Finding Similar Items

INF 553

Roadmap

- Problem formulation
- Applications
  - Collaborative filtering
    - Buyer-to-item and item-item recommendation
  - Finding similar web pages
    - Shingles
- Minhash signatures
- Locality-sensitive hashing

Problem Formulation

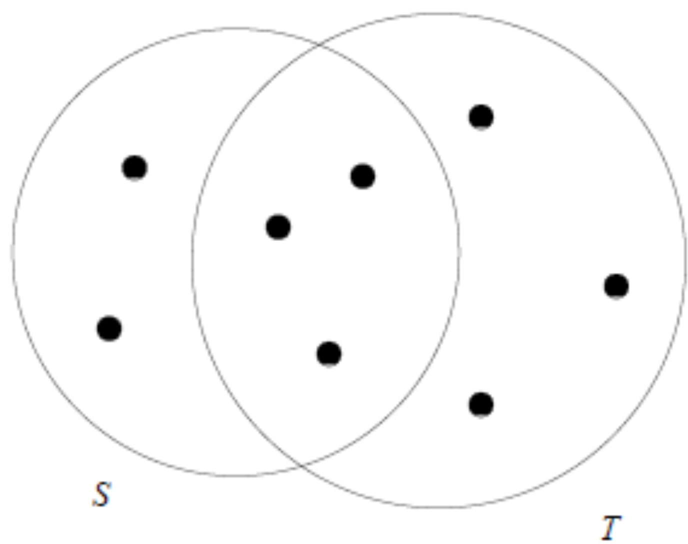
- Item represented as a set of objects

- Problem becomes: find similar sets

- Challenges:
  - Large sets
  - Large number of items/sets

Similarity of Sets

- Jaccard similarity
  - Size of intersection / size of union
  - Jaccard(S, T) = $|S \cap T| / |S \cup T|$



Finding Similar Buyers

Who is most similar to B1?

B1	{A, B}
B2	{A, B, D}
B3	{A, C, D}

Jaccard(B1, B2) =
Jaccard(B1, B3) =

Who is most similar to B2?

Jaccard(B2, B1) =
Jaccard(B2, B3) =

Who is most similar to B3?

Jaccard(B3, B1) = Jaccard(B1, B3)
Jaccard(B3, B2) = Jaccard(B2, B3)

Finding Similar Buyers

Who is most similar to B1?

B1	{A, B}
B2	{A, B, D}
B3	{A, C, D}

Buyer-to-item recommendation

Jaccard(B1, B2) = 2/3
Jaccard(B1, B3) = 1/4

B2 most similar to B1
B2 also bought D

Recommend D to B1

Who is most similar to B2?

Jaccard(B2, B1) = Jaccard(B1, B2)
Jaccard(B2, B3) = ?

Who is most similar to B3?

Jaccard(B3, B1) = Jaccard(B1, B3)
Jaccard(B3, B2) = Jaccard(B2, B3)

Finding Similar Items

B1	{A, B}
B2	{A, B, D}
B3	{A, C, D}

Buyer as a set

A	{B1, B2, B3}
B	{B1, B2}
C	{B3}
D	{B2, B3}

Item as a set

Finding Similar Items

Which item is most similar to A?

A	{B1, B2, B3}
B	{B1, B2}
C	{B3}
D	{B2, B3}

Jaccard(A, B) =
Jaccard(A, C) =
Jaccard(A, D) =

Which item is most similar to B?

Jaccard(B, A) =
Jaccard(B, C) =
Jaccard(B, D) =

B1	{A, B}
B2	{A, B, D}
B3	{A, C, D}

Finding Similar Items

Which item is most similar to A?

A	{B1, B2, B3}
B	{B1, B2}
C	{B3}
D	{B2, B3}

Jaccard(A, B) = 2/3
Jaccard(A, C) = 1/3
Jaccard(A, D) = 2/3

A is most similar to B and D

Which item is most similar to B?

Jaccard(B, A) = Jaccard(A, B) = 2/3
Jaccard(B, C) = 0
Jaccard(B, D) = 1/3

B is most similar to A

Item-to-item recommendation

B1	{A, B}
B2	{A, B, D}
B3	{A, C, D}

B1 has bought A, B (or put them in basket)
A is most similar to B and D
B is most similar to A

Recommend: D to B1

Formulated as frequent itemset prob.?

- Find similar items
  - Items bought together by many users

- Find similar users
  - Users that bought many common items

Formulated as frequent itemset prob.?

- Find similar items
  - Items bought together by many users

  $\Rightarrow$User = transaction

  $\Rightarrow$Similar items = frequent item pair

B1	{A, B}
B2	{A, B, D}
B3	{A, C, D}

- Find similar users
  - Users that bought many common items
  - Item = transaction

  => Find frequent user pairs

A	{B1, B2, B3}
B	{B1, B2}
C	{B3}
D	{B2, B3}

Finding Frequent Item Pairs

B1	{A, B}
B2	{A, B, D}
B3	{A, C, D}

Sup(A, B) = 2
Sup(A, C) = 1
Sup(A, D) = 2
Sup(B, C) = 0
Sup(B, D) = 1
Sup(C, D) = 1

min_sup = 2

(A, B)
(A, D)

Sup(A) = 3
Sup(B) = 2
Sup(C) = 1
Sup(D) = 2

min_sup = 2

prune

Sup(A) = 3
Sup(B) = 2
Sup(D) = 2

Candidate 2-itemsets

(A,B)
(A,D)
(B,D)

Apriori algorithm

So why new solution?

- How about just use Apriori to find
  - Frequent item pairs => similar items

B1	{A, B}
B2	{A, B, D}
B3	{A, C, D}

  - Frequent user pairs => similar users

A	{B1, B2, B3}
B	{B1, B2}
C	{B3}
D	{B2, B3}

Potential Problems

- Are buyer B1 and B2 still similar?
- What would Apriori say?

B1	{A, B}
B2	{A, B, D}
B3	{A, C, D}
B4	{E, F}
B5	{E, F, G}
B6	{X, Y, Z}
B7	{X, W}
B8	{S, T, O}
B9	{S, T}
…	…

Potential Problems

- Are buyer B1 and B2 still similar?
- What would Apriori say?
  - Form item->buyers basket
    - A large number of items
  - B1 and B2 only appear together in ? baskets
    - Hence very low support
  - But they are still similar

B1	{A, B}
B2	{A, B, D}
B3	{A, C, D}
B4	{E, F}
B5	{E, F, G}
B6	{X, Y, Z}
B7	{X, W}
B8	{S, T, O}
B9	{S, T}
…	…

Roadmap

- Problem formulation
- Applications
  - Collaborative filtering
    - Buy-to-item and item-item recommendation
  - Finding similar web pages
    - Shingles
- Minhash signatures
- Locality-sensitive hashing

Application: Find Similar Web Pages

- Similar = near-duplicate
  - Plagiarism
  - Mirror pages
  - News from same AP

- Can we use Apriori?
  - Web page = ?
  - Basket = ?
  - Item = ?

Application: Find Similar Web Pages

- Similar = near-duplicate
  - Plagiarism
  - Mirror pages
  - News from same AP

- Can we use Apriori?
  - Web page = a set of words/sentences
  - Basket = word/sentence
  - Item = web page

- Problems?

Problems

- Can we use Apriori?
  - Web page = a set of words/sentences
  - Basket = word/sentence
  - Item = web page

- Problems?
  - Low support but still similar
  - Word as basket: too fine; sentence? too coarse

Application: Find Similar Web Pages

- Similar = near-duplicate
  - Plagiarism
  - Mirror pages
  - News from same AP

- Web page = a set of shingles

Shingles

- Web page as a string of characters
- Shingle = subsequence of k-characters

- Web page = abcdabd, k = 2
- 2-shingles
  - ab, bc, cd, da, bd

- Max # of k-shingles for a page of n characters?

White Spaces

- Better not omit them
- Could turn multiple into one

- D1: “scored a touch   down” => “scored a touch down”
- D2: “touchdown at last”

- D1 and D2 have a common 9-shingle if space omitted

Shingle Size

- Too small
  - Many documents will falsely become similar

- Too big
  - Might miss truly similar documents

Example

- Doc = email, k = 5, character = letter + space
  - \# of possible 5-shingles = 27^5 ~ 14M
- Suppose email is N character long
  - Probability of a shingle appearing in the email?

~ N * 1/14M << 1

Rule of Thumb

- k should be picked large enough that
  – the probability of any given shingle appearing in any given document is low

- Caveat to previous example
  – Some letters/space are more common than others
  $\Rightarrow$Some shingles occur more often than others
  $\Rightarrow$May ignore less frequent ones (say # possible chars = 20 instead of 27)

Challenges

- A large number of shingles
  - Computationally expensive to compute Jaccard
  - Huge storage overhead

- Storage overhead
  - Millions of documents, 4K/document

=> Reduce large sets into small signatures

Roadmap

- Problem formulation
- Applications
  - Collaborative filtering
    - Buy-to-item and item-item recommendation
  - Finding similar web pages
    - Shingles
- Minhash signatures
- Locality-sensitive hashing

Similarity-Preserving Signatures

- Document D => S(D) = signature of D

- Goal: If D1 ~ D2, then S(D1) ~ S(D2)

Matrix Representations of Sets

- Characteristic matrix of sets

Element	S_1	S_2	S_3	S_4
a	1	0	0	1
b	0	0	1	0
c	0	1	0	1
d	1	0	1	1
e	0	0	1	0

Universal set: {a, b, c, d, e}

Minhashing a Set

- Pick a permutation of rows
- Minhash(S_j) = no. of 1^{st} row i where S[i, j] = 1

Element	S_1	S_2	S_3	S_4
b	0	0	1	0
e	0	0	1	0
a	1	0	0	1
d	1	0	1	1
c	0	1	0	1

- Simply denote minhash() as h()
- $h(S_1) = a$, $h(S_2)=c$, $h(S_3) = b$, $h(S_4) = a$

Minhash and Jaccard Similarity

- On a random permutation of rows
  - $Prob(h(S_i) = h(S_j)) = Jaccard(S_i, S_j)$
  - Explained next
- Minhash is a locality-sensitive "hash" function
  - Normally, hash function will place similar items in every buckets
  - But here, similar items are placed in the same bucket with high probability

Intuition

- Consider Prob($h(S_i) = h(S_j)$)

- Rows of S_i and S_j:
  - Case X: both 1
  - Case Y: one 1, one 0
  - Case Z: both 0

Element	S_1	S_2	S_3	S_4
b	0	0	1	0
e	0	0	1	0
a	1	0	0	1
d	1	0	1	1
c	0	1	0	1

Intuition

- Proceed from top
  - Type Z row, ignore
  - Type X row, stop
  - Type Y row,
    stop

Element	S_1	S_2	S_3	S_4
b	0	0	1	0
e	0	0	1	0
a	1	0	0	1
d	1	0	1	1
c	0	1	0	1

- Prob. of seeing a type X row ($h(S_i) = h(S_j)$) before Y row ($h(S_i) \neq h(S_j)$) = $x/(x+y)$

Minhash Signature

- "Hash" (actually just permute rows of) sets multiple times
- Record first bucket number (0~4 in this case) of sets where 1 appears

Row	S_1	S_2	S_3	S_4
0	1	0	0	1
1	0	0	1	0
2	0	1	0	1
3	1	0	1	1
4	0	0	1	0

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

Computing Minhash Signature

- Costly to explicitly permute millions of rows
- Use randomly chosen hash function h() on the row ID
  - n rows, h(x) = 0 ~ n - 1

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

Permutation via Hashing

- h(r) permutes row r to h(r)-th row in permuted order

$$h1(x) = (x + 1) \bmod 5$$

Row	**S1**	**S2**	**S3**	**S4**
0	1	0	0	1
1	0	0	1	0
2	0	1	0	1
3	1	0	1	1
4	0	0	1	0

New Row #	**Old row #**	**S1**	**S2**	**S3**	**S4**
0	**4**	0	0	1	0
1	**0**	1	0	0	1
2	**1**	0	0	1	0
3	**2**	0	1	0	1
4	**3**	1	0	1	1

old row 0 -> new row 1, …

Minhash Values using h1()

- Use new numbers
  - h1(S1) = 1, h1(S2) = 3, h1(S3) = 0, h1(S4) = 1

New Row #	**Old row #**	**S1**	**S2**	**S3**	**S4**
0	**4**	0	0	1	0
1	**0**	1	0	0	1
2	**1**	0	0	1	0
3	**2**	0	1	0	1
4	**3**	1	0	1	1

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

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

⇨

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

⇨

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

⇨

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

⇨

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

⇨

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

Estimate Similarity Using Signatures

Row	S_1	S_2	S_3	S_4
0	1	0	0	1
1	0	0	1	0
2	0	1	0	1
3	1	0	1	1
4	0	0	1	0

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

There may be many pairs

Pair	Actual Sim.	Estimated Sim.
(S1, S2)	0	0
(S1, S3)	1/4	1/2
(S1, S4)	2/3	1
(S2, S3)	?	?
(S2, S4)	?	?
(S3, S4)	?	?

