Chapter 3

Finding Similar Items

Finding “Similar” Items

- Many problems can be expressed as finding ***similar*** items
  - Find nearest neighbors in ***high dimensional*** space
  - One of the fundamental data mining problems

- Examples
  - Finding near-duplicate Web pages
    - Plagiarisms or mirrors
  - Finding pages with similar words
    - Duplicate detection, classification by topic
  - Finding customers who purchased similar products
    - Products with similar customers (recommender systems)
  - Finding images with similar features
    - Similarly, users who visited similar websites

Problem for This Chapter

- Given
  - High dimensional data points $x_1, x_2, \ldots$
    - (ex) Image: a long vector of pixel colors

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 2 & 1 \\ 0 & 1 & 0 \end{bmatrix} \rightarrow [1\ 2\ 1\ 0\ 2\ 1\ 0\ 1\ 0]$$

  - Some distance function $d(x_1, x_2)$
    - Which quantifies the "distance" between x_1 and x_2
- Goal
  - Find all pairs of data points (x_i, x_j) that are within some distance threshold $d(x_i, x_j) \leq s$
- Note
  - Naïve solution would take $O(N^2)$ where N is the number of data points
  - How can this be done in $O(N)$??

Finding “Similar” Documents

- Here, we focus on finding ***similar documents***
- Goal
  - Given a ***large*** (e.g., 10^9) number of documents, find “near duplicate” pairs
- Applications
  - Mirror websites (don’t want to show both in search results)
  - Similar news articles (cluster articles by “same story”)
- Difficulties
  - Many small pieces of one document can appear out of order in another
  - There are too many documents to compare all pairs
  - Documents are so large or so many that they cannot fit in main memory

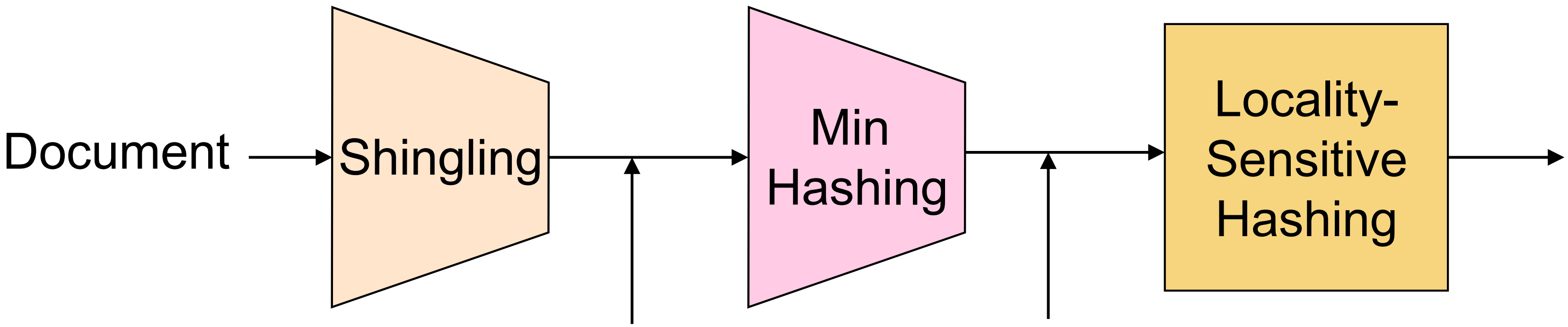
3 Steps for Finding Similar Documents

1. Shingling
   – Convert documents to ***sets***

2. Minhashing
   – Convert large sets to short ***signatures***, while preserving similarity

3. Locality-Sensitive Hashing
   – Focus on pairs of signatures likely to be from similar documents
   – The results are ***candidate pairs***

The Big Picture



The set of strings of length ***k*** that appear in the document

Signatures: short integer vectors that represent the sets, and reflect their similarity

Candidate pairs**:** those pairs of signatures that we need to test for similarity

