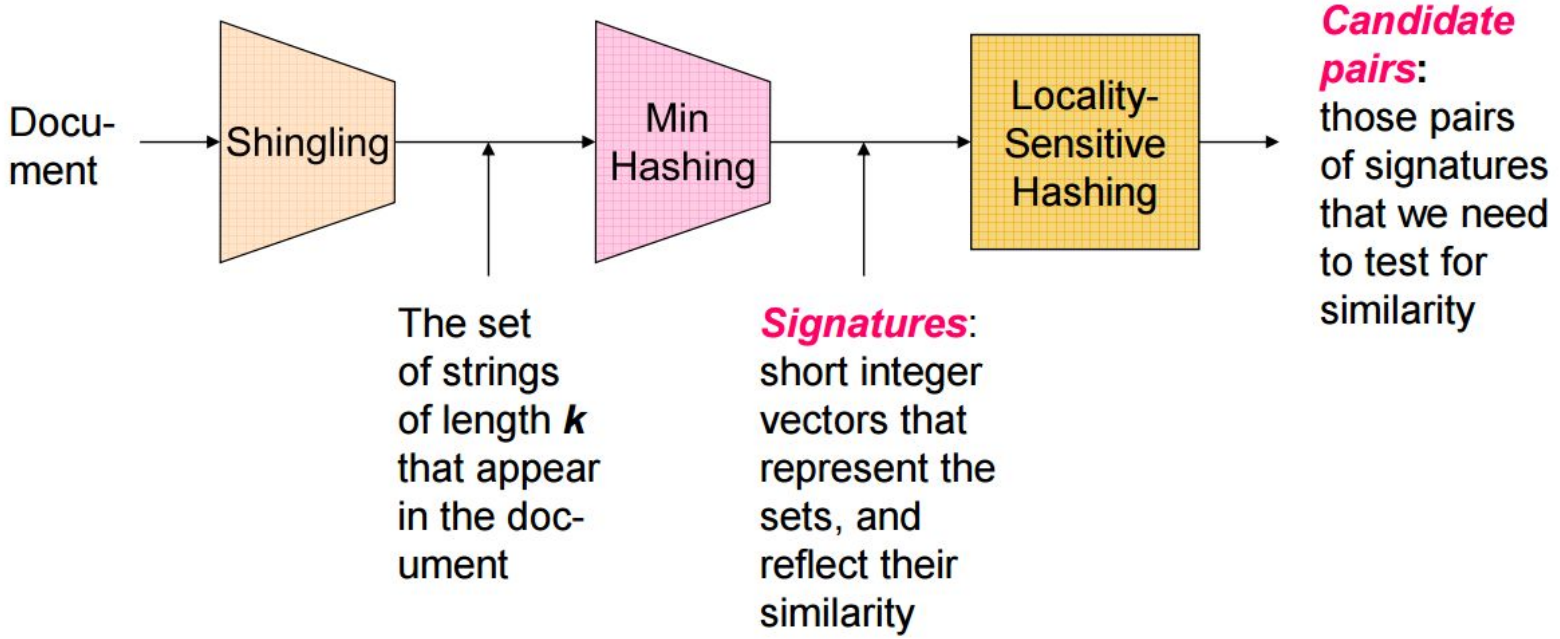
http://www.mmds.org/mmds/v2.1/ch03-lsh.pdf

Today's Setting

- Goal: Given a large number (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 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 main memory

http://www.mmds.org/mmds/v2.1/ch03-lsh.pdf

What are we going to see today?



Representations

Shingles: docs -> sets conversion

- A k-shingle (or k-gram) for a document is a sequence of k tokens that appears in the doc
- Tokens can be characters, words or something else, depending on the application

Example: bigrams

- k=2

- document D1 = knowledge is important
- bigrams: {knowledge is, is important}

- document D2 = nowadays imagination is important
- bigrams: {nowadays imagination, imagination is, is important}

Example: chars

- k=2

- document D1 = abcab
- 2-shingles: {ab, bc, ca}

- document D2 = caab
- 2-shingles: {?, ?}

Example: chars

- k=2

- document D1 = abcab
- 2-shingles: {ab, bc, ca}

- document D2 = caab
- 2-shingles: {ab, aa, ca}

What k?

- k should be chosen that:
  - Probability of two documents having the same shingle by chance is low
  - If two documents are similarity they contain at least one matching shingle

What k?

- k should be chosen that:
  - Probability of two documents having the same shingle by chance is low
  - If two documents are similarity they contain at least one matching shingle

How many 10-shingles are there?