Estimate Similarity Using Signatures

Row	S_1	S_2	S_3	S_4
0	1	0	0	1
1	0	0	1	0
2	0	1	0	1
3	1	0	1	1
4	0	0	1	0

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

There may be many pairs

Pair	Actual Sim.	Estimated Sim.
(S1, S2)	0	0
(S1, S3)	1/4	1/2
(S1, S4)	2/3	1
(S2, S3)	0	0
(S2, S4)	1/3	0
(S3, S4)	1/5	1/2

Challenge: Large # of Comparisons

- So we have reduced large sets into small signatures

- But we still have a large number of pairs to compare
  – 1 million documents
  => Half a trillion pairs
  – 2 microseconds /pair
  => Need 6 days to compute similarities

Roadmap

- Problem formulation
- Applications
  - Collaborative filtering
    - Buy-to-item and item-item recommendation
  - Finding similar web pages
    - Shingles
- Minhash signatures
- Locality-sensitive hashing

Idea of LSH

- Hash again!
  - This time, hash the signatures instead
  - If two signatures are similar, they should be in the same bucket with high probability
    - H(sig1) = H(sig2) if sig1 ~= sig2
  - Only estimate similarity for sets in the same bucket

- Recall “frequent bucket” idea in PCY algorithm
  - If an item pair is not hashed into a “frequent bucket”
  - We know it will be not frequent

Define Similar Signatures

- Two sets are similar
  – If their Jaccard similarity is greater than a threshold, say .8

- If two sets are similar, their signatures should be similar too
  – What do we mean by “similar” signatures?

Define Similar Signatures

- Which two signatures are similar?

S1'	S2'	S3'	S4'
0	2	0	0
1	0	1	0
2	1	2	0
1	0	1	1
0	0	0	0
2	0	0	1

Define Similar Signatures

- Two signatures are similar if they have the same value (can be 0, 1, …, n – 1, where n = # of elements in the universal set) at many rows
  - E.g., if similarity of two sets is .8, then 80% of rows in their signatures should have the same value

S1'	S2'	S3'	S4'
0	2	0	0
1	0	1	0
2	1	2	0
1	0	1	1
0	0	0	0
2	0	0	1

Signatures

Recall Similarity of Sets

- Two sets are similar if they have the same value (only 1's, ignore 0's) at many rows of characteristic matrix

Row #	S1	S2	S3	S4
0	0	1	0	1
1	1	0	1	1
2	0	0	0	1
3	0	1	0	0
4	1	1	1	1
5	0	1	1	0

Characteristic matrix

S1'	S2'	S3'	S4'
0	2	0	0
1	0	1	0
2	1	2	0
1	0	1	1
0	0	0	0
2	0	0	1

Signatures

Problem

- Signatures are not really sets (or even multisets)
  - Rather they are strings of symbols
  - Signatures of sets are strings of the same length
  - i-th symbols of signatures correspond to each other

Row #	S1	S2	S3	S4
0	0	1	0	1
1	1	0	1	1
2	0	0	0	1
3	0	1	0	0
4	1	1	1	1
5	0	1	1	0

Characteristic matrix

S1’	S2’	S3’	S4’
0	2	0	0
1	0	1	0
2	1	2	0
1	0	1	1
0	0	0	0
2	0	0	1

Signatures

String-Based Hash Function

- Minhash is really a set-based hash
- We need to hash signatures, i.e., strings of same length, instead
  - So that if two strings have the same symbol at many corresponding positions
  - They will be very likely placed in the same bucket
- Can you find such a hash function?

Idea

- Need to determine bucket for a signature
  - by comparing its symbols with the corresponding symbols for the signatures in a bucket
  - Amount to computing all pairwise similarities
  - This beats the purpose of hashing

- Can we take sample of rows in signature instead?
  - Hashing samples should be less costly

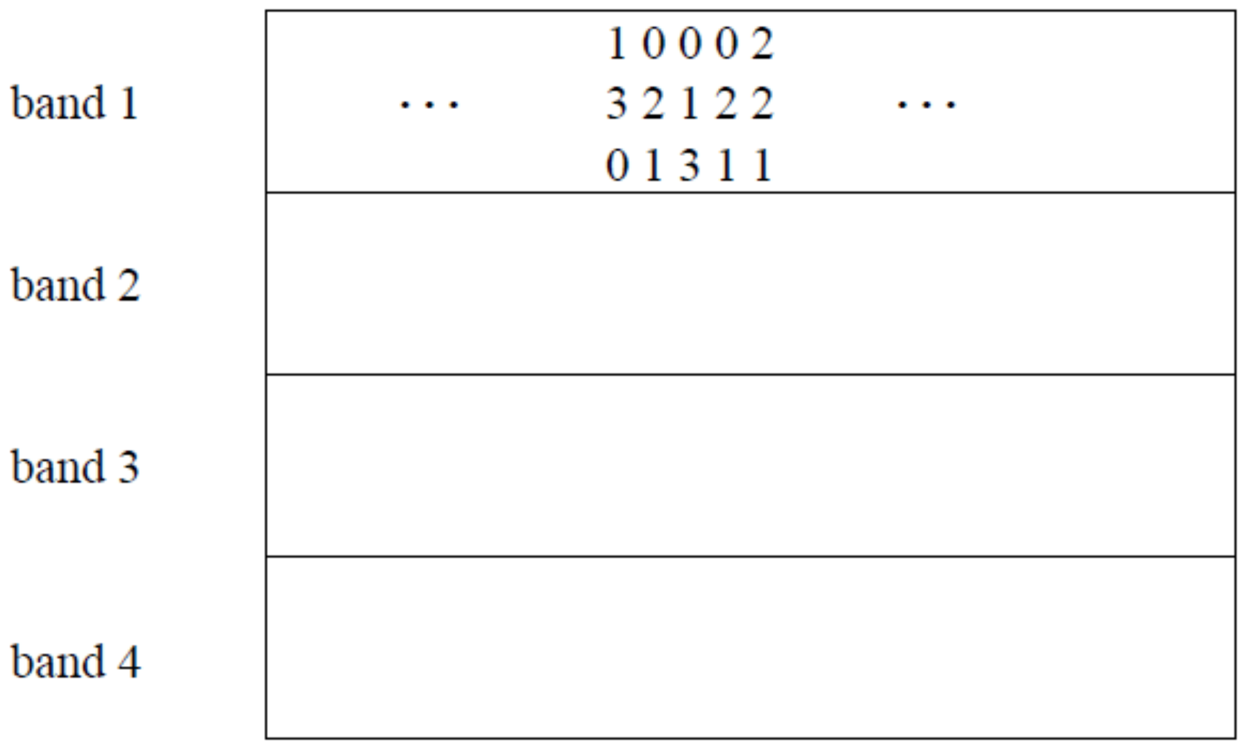
- Similar to sampling in "limited pass" algorithm in frequent item discovery

Sampling & Banding

- If two signatures are similar, their samples should be too
- Divide signature into "bands"
  - Hash each band individually (band ~ sample)
  - Will *consider pairs* if they are hashed into the same bucket in at least one band
    - (recall the multi-hash idea)
  - Such pairs are called "candidate pairs"
    - (for computing similarities later)

Band-Based Hashing

- 4 bands, 3 rows/each
- Vector = part of signature in a band
  - E.g., (1, 3, 0), (0, 2, 1), …



Band-Based Hashing

- Two vectors hashed to same bucket if and only they are identical
  - Require symbol-by-symbol comparisons
  - E.g., Cols 0, 2, 1 and 0, 2, 1

- Two signatures are candidate pairs as long as they agree on all rows in at least one band

Prob. of Becoming a Candidate Pair

- Two sets X and Y, Jaccard(X, Y) = s
- Prob. of minhash signatures of X & Y agree on any row of signature matrix = s
- Signature divided into b bands, r rows/band
- What is the prob. of two signatures agree on all rows in at least one band?

Calculating Probability

- What is the prob. that two signatures agree on all rows in at least one band?
  - Candidate pairs
  - Need to consider many cases:
    - E.g., signatures agree in 1, 2, …, b bands
- Compute complemental prob. first
  - p’ = Prob. that they disagree on at least one row in all bands
  - Final prob. p = 1 – p’

Derivations

- p' = Prob. that they disagree on at least one row in all bands
  - q = Prob. that they disagree in at least one row in a band
  - $p' = q^b$
  - Prob. of all rows in a band agree is s^r (why?)
  - r is the number of rows in a band
- $q = 1 - s^r$

$\Rightarrow p = 1 - (1 - s^r)^b$

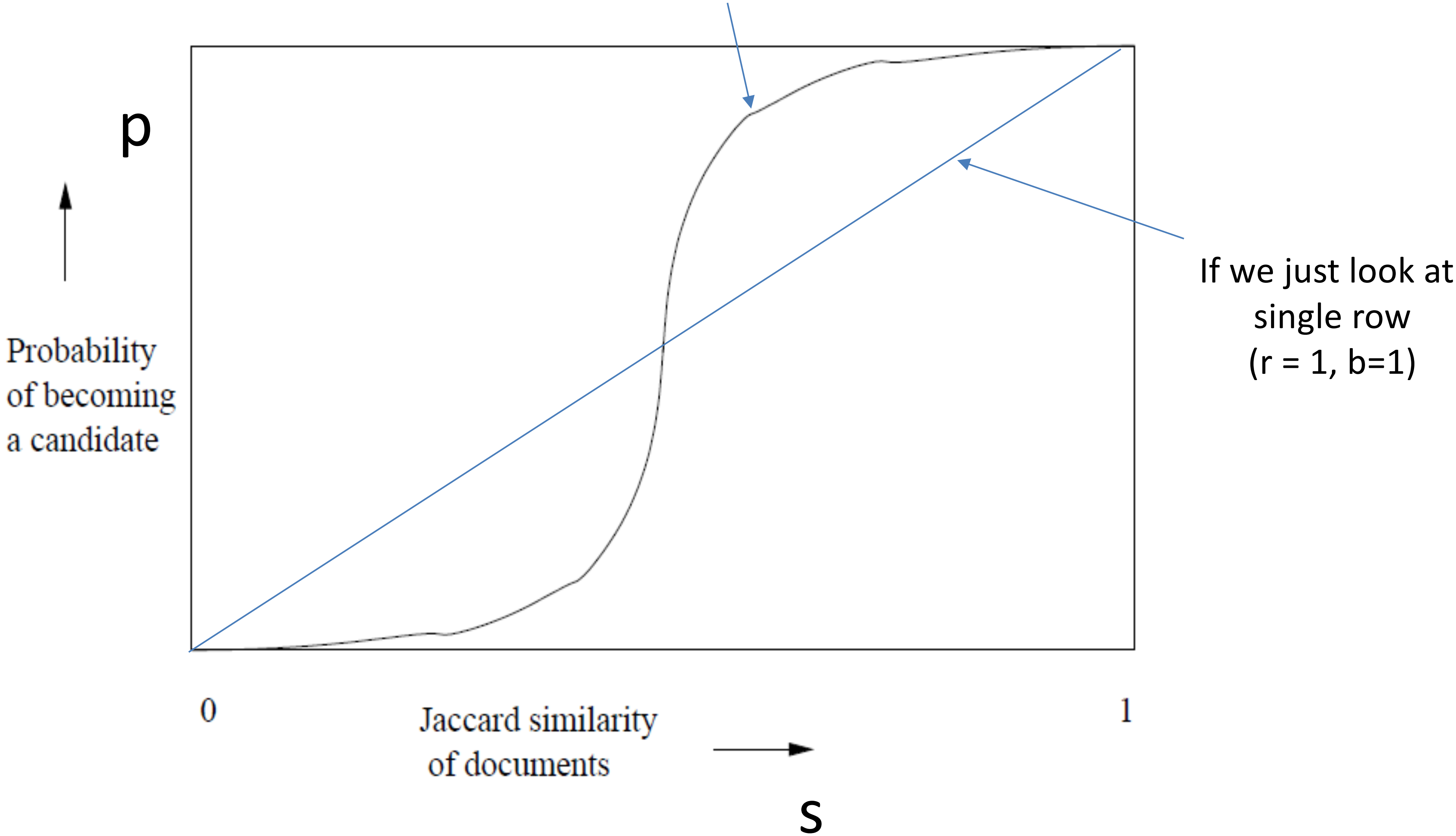
Put Together

- Prob. of agreeing on all rows of a band = s^r
  - E.g., 0, 2, 1 and 0, 2, 1
- Prob. of disagreeing on at least one row of a band = $1 – s^r$
  - E.g., 0, 2, 1 and 2, 2, 1