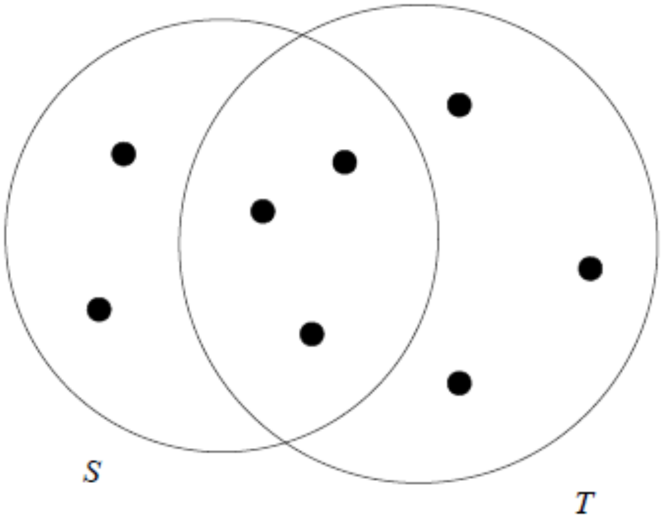
Jaccard Similarity

- The similarity of ***sets*** by looking at the size of their intersection

- The Jaccard similarity of sets S and T

$$SIM(S,T) = \frac{|S \cap T|}{|S \cup T|}$$

- Example
  - $SIM(S, T) = 3/8$



Similarity of Documents

- Here, we focus on ***character-level (i.e., textual)*** similarity
  - Note that “similar meaning” requires other techniques
- Testing whether two documents are exact duplicates is easy
  - However, in many applications, the documents are ***not*** identical
- Applications of textual similarity
  - Plagiarism
  - Mirror pages
    - Search engines should avoid showing two pages that nearly identical
  - Articles from the same source
    - News aggregators(e.g., Google News) should show only one for each article

Collaborative Filtering

- A process whereby we recommend to users items that were liked by other users who have exhibited similar tastes
  - Another class of applications where ***similarity of sets*** is very important

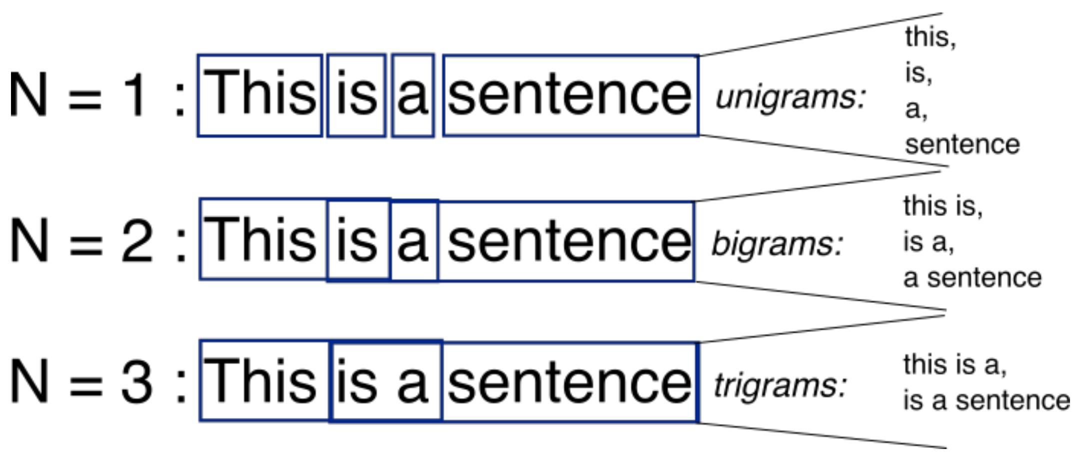
- Examples
  - On-line purchases (e.g., Amazon.com)
    - Two customers are similar if their sets of purchased items have a high Jaccard similarity
  - Movie ratings (e.g., NetFlix)
    - Moves are similar if they were rented or rated highly by many of the same customers
    - Customers are similar if they rented or rated highly many of the same movies

Documents as Sets

- Simple approaches
  - Document = set of words appearing in document
  - Document = set of “important” words
  - Don’t work well for this application. ***Why?***

- Need to account for ***ordering*** of words!

- A different way: Shingles (or grams)!