- k=10, ASCII chars (27)
  - 27^10 ~ 2 * 10^14 - pretty many
- But most (99.9%) of them are unlikely
- Let’s compress:
  - represent a document by the set of hash values of its k-shingles:
  - Abcab -> {ab, bc, ca} -> {1, 5, 7} (2^32 numbers)

Jaccard

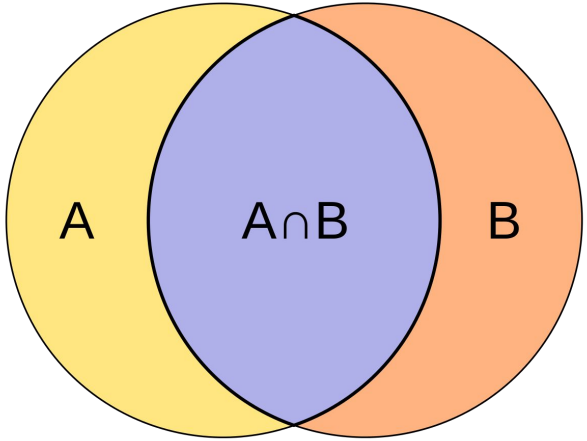
Similarity Measures: Jaccard Similarity

- we have docs as sets (for example of n-grams)

but

- **what “near duplicated” / “similar” means?**

Jaccard Similarity



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

coffee ↗

Jaccard Similarity example

knowledge is very very very important

imagination is more important than knowledge

abs is more important than imagination

potato is the most important

Jaccard Similarity example

knowledge is very very very important

S_1 = {*knowledge, is, very, important*}

imagination is more important than knowledge

S_2 = {*imagination, is, more, important, than, knowledge*}

abs is more important than imagination

S_3 = {*abs, is, more, important, than imagination*}

potato is the most important

S_4 = {*potato, is, the, most, important*}

Jaccard Similarity example

knowledge more most potato than the music is important very and cabbage imagination

S_1 = {*knowledge, is, very, important*}

S_2 = {*imagination, is, more, important, than, knowledge*}

Preprocessing is important [Assignment]

potato

knowledge

music

important

cabbage

imagination

S_1 = {*knowledge*, *important*}

S_2 = {*imagination*, *important*, *knowledge*}

Preprocessing is important

potato

knowledge

music

important

cabbage

imagination

S_1 = {*knowledge*, *important*}

S_4 = {*potato*, *important*}

MinHashing

Setting

- Goal: Find similar documents (sets)
  - Documents are already converted to sets
- Similarity between documents = J
- **Documents are large**
  - Example: 10000 characters per doc, 9-shingles
  - How much additional space for representations?
- Every document is compared many times

How do we solve it?

- Comparing is too slow!
- We need: smaller representations of documents (=sets) that can be compared even faster

How do we solve it?

- Comparing is too slow!
- We need: smaller representations of documents (=sets) that can be compared even faster
  - Like: MinHash signatures!

Sets can be encoded as bit (column) vectors [Assignment]

For now all possible elements (words) = {*a*,*g*,*i*,*k*,*p*}

Sets (documents) = {S_1, S_2, S_3, S_4}

S_1 = {*k*, *i*} -> S_4 = $(0\ 0\ 1\ 1\ 0)^T$

S_2 = {*g*, *i*, *k*} -> S_4 = $(0\ 1\ 1\ 1\ 0)^T$

S_3 = {*a*, *i*, *g*} -> S_4 = $(1\ 1\ 1\ 0\ 0)^T$

S_4 = {*p*, *i*} -> S_4 = $(???)^T$

Sets can be encoded as bit (column) vectors

For now all possible elements (words) = {*a*,*g*,*i*,*k*,*p*}

Sets (documents) = {S_1, S_2, S_3, S_4}

S_1 = {*k*, *i*} -> S_4 = $(0\ 0\ 1\ 1\ 0)^T$

S_2 = {*g*, *i*, *k*} -> S_4 = $(0\ 1\ 1\ 1\ 0)^T$

S_3 = {*a*, *i*, *g*} -> S_4 = $(1\ 1\ 1\ 0\ 0)^T$

S_4 = {*p*, *i*} -> S_4 = $(0\ 0\ 1\ 0\ 1)^T$

Matrix representation

Ixs <-> Elements

Ix		S1	S2	S3	S4
0	a	0	0	1	0
1	g	0	1	1	0
2	i	1	1	1	1
3	k	1	1	0	0
4	p	0	0	0	1

Minhash Signature: random permutations + min

Ix		S1	S2	S3	S4
0	a	0	0	1	0
1	g	0	1	1	0
2	i	1	1	1	1
3	k	1	1	0	0
4	p	0	0	0	1

Permutate rows

Ix		S1	S2	S3	S4	Ix2
4	p	0	0	0	1	0
0	a	0	0	1	0	1
1	g	0	1	1	0	2
2	i	1	1	1	1	3
3	k	1	1	0	0	4

Min(Ix2) containing element

	S1	S2	S3	S4
h1	3	2	1	0

First row of a signature

Minhash Signature: random permutations + min [Assignment]

Ix		S1	S2	S3	S4
0	a	0	0	1	0
1	g	0	1	1	0
2	i	1	1	1	1
3	k	1	1	0	0
4	p	0	0	0	1

Permutate rows

Ix		S1	S2	S3	S4	Ix2
3	k	1	1	0	0	0
0	a	0	0	1	0	1
2	i	1	1	1	1	2
4	p	0	0	0	1	3
1	g	0	1	1	0	4

Min(Ix2) containing element

	S1	S2	S3	S4
h2	?	?	?	?