- Prob. of disagreeing on at least one row in all bands
  - $(1 – s^r)^b$

- Prob. of agreeing on all rows in at least one band
  - $1 - (1 – s^r)^b$

$$p = 1 - (1 - s^r)^b, \; b = 20, \; r = 5$$



Questions

- Relationship btw. curve shapes and errors
  - Spot false positives and negatives
- How does the curve change its shape? when
  - r = b = 1
  - r varies, b = 1
  - r = 1, b varies
  - both b and r > 1
- How to determine good similarity threshold? When
  - b * r = (fixed) n
  - n = length of signature

How to find a good b & r

- For b=20 and r=5, at S=0.8
  - 1-(0.8)^5 = 0.672…(1-33%)
  - 0.672^20 = 0.00035…(1/3000)
- For two documents with 80% Jaccard similarity:
  - 33% chance in agreeing in all 5 rows
  - 1 in 3000 pairs will fail to become a candidate pair

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

Confusion Matrix

	P (predicted)	N (predicted)
P (Actual)	TP	FN
N (Actual)	FP	TN

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

Confusion Matrix

	P (predicted)	N (predicted)
P (Actual)	TP	FN
N (Actual)	FP	TN

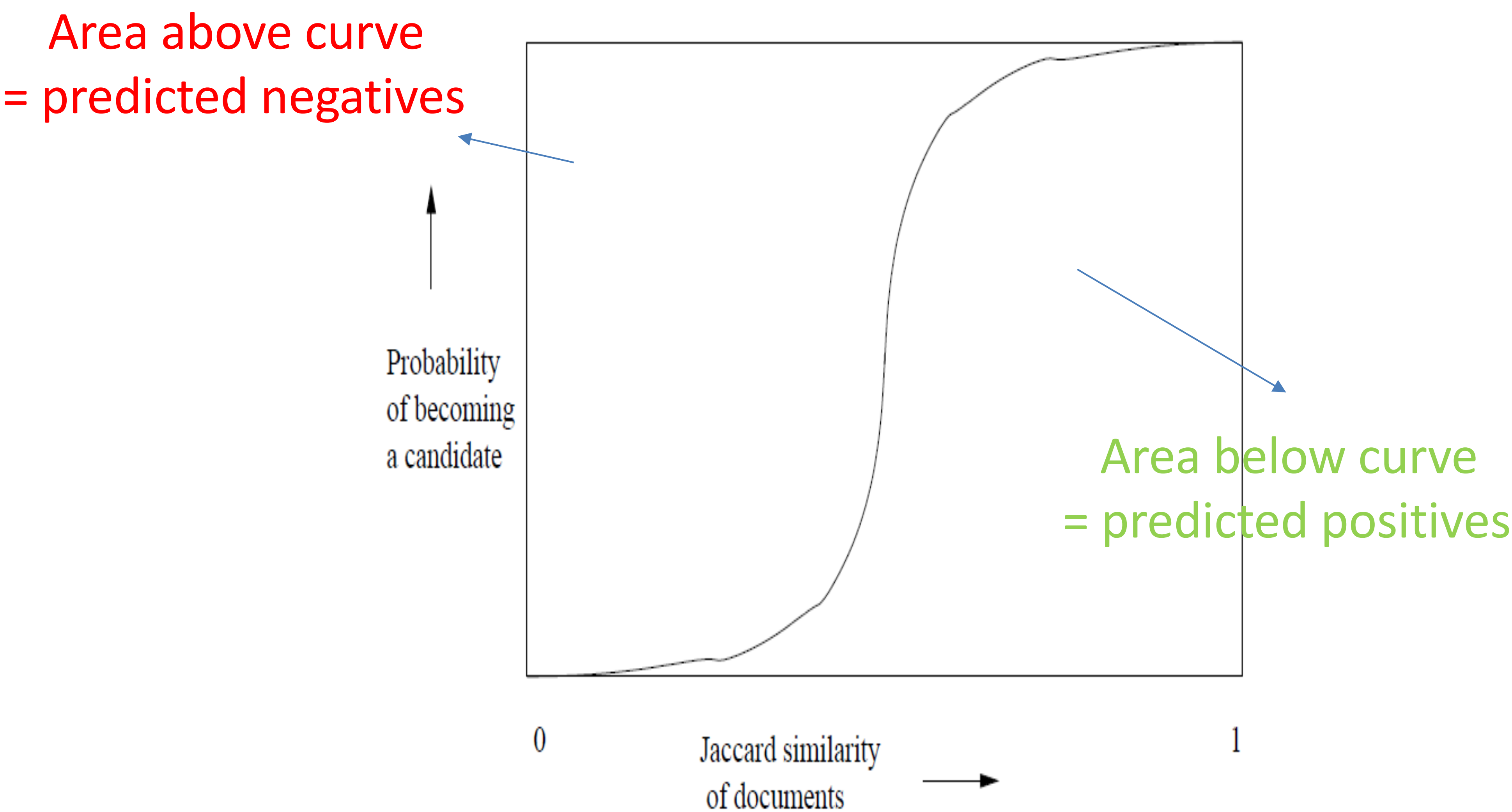
$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

When FP goes down, precision will go up or down?
What about FN?

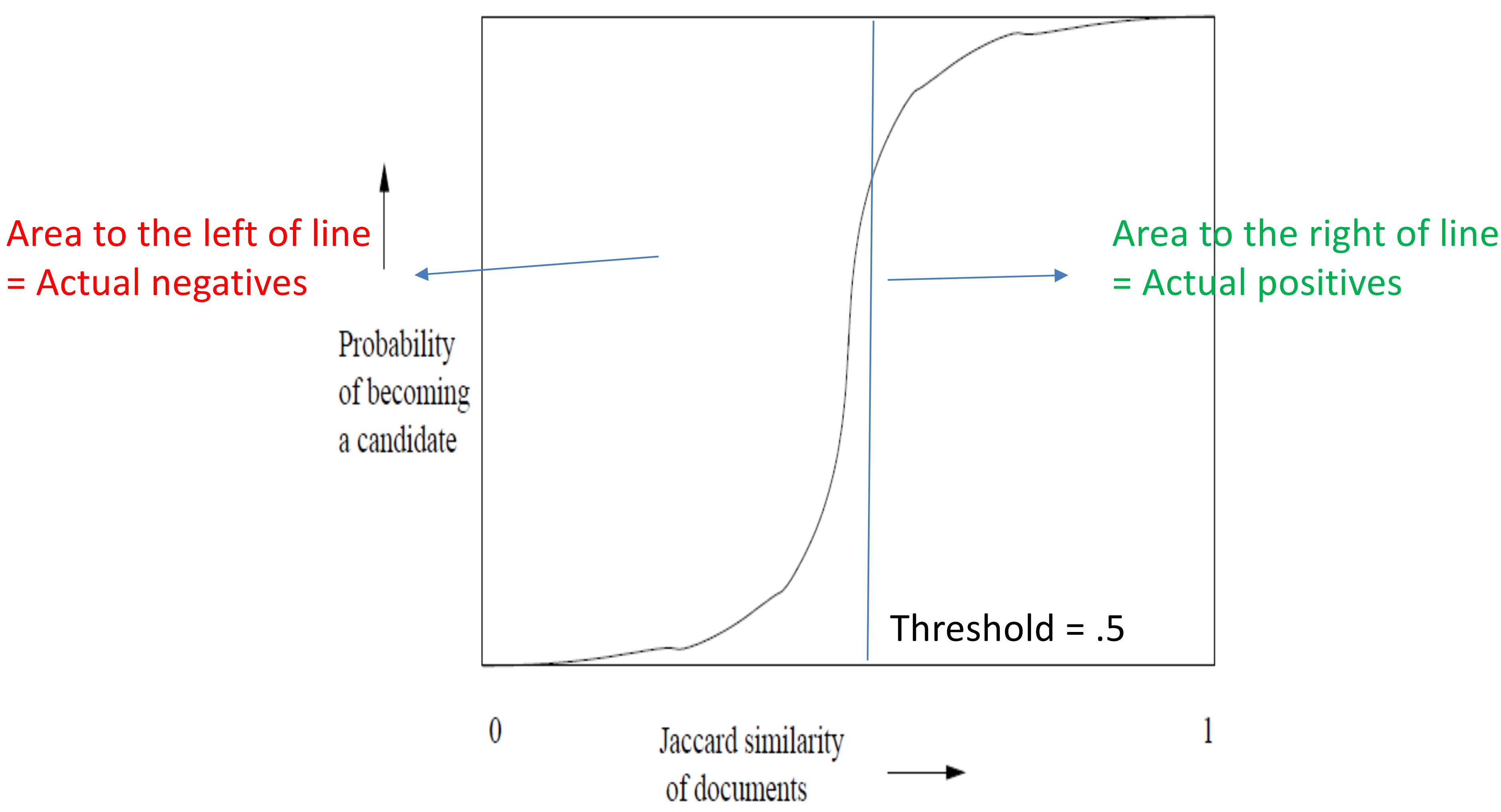
False Positives and Negatives

- False positives: dissimilar but hashed to same bucket
- False negatives: similar but not hashed to different bucket