Shingling of Documents

- The most effective way to represent documents as ***sets***
  – For the purpose of identifying lexically similar documents

- Construct from the documents ***the set of short strings*** that appear within it
  – (ex) Document = `abcab` → the set of 3-shingles = {`abc`, `bca`, `cab`}

- Documents that share sentences (or phrases) will have many ***common*** elements in their shingling sets
  – Even if they appear in different orders in the two documents

Definition: k-Shingles (or n-Grams)

- A k-shingle for a document
  - Any substring of length k found within the document
- Then, we associate with each document the set of k-shingles that appear one or more times within that document
- Example
  - Document $D = \texttt{abcdabd}$
  - The set of 2-shingles for $D = \{\texttt{ab}, \texttt{bc}, \texttt{cd}, \texttt{cd}, \texttt{da}, \texttt{bd}\}$
    - A variation of shingling produces a bag, rather than a set

Treating White Space

- White space
  - Blank, tab, newline, etc.
- Options
  - Replace any sequence of white-space characters by a single blank
  - We may ***eliminate*** whitespace altogether
- Example
  - $D_1 =$ "`The plane was ready for touch down`"
  - $D_2 =$ "`The quarterback scored a touchdown`"
  - If we use $k = 9$ and retain the blanks, D_1 has shingles `touch dow` and `ouch down`, while D_2 has `touchdown`
  - If we eliminated the blanks, both would have `touchdown`

Choosing the Shingle Size

- If we pick k too small
  - Even the documents that have no the same sentences or even phrases would have ***high*** Jaccard similarity (e.g., $k = 1$)
- Rule of thumb
  - k should be pick ***large*** enough that the probability of any given shingle appearing in any given document is low
- Example
  - $k = 5$ is ok for short documents (e.g., emails)
  - $k = 10$ is better for long documents (e.g., research articles)

Compressing Shingles

- We can ***compress*** long shingles by using a ***hash function*** h
  - h maps each k-shingle to $0, \ldots, B-1$, where B is the number of buckets
  - Then, a document is represented as the set of ***hash values*** of its k-shingles

- Not only the data been compacted, but we can now manipulate (hashed) shingles by single-word machine operations

- Example
  - $k = 2$, document $D = \texttt{abcab}$
  - The set of 2-shingles $= \{\texttt{ab}, \texttt{bc}, \texttt{ca}\}$ ($9 \times 3 = 27$ bytes)
  - The set of their hash values $= \{1, 5\ , 7\}$ ($4 \times 3 = 12$ bytes)

Shingles Built from Words

- In many applications, we want to ignore stop words
  - (ex) “a,” “and,” “for,” etc.
- However, for the problem of finding similar news articles, defining a shingle to be ***a stop word followed by the next two words*** forms a useful set of shingles
  - Bias the set of shingles in favor of the article, rather than its surrounding material
- Example
  - An ad: “Buy Sudzo”
  - A news article: “***A*** spokesperson ***for the*** Sudzo Corporation revealed today ***that*** studies ***have*** shown ***it is*** good ***for*** people ***to*** buy Sudzo Product”
  - The set of shingles = {“A spokesperson for”, “for the Sudzo”, ….}
    - Note that ***none*** are from the ad

Motivation for Minhash/LSH

- Suppose we need to find near-duplicate documents among $N =$ 1 million documents

- Naïvely, we would have to compute ***pairwise*** Jaccard similarities for every pair of docs
  - $N(N-1)/2 \approx 5 \times 10^{11}$ comparisons
  - At 10^5 secs/day and 10^6 comparisons/sec, it would take 5 days

- For $N = 10$ million, it takes more than a year…

Similarity-Preserving Summaries of Sets

- Even if we hash each shingle to 4 bytes, the space needed to store all sets of shingles is still large
  - We may have millions of documents
- Our goal
  - Replace large sets by much smaller representations called ***signatures***
    - (ex) 200,000 byte hashed-shingle sets → 1,000 byte signatures
- Important property required for signatures
  - We must be able to estimate the Jaccard similarity of two sets from their signatures ***alone***
    - Note that it is not possible that the signatures give the exact similarity

Matrix Representation of Sets

- Characteristic matrix
  - Columns: sets (documents)
  - Rows: elements (shingles)
  - 1 in row e and column s, if and only if e is a member of s

- Example
  - $S_1 = \{a, d\}$, $S_2 = \{c\}$, $S_3 = \{b, d, e\}$, $S_4 = \{a, c, d\}$

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

Minhashing (1/2)

- **Goal:** Find a hash function h such that
  - If $SIM(S_1, S_2)$ is ***high***, then $h(S_1) = h(S_2)$ with a ***high*** probability
  - If $SIM(S_1, S_2)$ is ***low***, then $h(S_1) \neq h(S_2)$ with a ***high*** probability
- Clearly, the hash function depends on the similarity metric
  - Not all similarity metrics have a suitable hash function
- There is a suitable hash function for the Jaccard similarity
  - It is called ***minhashing***