Second row of a signature

Minhash Signature: random permutations + min

Ix		S1	S2	S3	S4
0	a	0	0	1	0
1	g	0	1	1	0
2	i	1	1	1	1
3	k	1	1	0	0
4	p	0	0	0	1

Permutate rows

Ix		S1	S2	S3	S4	Ix2
3	k	1	1	0	0	0
0	a	0	0	1	0	1
2	i	1	1	1	1	2
4	p	0	0	0	1	3
1	g	0	1	1	0	4

Min(Ix2) containing element

	S1	S2	S3	S4
h2	0	0	1	2

Second row of a signature

Minhash Signature

	S1	S2	S3	S4
h1	3	2	1	0

	S1	S2	S3	S4
h2	0	0	1	2

	S1	S2	S3	S4
h1	3	2	1	0
h2	0	0	1	2

Minhash Signature

Ix		S1	S2	S3	S4
0	a	0	0	1	0
1	g	0	1	1	0
2	i	1	1	1	1
3	k	1	1	0	0
4	p	0	0	0	1

	S1	S2	S3	S4
h1	3	2	1	0
h2	0	0	1	2

Great! We have only smaller fixed (here =2) number of signature rows (= new elements) to deal with. Not-even binary…

How to compare them to get similarity?

Derivation: probability? *

Ix		S1	S2	Ix2
4	k	1	1	0
0	a	0	0	1
1	i	1	1	2
2	p	0	0	3
3	g	0	1	4

Let's calc the **probability** of:

h(S1) = h(S2)

for some random permutation

Derivation: Row types *

lx		S1	S2	lx2
4	k	1	1	0
0	a	0	0	1
1	i	1	1	2
2	p	0	0	3
3	g	0	1	4

We have three row types:

- x rows with 1 in both cols
- y rows with 1 in one col
- z rows with 0 in both cols

Derivation: observation *

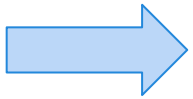
Ix		S1	S2	Ix2
4	k	1	1	0
0	a	0	0	1
1	i	1	1	2
2	p	0	0	3
3	g	0	1	4

Let's calc the probability of:
h(S1) = h(S2)
for some random permutation

Observation 1: gray rows can change h(S1) or h(S2) but can not change if h(S1) = h(S2)

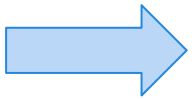
Derivation: green and red left *

Observation 1: gray rows can change h(S1) or h(S2) but can not change if h(S1) = h(S2)



We can ignore them for now!

Ix		S1	S2	Ix2
4	k	1	1	0
1	i	1	1	2
3	g	0	1	4



Now:

h(S1) = h(S2) if the first row is green

h(S1) != h(S2) if the first row is green

Derivation: what's the solution? *

Now:

h(S1) = h(S2) if the first row is red

h(S1) != h(S2) if the first row is green

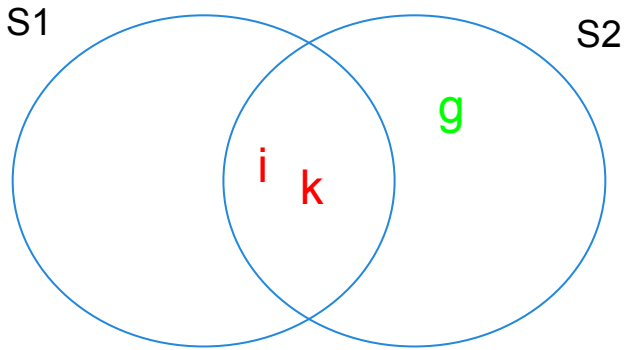
Ix		S1	S2	Ix2
4	k	1	1	0
1	i	1	1	2
3	g	0	1	4

There are x red rows and y green rows

For how many permutations red is on the top?

Derivation: what's the solution? *

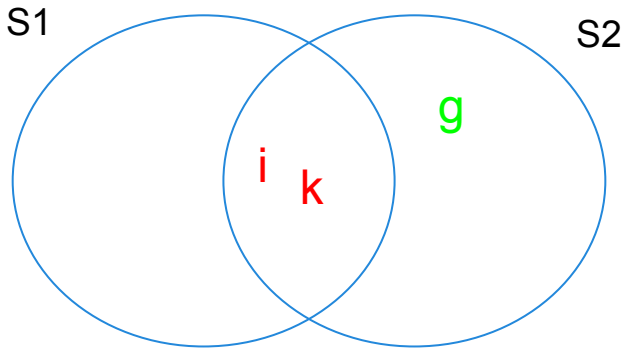
P[h(S1) = h(S2)] = x / (x+y) = ?



lx		S1	S2	lx2
4	k	1	1	0
1	i	1	1	2
3	g	0	1	4

Derivation: what's the solution?