Predicted Positives and Negatives

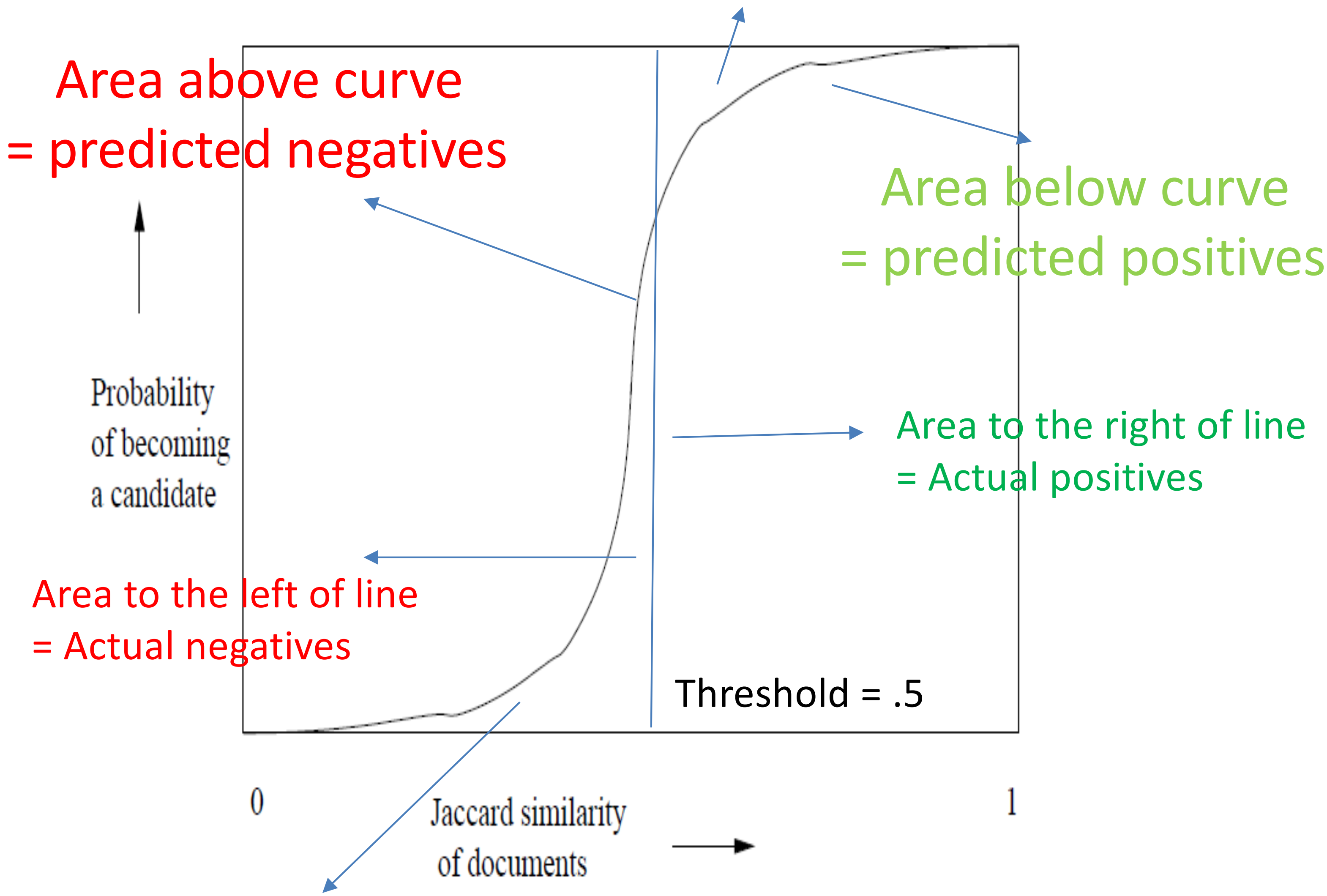


Actual Positives and Negatives



Errors

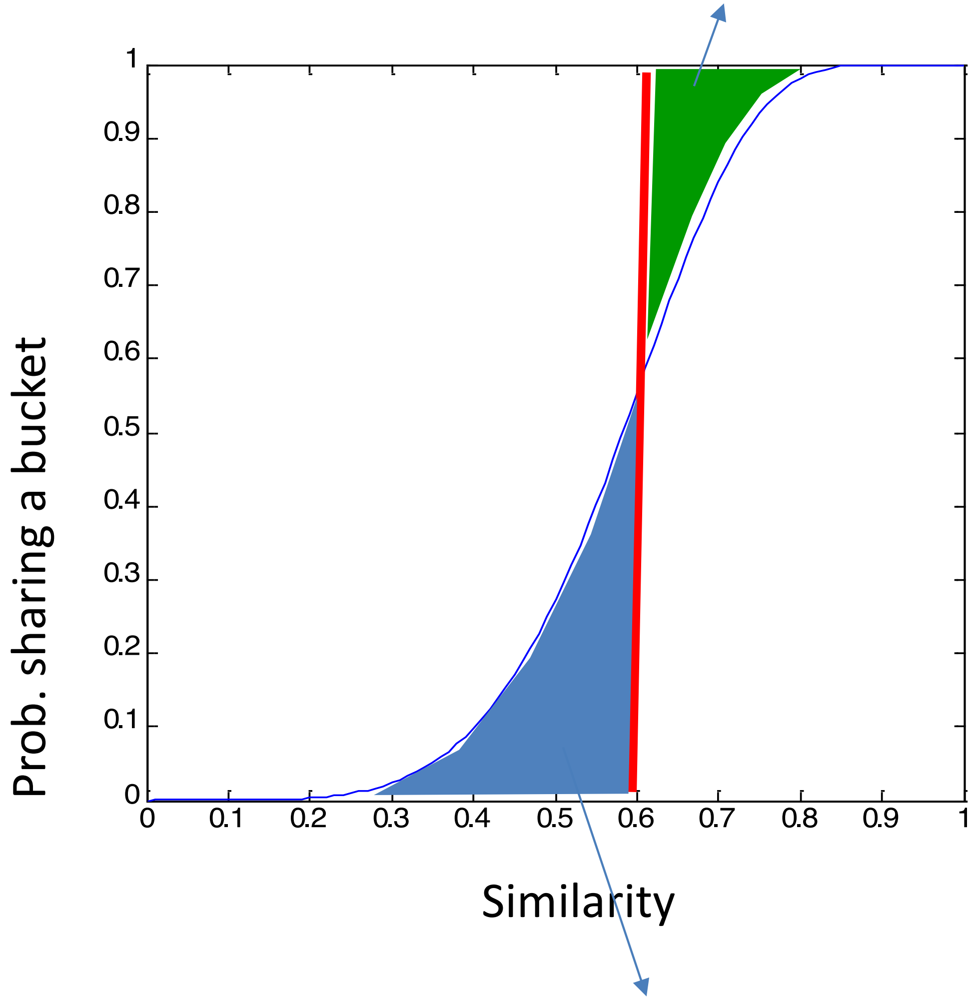
False negatives = area above the curve & to the right of line



False positives = area below the curve & to the left of line

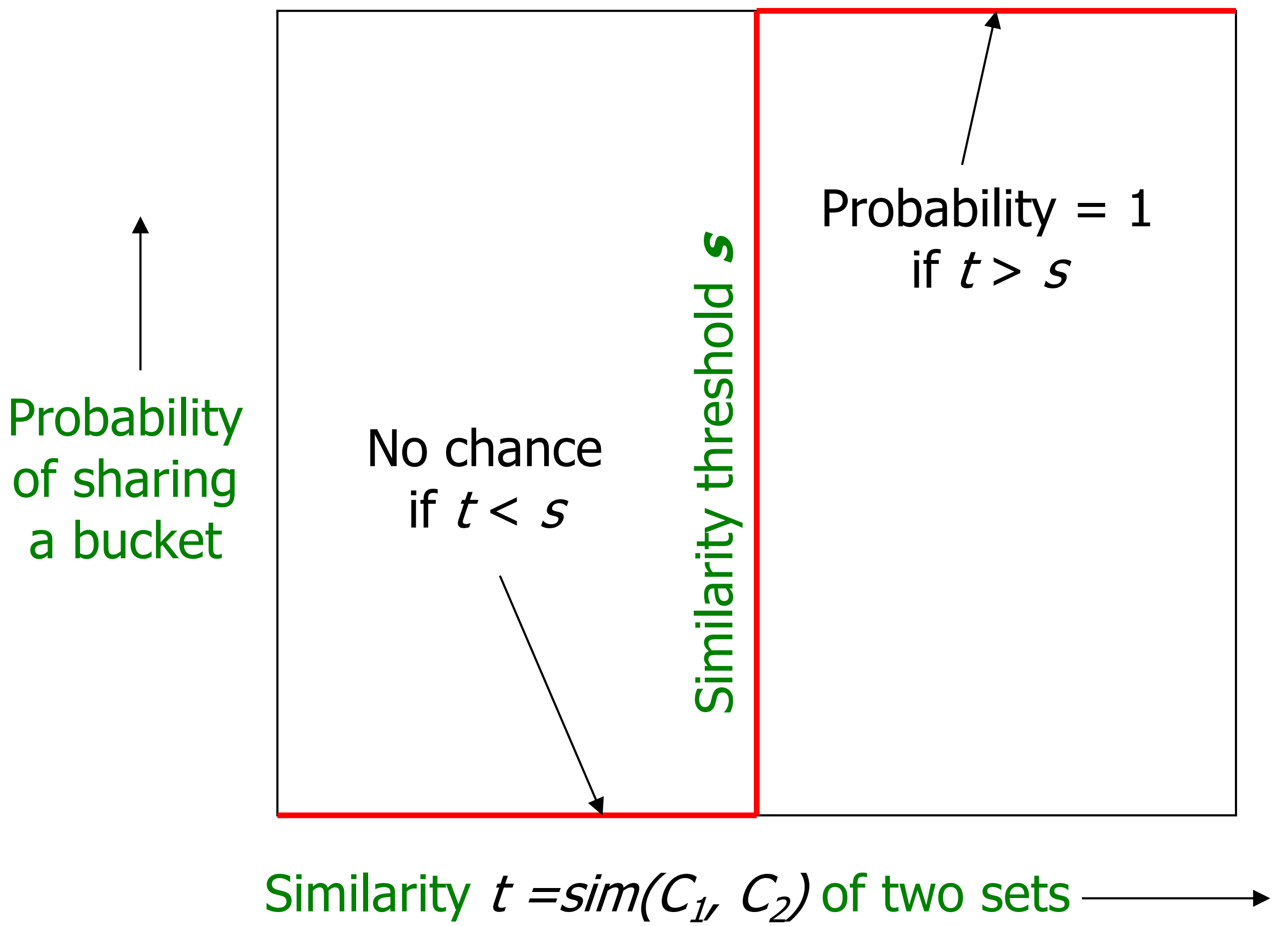
Error Zone

False negatives = area above the curve & to the right of line



False positives = area below the curve & to the left of line

Ideal s-p Curve



Reasons for Errors

- If two sets are identical
  - They will have the same signatures
  - Surely will be hashed to the same bucket
  - No errors

Reason for Errors

- If two sets are not identical, but similar
  - Still possible that signatures never totally agree in any band
  - End up in different buckets

  => False negatives

- Dissimilar signatures may happen to agree in some band
  - End up in same bucket

  => False positives

Example: False Negative (1)

- Similarity threshold t = .8
  - So if Jaccard(S_i, S_j) >= t, they are similar
- b = 20, r = 5
- Suppose Jaccard(S_1, S_2) = s = .8, so they are similar
- What is the probability that S_1 and S_2 are not identified as a candidate pair?
  - i.e., false negative

Example: False Negative (2)

- Probability of S_1 and S_2 identified as a candidate pair in a single band
  - $s^r = .8^5 = .328$
- So prob. that S_1 and S_2 are not candidate pair in any band
  - $(1-s^r)^b = (1 - .328)^{20} = .00035$

=> .035% of chances where two sets with similarity .8 are false negatives

Example: False Positive (1)

- Similarity threshold t = .8
  - So if Jaccard(S_i, S_j) >= t, they are similar
- b = 20, r = 5
- Suppose Jaccard(S_1, S_2) = s = .3, so they are not similar
- What is the probability that S_1 and S_2 are identified as a candidate pair?
  - i.e., false positive

Example: False Positive (2)

- Probability of S_1 and S_2 identified as a candidate pair in a band
  - $s^r = .3^5 = .00243$
- So prob. of S_1 and S_2 being candidate pair in at least one band
  - $1 - (1-s^r)^b = 1 - (1 - .00243)^{20} = .0474$
- So

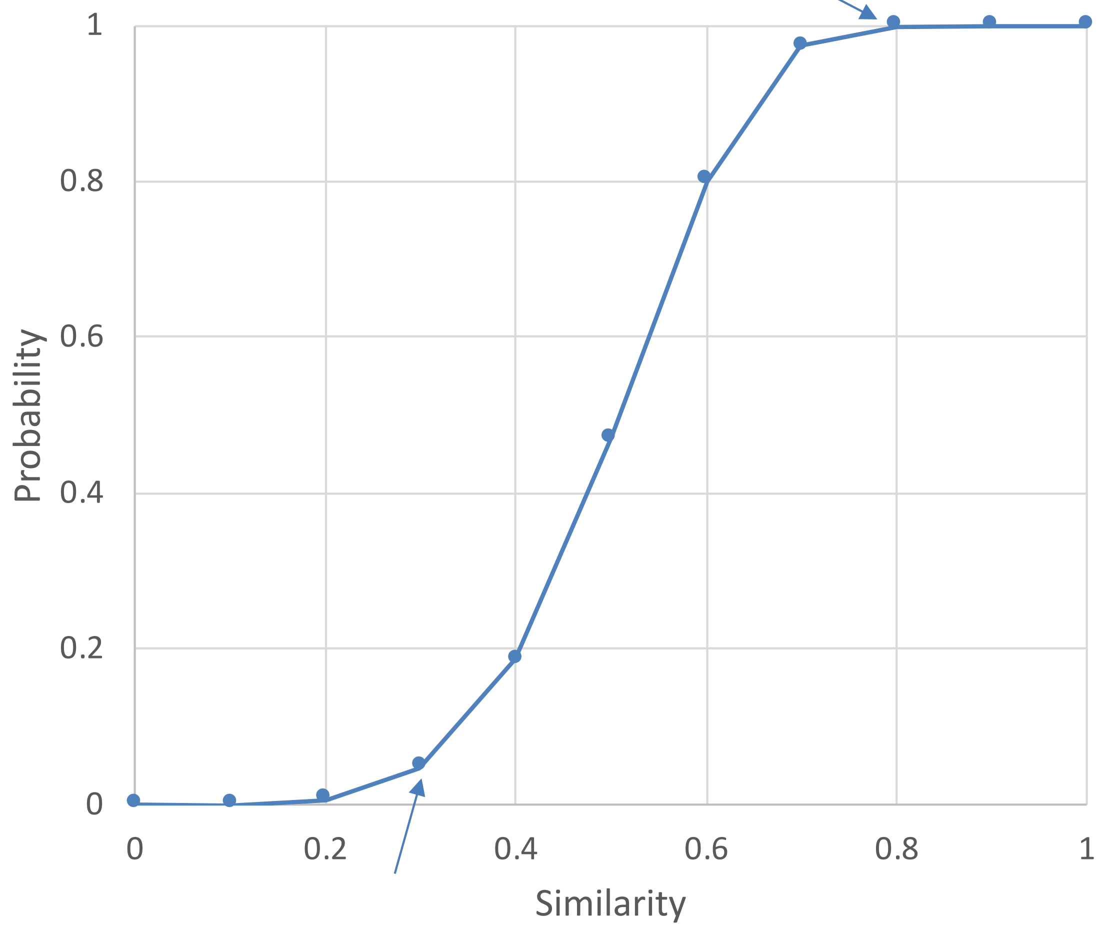
=> 4.74% of chances where two sets with similarity .3 end up in the same bucket

=> false positives

Example of Error Rates

s	p
0	0
0.1	0.0002
0.2	0.006381
0.3	0.047494
0.4	0.18605
0.5	0.470051
0.6	0.801902
0.7	0.974781
0.8	0.999644
0.9	1
1	1

.00035 = Prob. of being false negatives



.047 = prob. of being false positives

Differences btw PCY and LSH

- PCY
  - No false negatives (If an item pair is not in a frequent bucket, we know that it will not be frequent)
  - May have false positives, but the second pass will rule them out
- LSH
  - Can have both false positives and false negatives

Questions

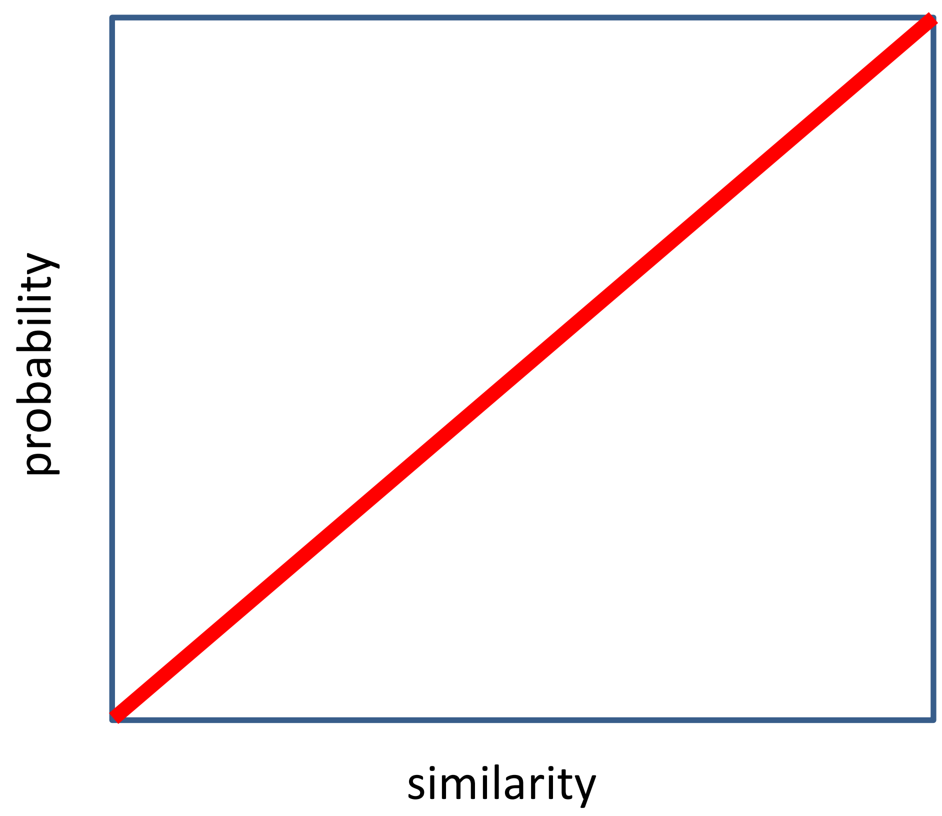
- Relationship btw curve shapes and errors
  - Spot false positives and negatives
- How does the curve change its shape? when
  - r = b = 1
  - r varies, b = 1
  - r = 1, b varies
  - both b and r > 1
- How to determine right similarity threshold? When
  - b * r = (fixed) n
  - n = length of signature

$r = b = 1$

- Prob. of becoming candidate: $p = 1 - (1 - s^r)^b$
  - Candidate = hashed to the same bucket
- if $r = 1$ & $b = 1$, then $p = s$
  - This is what minhash theorem told us
  - Prob. of two signature values (i.e., minhash values) hashed to the same bucket is the Jaccard similarity of their corresponding sets

$r = b = 1$

- Prob. of becoming candidate: $p = 1 - (1 – s^r)^b$
  - if $r = 1$ & $b = 1$, then $p = s$



$r = 5, b = 1$

- Prob. of becoming candidate: $p = 1 - (1 - s^r)^b$
  - Candidate = hashed to the same bucket

- When $r = 5$, $b = 1$, i.e., only one band
  - $p = s^r$
  - Note that p is now much lower than s
  - E.g., s = .8, p = .3

  => Need very high similarity to be
  hashed to the same bucket
  e.g., p > .5 only when s > .8

  => Reduce false positives

s	p
0.1	0.00001
0.2	0.00032
0.3	0.00243
0.4	0.01024
0.5	0.03125
0.6	0.07776
0.7	0.16807
0.8	0.32768
0.9	0.59049

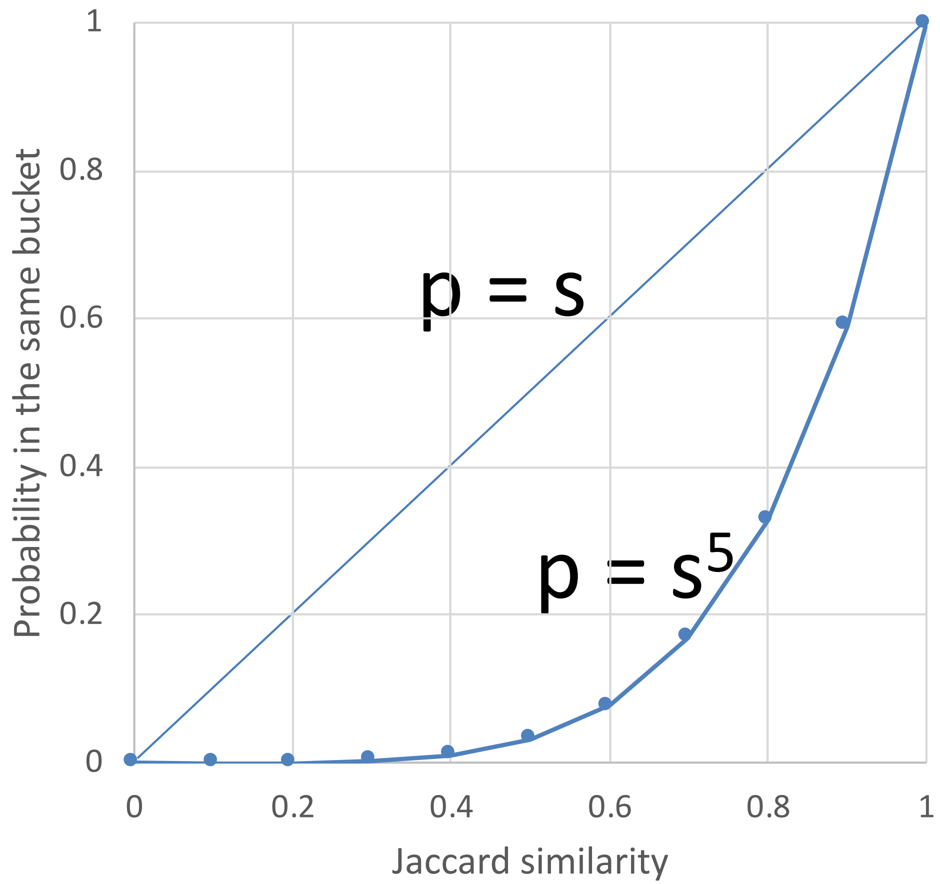
Amplification Effect

- When r = 5, b = 1, i.e., only one band
  - $p = s^r = s^5$
- p is much smaller than s
  - Especially true on small s values
  - E.g., p is 5 orders of magnitude smaller than s, when s = .1
  - i.e., r makes dissimilar pairs even more dissimilar

s	p
0.1	0.00001
0.2	0.00032
0.3	0.00243
0.4	0.01024
0.5	0.03125
0.6	0.07776
0.7	0.16807
0.8	0.32768
0.9	0.59049

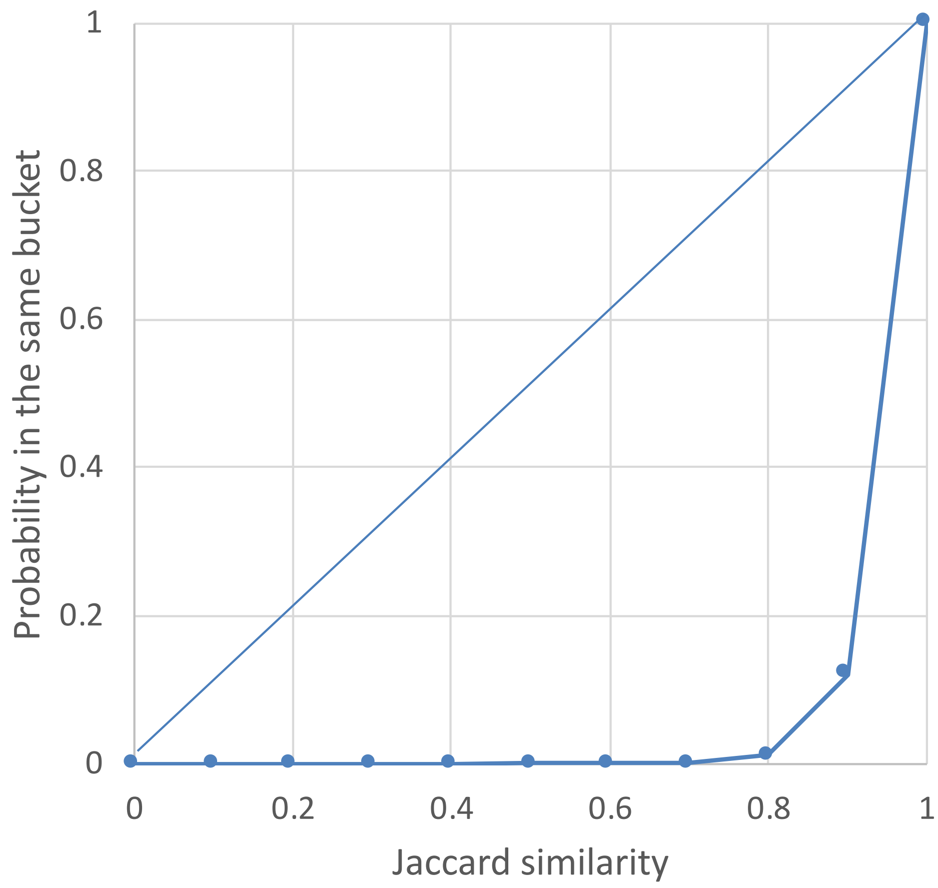
$r = 5, b = 1$

s	p
0	0
0.1	0.00001
0.2	0.00032
0.3	0.00243
0.4	0.01024
0.5	0.03125
0.6	0.07776
0.7	0.16807
0.8	0.32768
0.9	0.59049
1	1



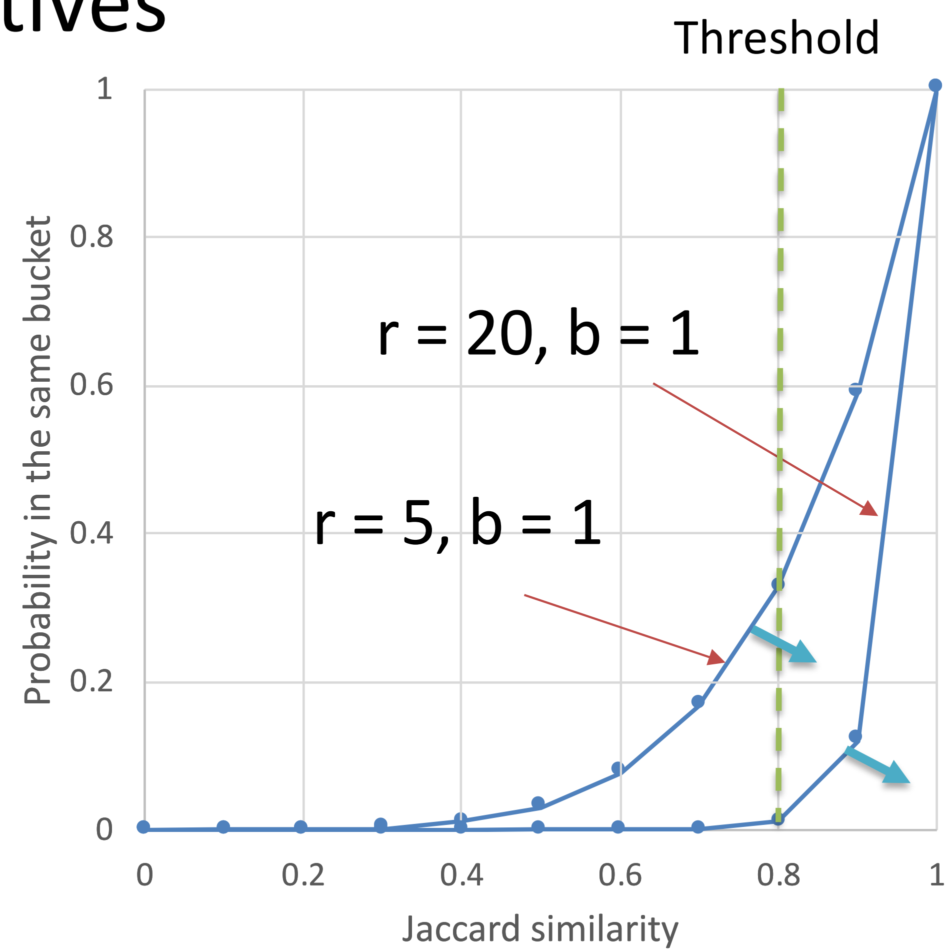
r = 20, b = 1

s	p
0	0
0.1	1E-20
0.2	1.05E-14
0.3	3.49E-11
0.4	1.1E-08
0.5	9.54E-07
0.6	3.66E-05
0.7	0.000798
0.8	0.011529
0.9	0.121577
1	1