P[h(S1) = h(S2)] = x / (x+y) = J(S1,S2)



Ix		S1	S2	Ix2
4	k	1	1	0
1	i	1	1	2
3	g	0	1	4

Problem?

Nice! We know that there is some probability s:

$$s = P[\, h(S1) = h(S2) \,] = J(S1,S2)$$

But we do not observe probabilities!
We observe events:
if h(S1) = h(S2) for some h or not

Problem?

We need to estimate somehow this probability s:

- Permute i = 1..H times
- Check how many times: $h_i(S1) = h_i(S2)$
- Then the estimator of s:

$$s_{MLE} = \#\, h_i(S1) = h_i(S2) \,/\, H$$

Back to our example: results vs true

	S1	S2	S3	S4
h1	3	2	1	0
h2	0	0	1	2

Reality

- Matrix representation would be huge
- Materializing a permutation takes a lot of space
- What with a new document?

Solutions:

- We do not store the data as a matrix
- We do not permute -> we simulate permutating with hash functions
- Signatures are computed iteratively

Simulating permutations: general formula

$$\mathbf{ix2(ix) = ((a \cdot ix + b) \bmod p)}$$

For and a,b and prime p

Simulating permutations: *example*

$$ix2(ix) = ((3 \cdot ix + 1) \bmod 5)$$

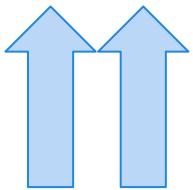
Ix	Ix2
0	1
1	4
2	2
3	0
4	3

Algorithm to calc a signatures of all docs (from the book)

1. Initialize signatures with inf
2. For every element indexed with ix:
    a. For every perumutation:
        i. For every document:
            1. If element not present in a doc skip, otherwise:
            2. signature = min(signature, ix2(ix))

Algorithm example *(blackboard)*

Ix		S1	S2	S3	S4	Ix2	Ix2
0	a	0	0	1	0	1	1
1	g	0	1	1	0	2	4
2	i	1	1	1	1	3	2
3	k	1	1	0	0	4	0
4	p	0	0	0	1	0	3

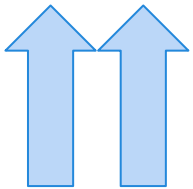


h1 h2

1. Initialize signatures with inf
2. For every element indexed with ix:
   a. For every perumutation:
      i. For every document:
         1. If element not present in a doc skip, otherwise:
         2. signature = min(signature, ix2(ix))

Algorithm sample

lx		S1	S2	S3	S4	lx2	lx2
0	a	0	0	1	0	1	1
1	g	0	1	1	0	2	4
2	i	1	1	1	1	3	2
3	k	1	1	0	0	4	0
4	p	0	0	0	1	0	3



h1 h2

	S1	S2	S3	S4
h1	inf	inf	inf	inf
h2	inf	inf	inf	inf

	S1	S2	S3	S4
h1	inf	inf	1	inf
h2	inf	inf	1	inf

	S1	S2	S3	S4
h1	inf	2	1	inf
h2	inf	4	1	inf

	S1	S2	S3	S4
h1	3	2	1	3
h2	2	2	1	2

	S1	S2	S3	S4
h1	3	2	1	3
h2	0	0	1	2

	S1	S2	S3	S4
h1	3	2	1	0
h2	0	0	1	2

LSH

Number of pairs problem

- We know how to compare 2 documents in a short (for H fixed = O(1)) time

But still:

- with 10M documents we have ~10M^2 pairs = few days of computation
- Solution: LSH

LSH: General idea

- Use a function f(S1,S2) that tells whether S1 and S2 is a candidate pair
- candidate pair = a pair of elements whose similarity must be evaluated