Summary: b = 1, r increases

- Reduces false positives
- Increases false negatives



Questions

- Relationship btw curve shapes and errors
  - Spot false positives and negatives
- How does the curve change its shape? when
  - r = b = 1
  - r varies, b = 1
  - r = 1, b varies
  - both b and r > 1
- How to determine right similarity threshold? When
  - b * r = (fixed) n
  - n = length of signature

$r = 1, b = 5$

- Prob. of becoming candidate: $p = 1 - (1 - s^r)^b$
  - i.e., each band has only one row

  $\Rightarrow p = 1 - (1 - s)^b$

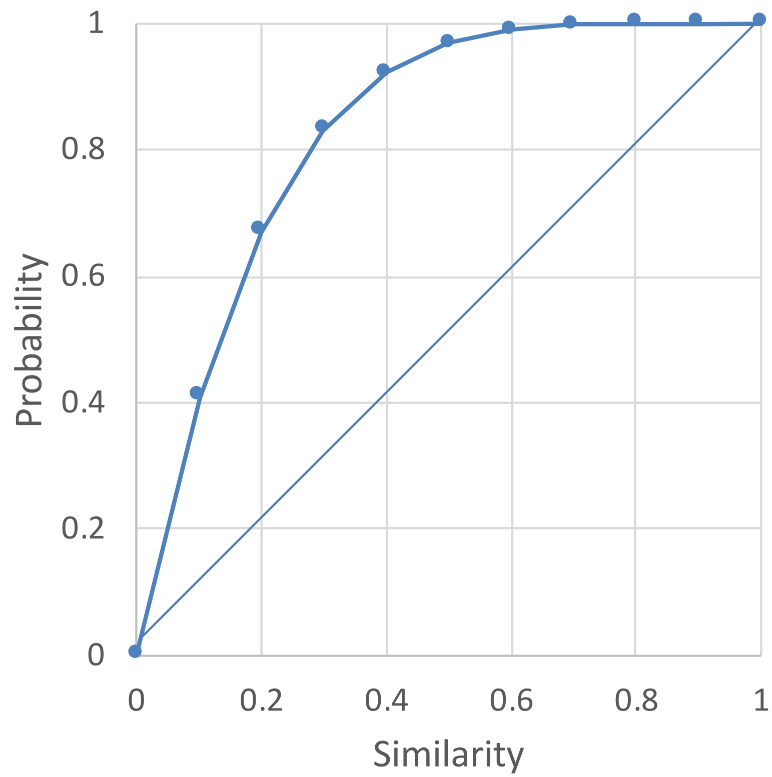
- b increases the probability of similar pairs placed in the same bucket

=> Reduce false negatives

$r = 1, b = 5$

s	p
0	0
0.1	0.40951
0.2	0.67232
0.3	0.83193
0.4	0.92224
0.5	0.96875
0.6	0.98976
0.7	0.99757
0.8	0.99968
0.9	0.99999
1	1

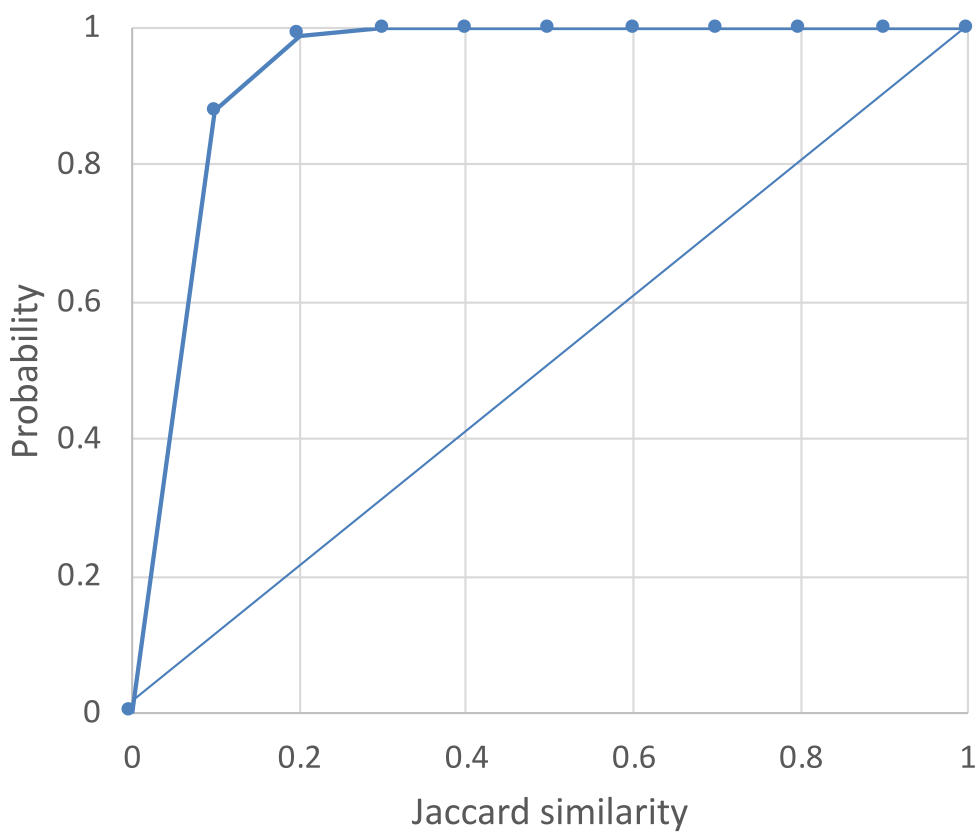
$$p = 1 - (1 - s)^5$$



r = 1, b = 20

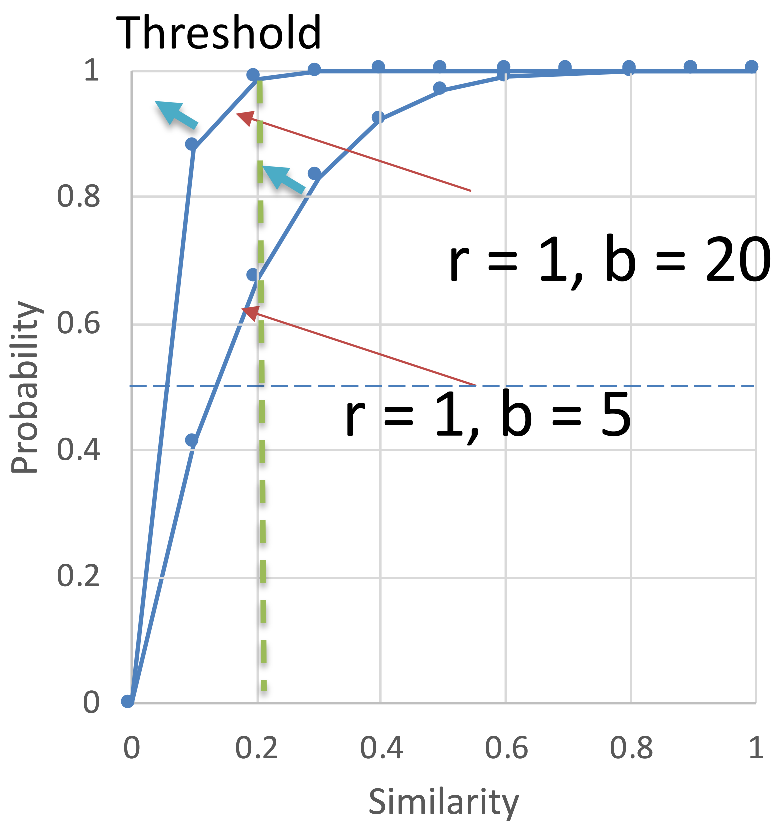
s	p
0	0
0.1	0.878423
0.2	0.988471
0.3	0.999202
0.4	0.999963
0.5	0.999999
0.6	1
0.7	1
0.8	1
0.9	1
1	1

$$p = 1 - (1 - s)^{20}$$



Summary: r = 1, b increases

- Reduces false negatives
- Increases false positives



Questions

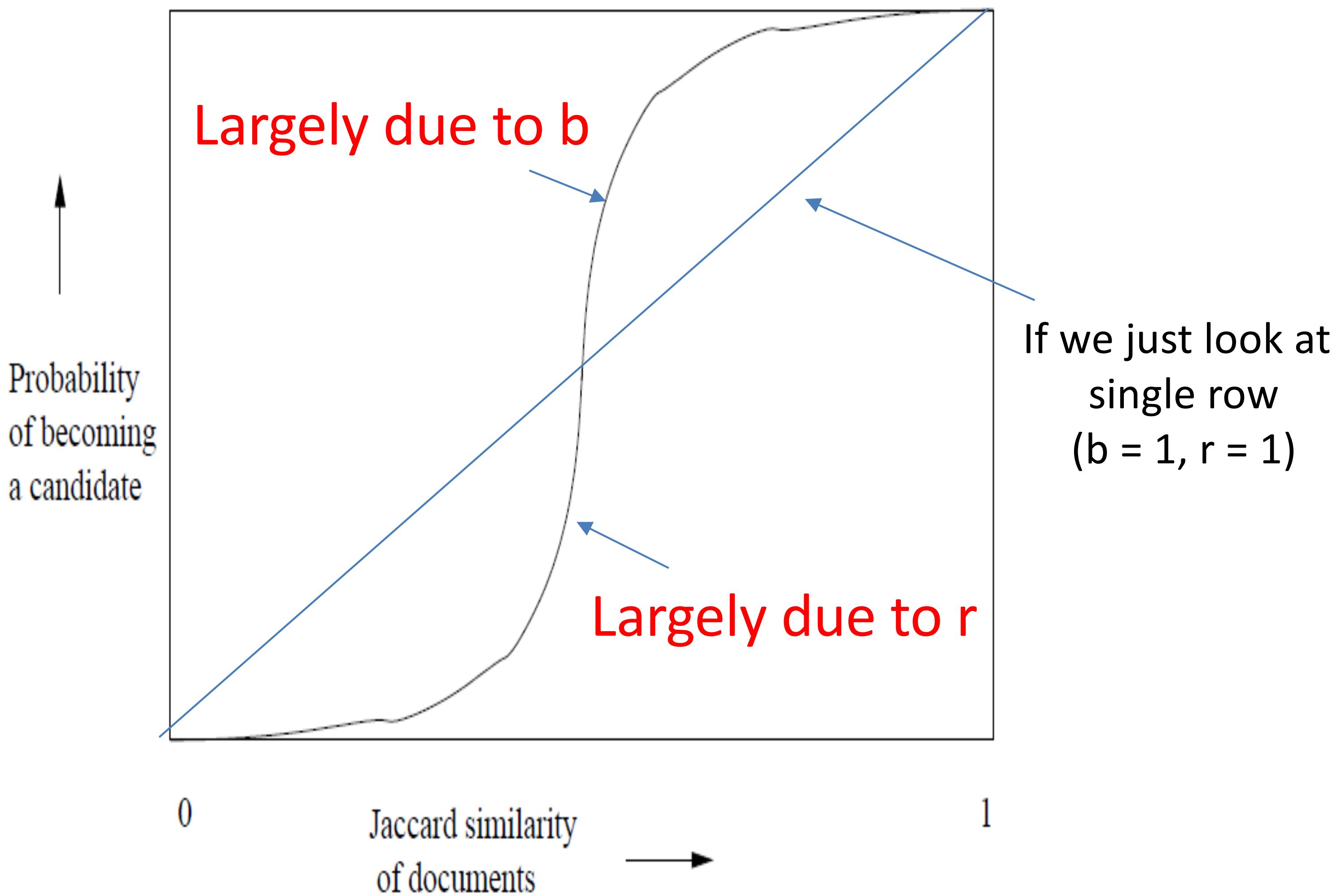
- Relationship btw curve shapes and errors
  - Spot false positives and negatives
- How does the curve change its shape? when
  - r = b = 1
  - r varies, b = 1
  - r = 1, b varies
  - both b and r > 1
- How to determine right similarity threshold? When
  - b * r = (fixed) n
  - n = length of signature

r = 5, b = 20

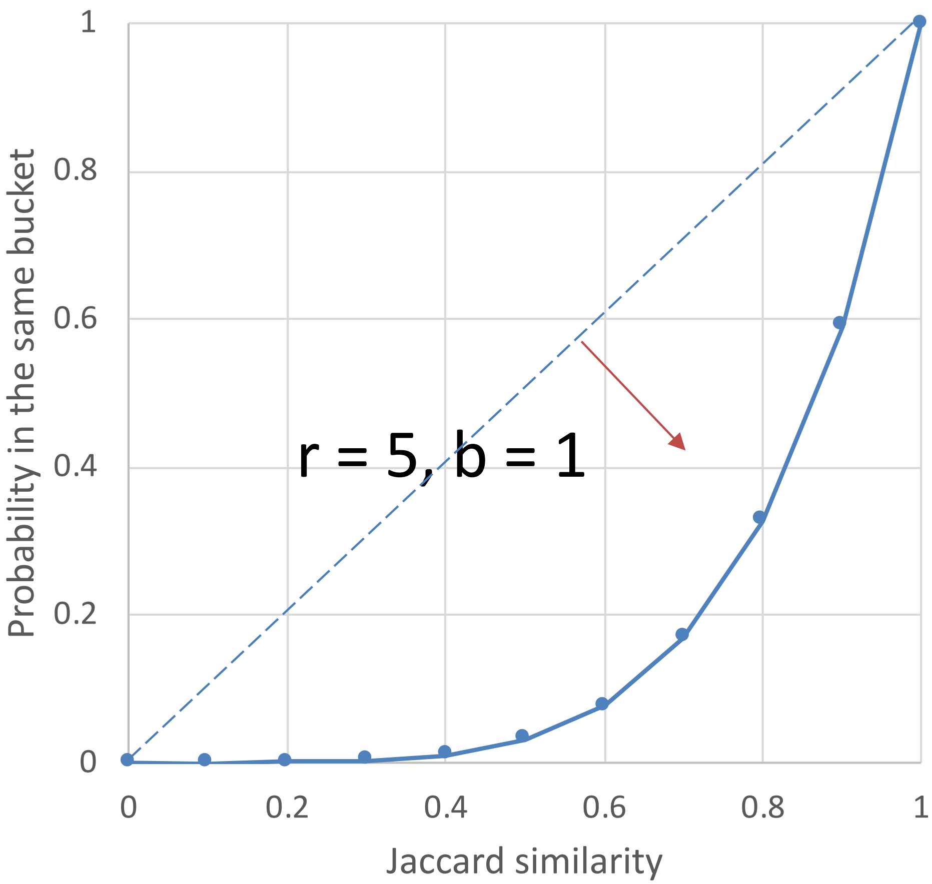
- Dissimilar pairs become even more dissimilar
  - E.g., s = .2 => p = .006
  - We know this is largely due to r
- Similar pairs become even more similar
  - e.g., s = .6 => p = .8
  - We know this is largely due to b

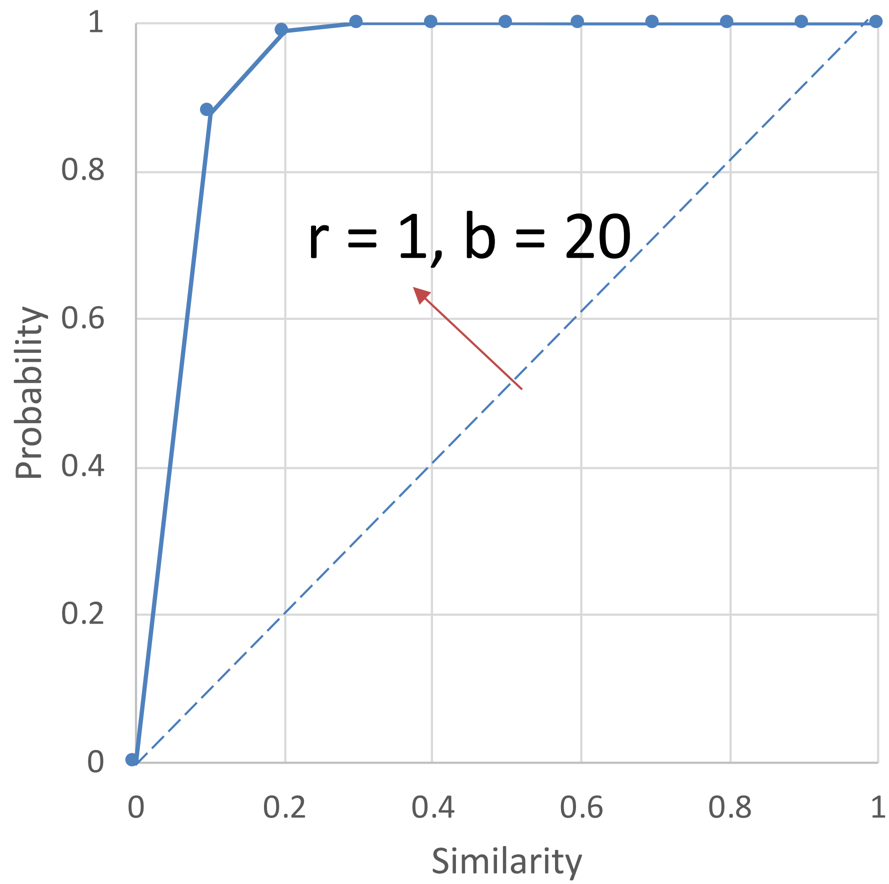
s	$1-(1-s^r)^b$	
.2	.006	Dissimilar
.3	.047	
.4	.186	
.5	.470	r = 5, b = 20
.6	.802	Similar
.7	.975	
.8	.9996	

$P = 1 - (1 - s^r)^b$ r= 5, b= 20



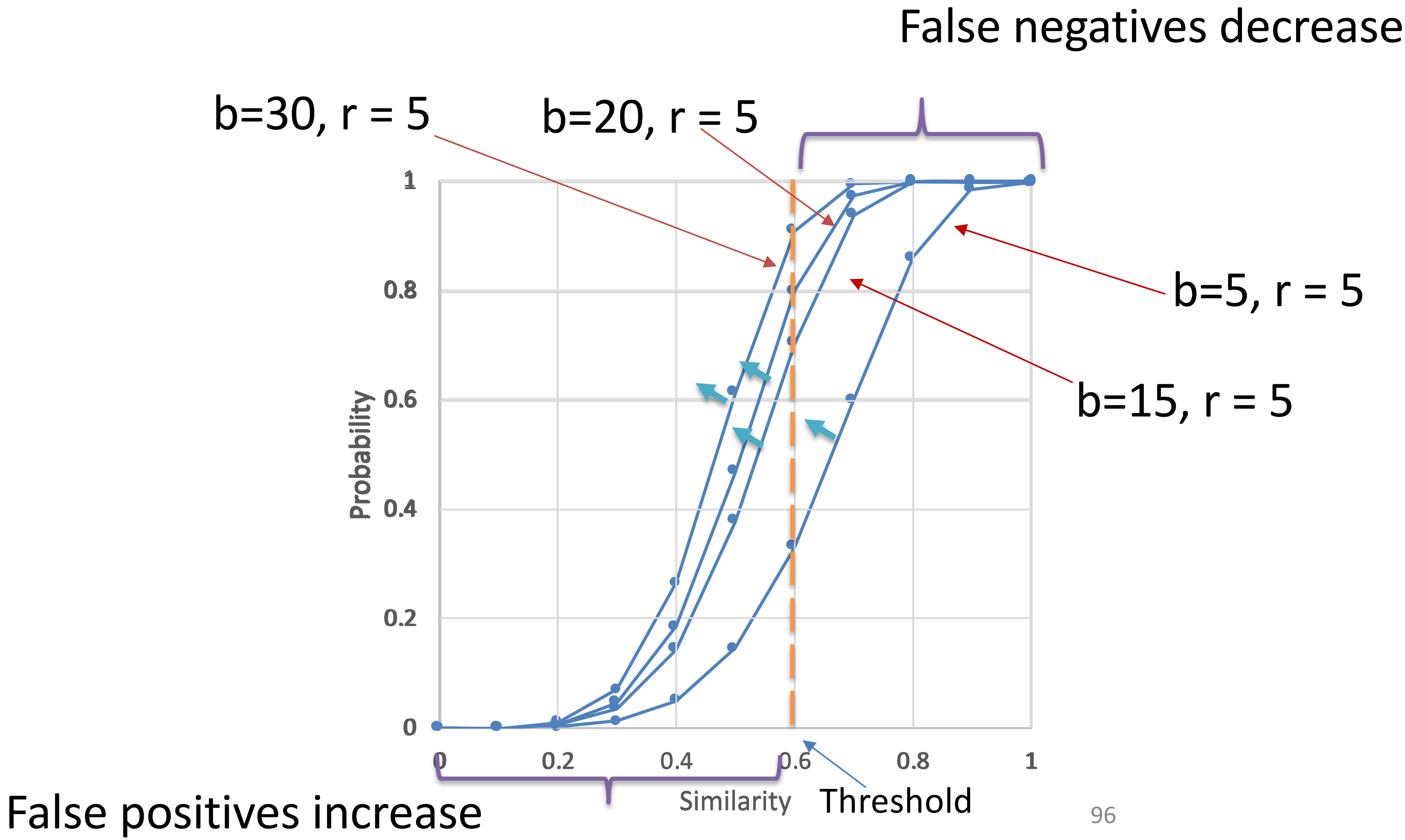
Decomposition Graph



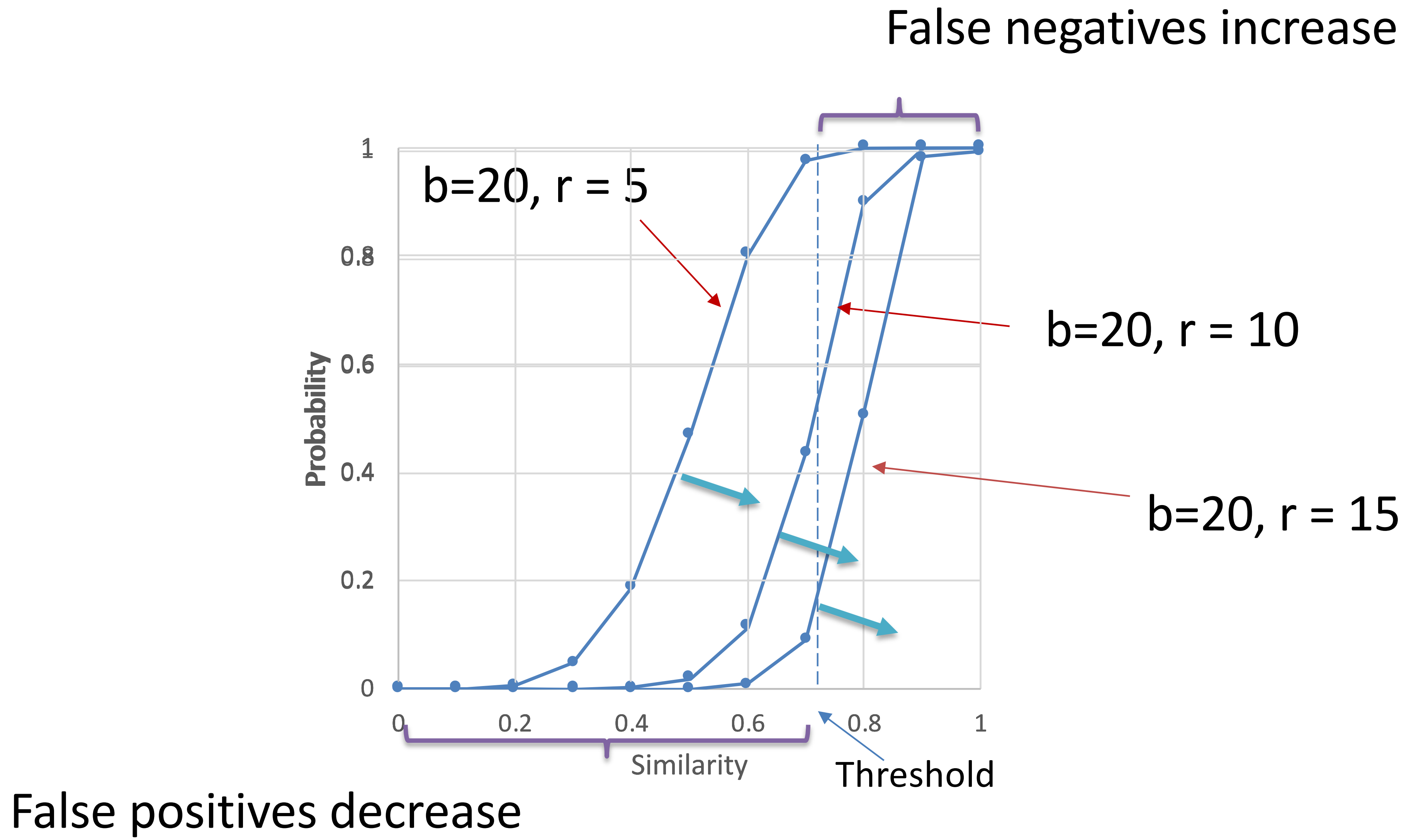


r brings down the curve, b raises it up

Increasing b



Increasing r



Questions

- Relationship btw curve shapes and errors
  - Spot false positives and negatives
- How does the curve change its shape? when
  - r = b = 1
  - r varies, b = 1
  - r = 1, b varies
  - both b and r > 1
- How to determine right similarity threshold? When
  - b * r = (fixed) n
  - n = length of signature