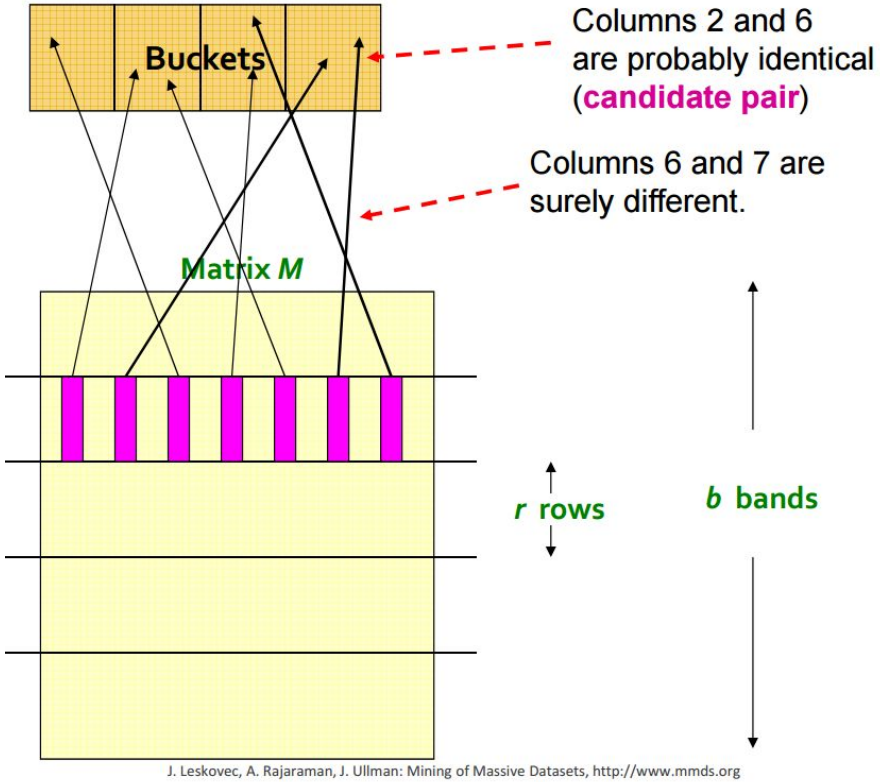
  (to check if they are really similar)

  Number of candidate pairs << number of all pairs
- For some **similarity threshold t** we want
  - Almost all pairs S1, S2 with s(S1,S2) >= t
  - Almost none pairs with s(S1,S2) < t

LSH: algorithm

- Input: Signature Matrix (Docs x Signatures)
- Output: list of candidate pairs S1, S2 to be later evaluated with s(S1,S2)

Overview & algorithm



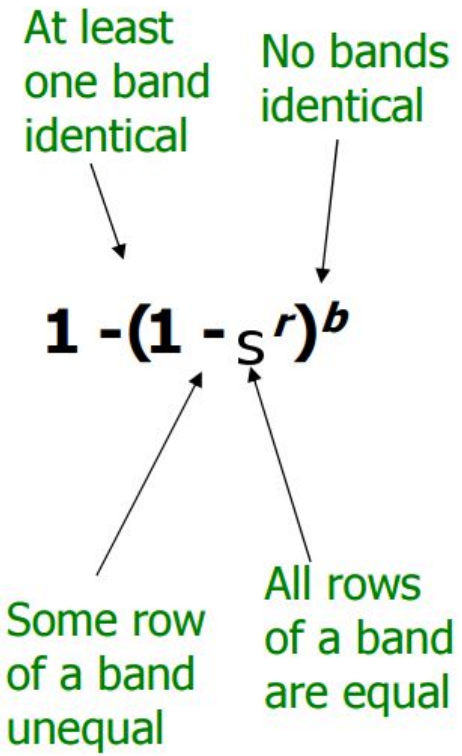
- Split M into b bands with r rows each
- For each doc (column):
  - For each band:
    - Hash band signature part into one of buckets
- Go over all buckets for all bands:
  - If there are two docs in one bucket
    -> candidate pair

Analysis *

Probability that signatures between S1,S2 **agree in one row: s** (like in MinHash)

What's the **probability** that S1,S2 will become **a candidate pair**?

Analysis *


At least one band identical
No bands identical
$1 - (1 - s^r)^b$
Some row of a band unequal
All rows of a band are equal

Analysis *

Probability that signatures between S1,S2 **agree in one row: s**

What's the **probability** that S1,S2 will become **a candidate pair**?

$$P(\text{candidate}) = 1 - (1-s^r)^b$$

Example: calc one curve, r=5, b=10 *

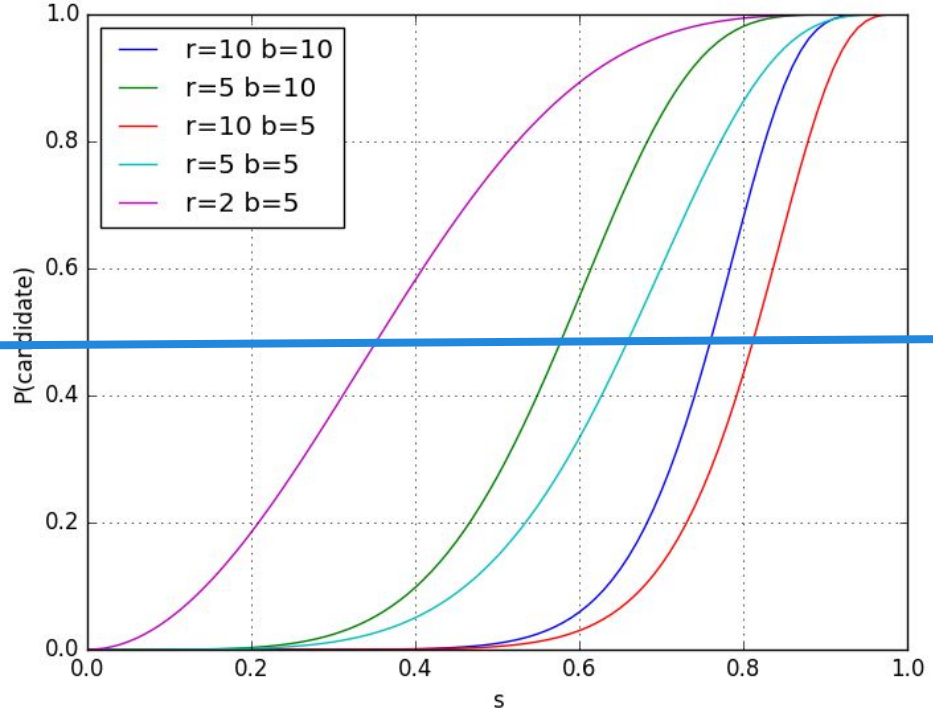
```
import numpy as np


s = np.array(range(0,5))/10
pc = lambda r,b: 1-(1-s**r)**b

print(list(s))
print(list(pc(5,10)))
```

Analysis *

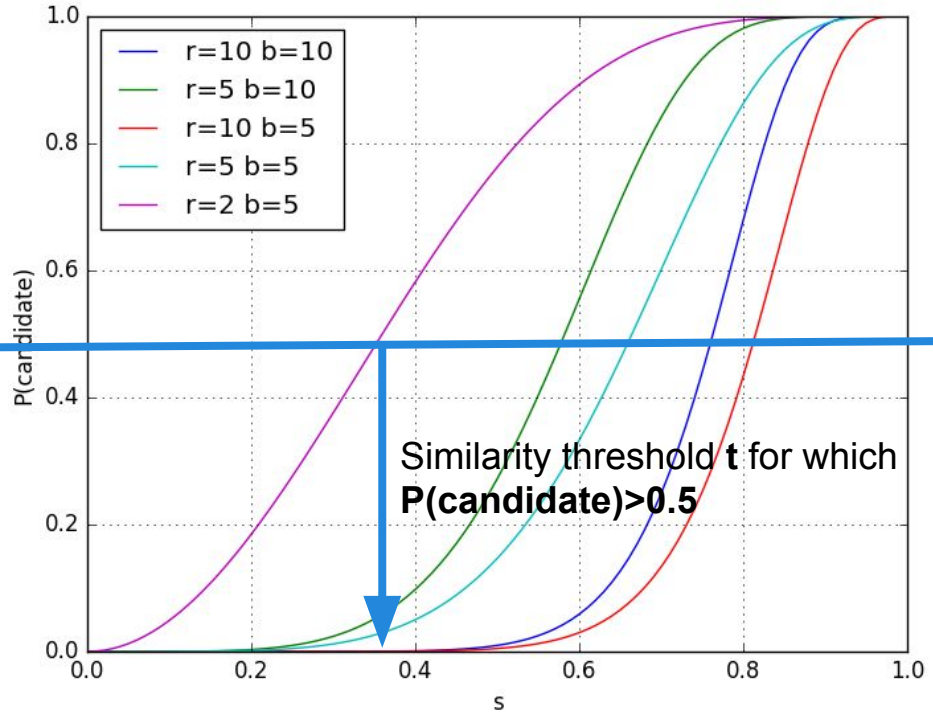
Pairs more likely to be found



Pairs that most likely will not be find

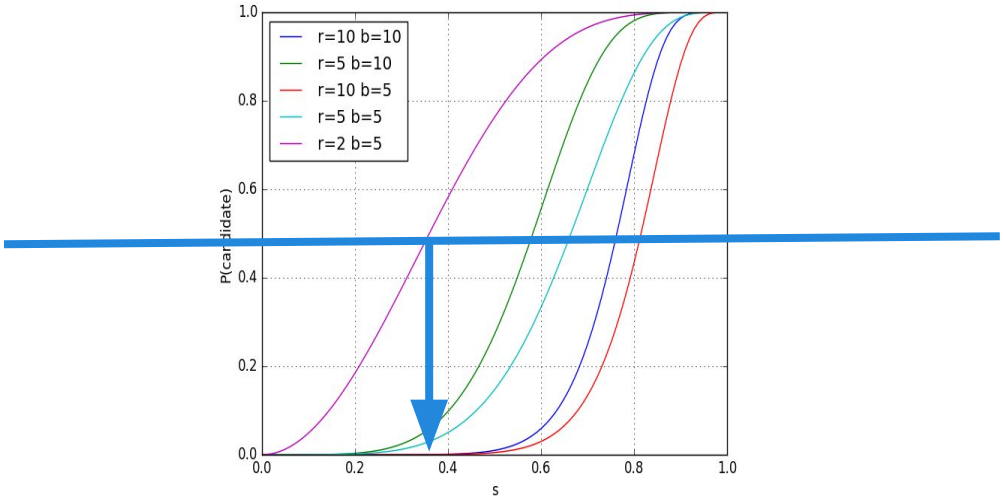
Analysis *

Pairs more likely to be found



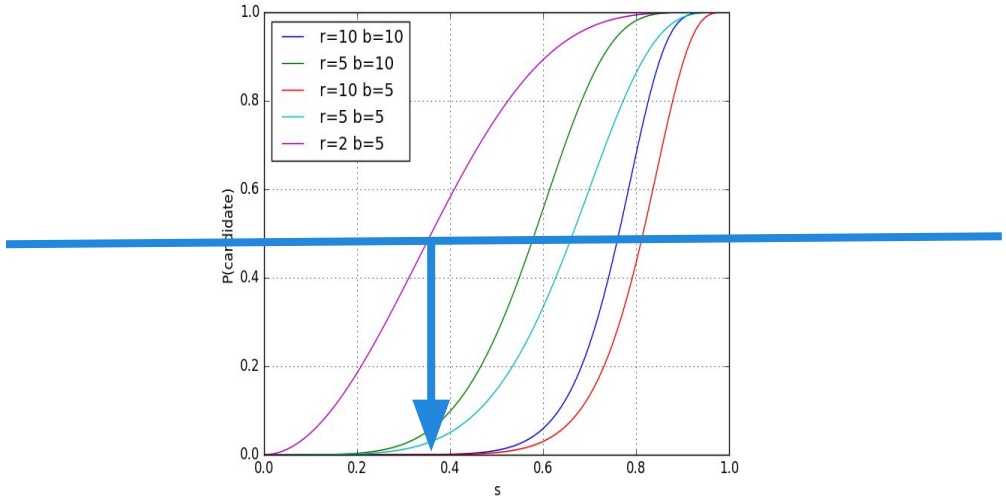
Pairs that most likely will not be find

Analysis: How to find t? *