Determine Threshold

- Manually set, e.g., t = .8
- This is the threshold that defines how similar two documents have to be in order for them to be regarded as a desired “similar pair”.
- Or set it to the value where p = .5
  - I.e., when curve rises half way
  - This is a predicted threshold

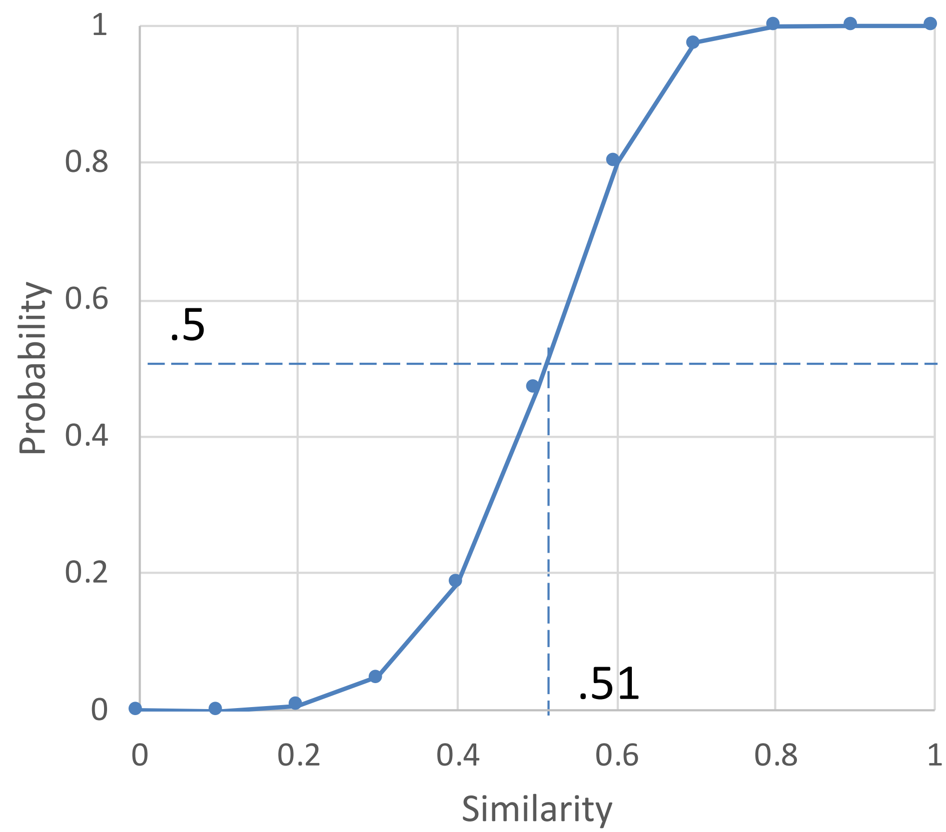
Determine Predicted Threshold

$p=1-(1-s^r)^b \Rightarrow s = (1-(1-p)^{1/b})^{1/r}$

$r = 5, b = 20, p = .5 \Rightarrow s = .51$

s	p
0	0
0.1	0.0002
0.2	0.006381
0.3	0.047494
0.4	0.18605
0.5	0.470051
0.6	0.801902
0.7	0.974781
0.8	0.999644
0.9	1
1	1

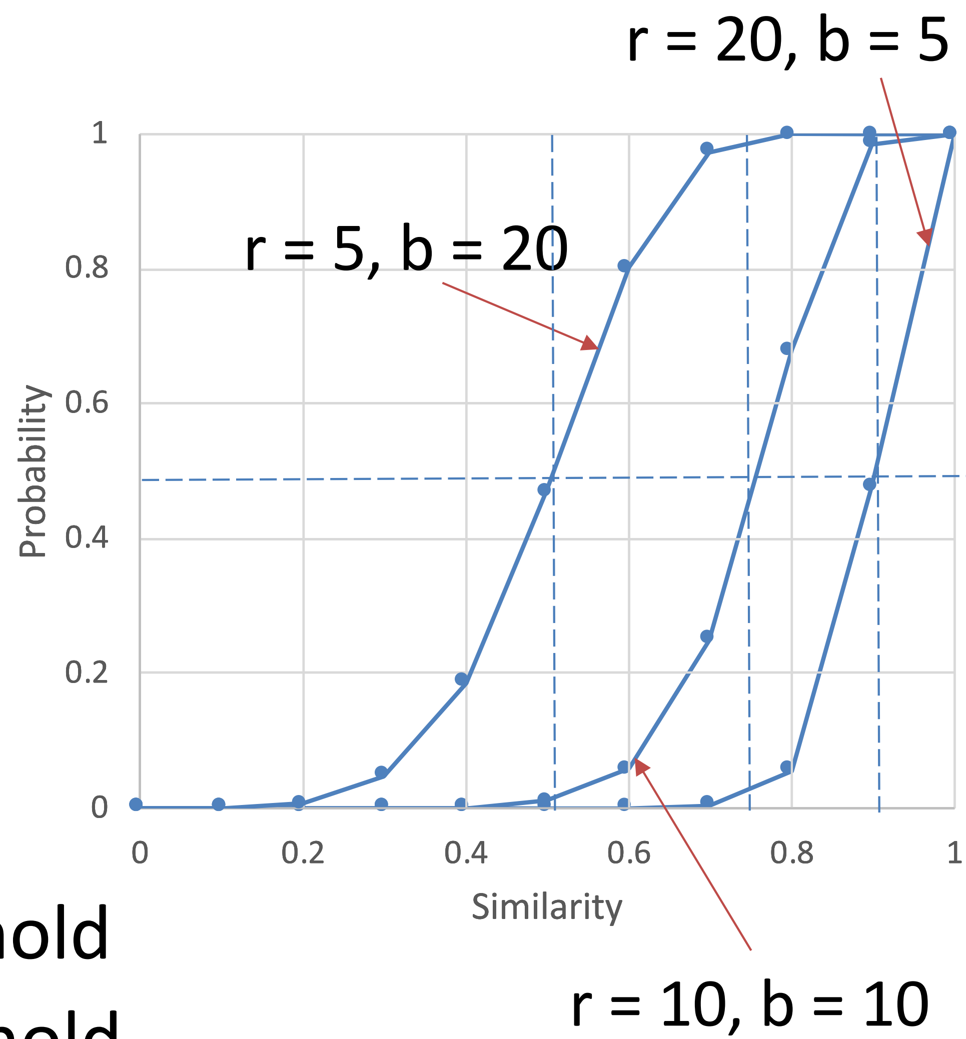
b=20, r = 5



Adjust b and r

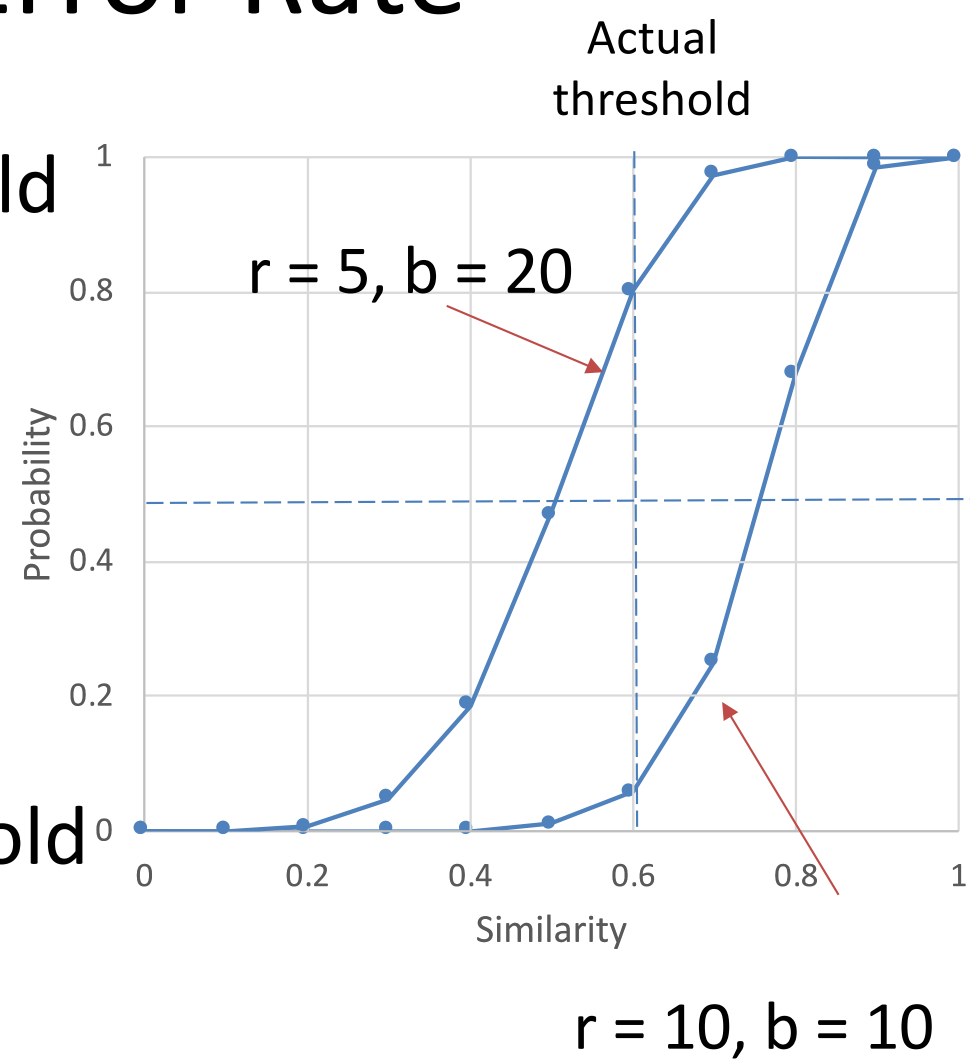
- n = 100, n = b*r
  - r = 5, b = 20
  - r = 10, b = 10
  - r = 20, b = 5

- Typically for predicted:
  - Larger r => larger threshold
  - Larger b => lower threshold



Effect on Error Rate

- Increase predicted threshold
  - Reduce false positives
  - Less checking time later



- Decrease predicted threshold
  - Reduce false negatives
  - But longer checking later

Finding Similar Documents

1. Construct k-shingles, turn them into integers
2. Build minhash signatures of length n
3. Choose b and r, s.t., br = n, to adjust (predicted) threshold
   – Larger threshold => reduce false positives (precision up)
   – Smaller threshold => reduce false negatives (recall up)
4. Construct candidate pairs
5. Examine signatures of candidates to see if the fraction of common rows > t (actual threshold)
6. May check documents if their signatures are similar

Example

```
SELECT DocName
FROM Documents
WHERE (
BandOneMinHashAlgorithmOne = 1 AND
BandOneMinHashAlgorithmTwo = 3 AND
BandOneMinHashAlgorithmThree = 6 AND
BandOneMinHashAlgorithmFour = 0
) OR
(
BandTwoMinHashAlgorithmOne = # AND
BandTwoMinHashAlgorithmTwo = # AND
BandTwoMinHashAlgorithmThree = # AND
BandTwoMinHashAlgorithmFour = #
) OR (
BandThreeMinHashAlgorithmOne = # AND
BandThreeMinHashAlgorithmTwo = # AND
BandThreeMinHashAlgorithmThree = # AND
BandThreeMinHashAlgorithmFour = #
) OR (
BandFourMinHashAlgorithmOne = # AND
BandFourMinHashAlgorithmTwo = # AND
BandFourMinHashAlgorithmThree = # AND
BandFourMinHashAlgorithmFour = #
) OR (
BandFiveMinHashAlgorithmOne = # AND
BandFiveMinHashAlgorithmTwo = # AND
BandFiveMinHashAlgorithmThree = # AND
BandFiveMinHashAlgorithmFour = #
)
```

		MinHash Algorithm One	MinHash Algorithm Two	MinHash Algorithm Three	MinHash Algorithm Four
Band One	Document One	1	3	6	0
	Document Two	2	3	1	0
	Document Three	1	3	6	0
	Document Four	2	1	3	1

http://matthewcasperson.blogspot.com/2013/11/minhash-for-dummies.html