(*blackboard*)

Analysis: How to find t? *

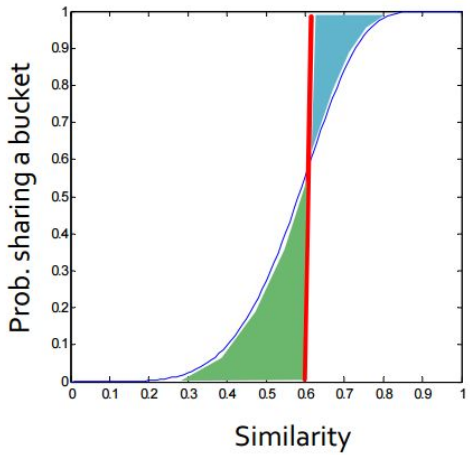


$t \sim (1/b)^{1/r}$

Fix t and b/r
-> get the rest
(blackboard example)

False positives, false negatives

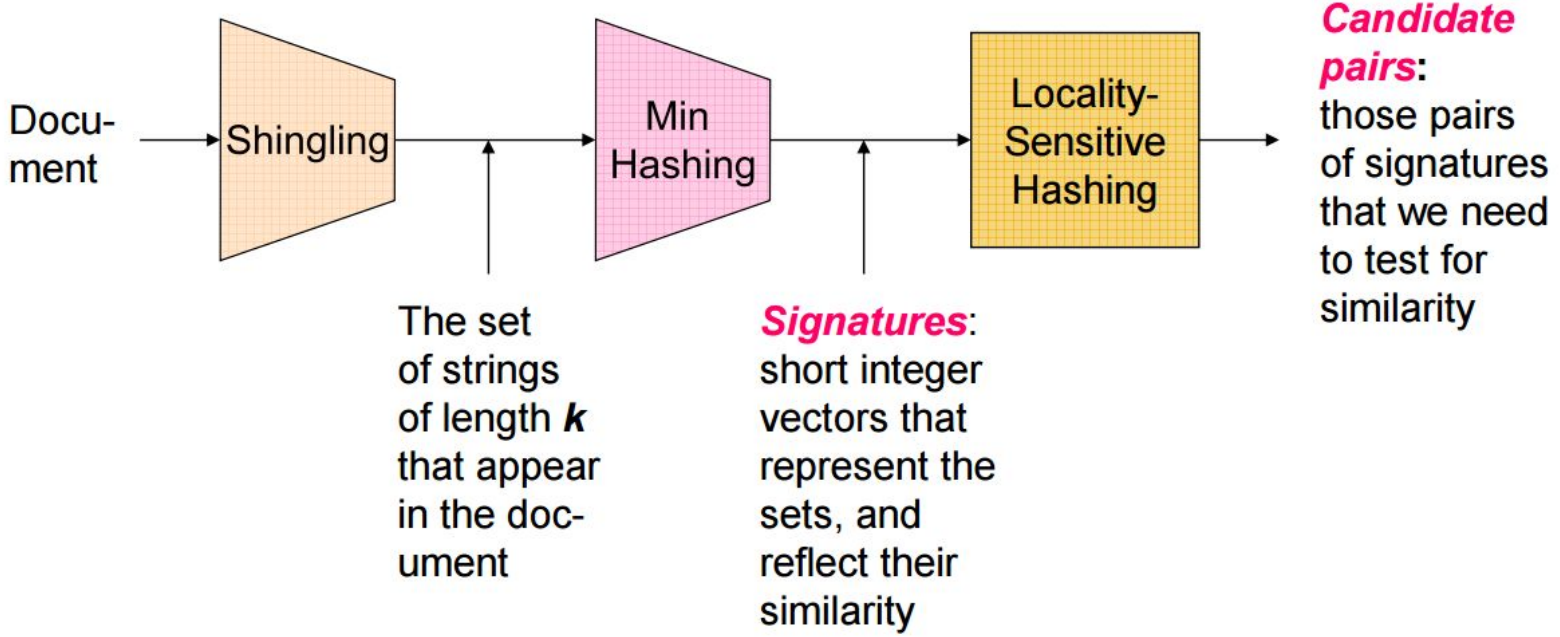
- **Picking *r* and *b* to get the best S-curve**
  - 50 hash-functions (r=5, b=10)



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

http://www.mmds.org/mmds/v2.1/ch03-lsh.pdf

Final pipeline



Assignment

Given the following documents:
D1 = be or not to be
D2 = to be two bees
D3 = not to bees

1) Represent them using bigrams.
2) Calculate Jaccard Similarity between them
3) Represent as bit-vectors (vector positions should be ordered alphabetically for example: “be to” before “to be” as long as ‘b’<’t’, “be to” before “be too” as long as ‘’<’o etc.)
4) Store as a matrix
5) calculate MinHash signatures for two permutations:
   a) Identity permutation ix2(ix) = ix
   b) ix2(ix) = (3*ix+1) mod 7
6) estimate Jaccard Similarity from MinHash signatures and compare to true values

Solution 1

Given the following documents:
D1 = be or not to be
D2 = to be two bees
D3 = not to bees

1) Represent them as using bigrams.
   S1 = {be_or, or_not, not_to, to_be}
   S2 = {to_be, be_two, two_bees}
   S3 = {not_to, to_bees}
2) Calculate Jaccard Similarity between them
   J(S1,S2) = 1 / 6
   J(S1,S3) = 1 / 5
   J(S2,S3) = 0

Solution 2

3) Represent as bit-vectors:

Elements ordered alphabetically along with vector positions:
be_or, be_two, not_to, or_not, to_be, to_bees, two_bees
0 1 2 3 4 5 6

Pos: 0 1 2 3 4 5 6
S1 = $(1\ 0\ 1\ 1\ 1\ 0\ 0)^T$
S2 = $(0\ 1\ 0\ 0\ 1\ 0\ 1)^T$
S3 = $(0\ 0\ 1\ 0\ 0\ 1\ 0)^T$

Solution 3

4) Store as matrix

Pos: 0 1 2 3 4 5 6

S1 = $(1\ 0\ 1\ 1\ 1\ 0\ 0)^T$

S2 = $(0\ 1\ 0\ 0\ 1\ 0\ 1)^T$

S3 = $(0\ 0\ 1\ 0\ 0\ 1\ 0)^T$

Ix	S1	S2	S3
0	1	0	0
1	0	1	0
2	1	0	1
3	1	0	0
4	1	1	0
5	0	0	1
6	0	1	0

Solution 4

5) calculate MinHash signatures for two permutations:

a) Identity permutation ix2(ix) = ix

Ix	S1	S2	S3	Ix2
0	1	0	0	0
1	0	1	0	1
2	1	0	1	2
3	1	0	0	3
4	1	1	0	4
5	0	0	1	5
6	0	1	0	6

	S1	S2	S3
h1	0	1	2

Solution 4

5) calculate MinHash signatures for two permutations:

b) ix2(ix) = (3*ix+1) mod 7

Ix	S1	S2	S3	Ix2
0	1	0	0	1
1	0	1	0	4
2	1	0	1	0
3	1	0	0	3
4	1	1	0	6
5	0	0	1	2
6	0	1	0	5

Ix	S1	S2	S3	Ix2
2	1	0	1	0
0	1	0	0	1
5	0	0	1	2
3	1	0	0	3
1	0	1	0	4
6	0	1	0	5
4	1	1	0	6

	S1	S2	S3
h2	0	4	0

Solution 5

	S1	S2	S3
h1	0	1	2

	S1	S2	S3
h2	0	4	0

	S1	S2	S3
h1	0	1	2
h2	0	4	0

6) estimate Jaccard Similarity from MinHash signatures and compare to true values:

J'(S1,S2) = 0 < J(S1,S2) = 1 / 6

J'(S1,S3) = ½ > J(S1,S3) = 1 / 5

J'(S2,S3) = 0 = J(S2,S3) = 0