Minhashing (2/2)

- Pick a random permutation of the rows of the characteristic matrix
  - (ex) $abcde \rightarrow beadc$
- Minhash function $h(S)$ for a set S
  - The index of the first row, ***in the permuted order***, in which the column has 1
- Example
  - Permuted order = $beadc$
  - $h(S_1) = a$
  - $h(S_2) = c$
  - $h(S_3) = b$
  - $h(S_4) = a$

$Element$	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

Minhashing and Jaccard Similarity

- Let S_1 and S_2 be two sets
- Let $SIM(S_1, S_2)$ be the Jaccard similarity of S_1 and S_2
- Let h be the minhash function for a random permutation of rows
- ***Then, the probability that*** $\boldsymbol{h(S_1) = h(S_2)}$ ***equals*** $\boldsymbol{SIM(S_1, S_2)}$

Proof (1/2)

- Consider the columns for sets S_1 and S_2

Element	S_1	S_2	
b	1	0	Type Y
e	1	0	
a	0	1	
d	1	1	Type X
c	0	1	

  – Type X rows have 1 in both columns

  – Type Y rows have 1 in one of the columns and 0 in the other

  – Type Z rows have 0 in both columns

- Let x and y be the number of rows of type X and Y, respectively

- Then, $SIM(S_1, S_2) = x/(x + y)$

Proof (2/2)

- Now consider the probability that $h(S_1) = h(S_2)$
- Suppose we proceed from the top the rows permuted randomly
- The probability that we shall meet a type X row before we meet a type Y row
  - $x/(x + y)$
  - This corresponds to the probability that $h(S_1) = h(S_2)$
- Therefore, the probability that $h(S_1) = h(S_2)$ is $x/(x + y)$, which is also $SIM(S_1, S_2)$

Minhash Signatures

- Suppose we represent sets by their characteristic matrix M
- We pick at n random permutations of the rows of M
  - $n = 100$ or several hundreds
- Let h_1, h_2, … h_n be the minhash functions for n permutations
- The minhash signature for S is the vector $[h_1(S), h_2(S), \ldots, h_n(S)]$
- Thus, we can form a signature matrix from M
  - The ith column of M is replaced by the minhash signature for the ith column

(Ex) Minhash Signatures

Characteristic matrix M

S_1	S_2	S_3	S_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

3 Permutations

This row becomes the 3rd row in the 3rd permutation

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

This row becomes the 4th row in the 1st permutation

Signature matrix

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

Computing Minhash Signatures

- Picking a random permutation and permuting rows is ***not*** feasible
  - Permuting millions or billions of rows is time-consuming

- Instead, we can simulate a random permutation by a ***randomly chosen*** hash function
  - A hash function h that maps $0, \ldots, k-1$ to bucket numbers $0, \ldots, k-1$
  - (ex) $h(x) = (ax + b) \; mod \; k,$ where a and b are random integers
  - h "permutes" row r to position $h(r)$ in the permuted order

- Thus, instead of picking n random permutations of rows, we pick n randomly chosen hash functions $h_1, h_2, \ldots, h_n$ on the rows
  - The signature matrix is then constructed by considering each row in their given order

Implementation

- SIG(i, c)
  - The element of the signature matrix for the ith permutation and column c

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

$SIG(2, 4)$

- Algorithm

```
Initially, set SIG(i, c) to ∞ for all i and c

For each row r do
  Compute h_1(r), h_2(r), …, h_n(r)
  For each column c do
    (a) If c has 0 in row r, do nothing
    (b) If c has 1 in row r, then for each i = 1, 2, …, n
        set SIG(i, c) to the smaller of the current value
        of SIG(i, c) and h_i(r)
```

(Ex) Signature Matrix (1/7)

- Consider the following characteristic matrix and two hash functions: $h_1(x) = x + 1 \bmod 5$ and $h_2(x) = 3x + 1 \bmod 5$

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

- Initially, we set $\mathrm{SIG}(i, c)$ to ∞ for all i and c

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

(Ex) Signature Matrix (2/7)

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

- First we consider row 0
  - $h_1(0) = 1$, $h_2(0) = 1$

	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

For S_1, row ***1*** (originally row 0) is the first row whose column is 1 ***for now***

(Ex) Signature Matrix (3/7)

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

- Next we consider row 1
  - $h_1(1) = 2$, $h_2(1) = 4$

	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

For S_3, row ***4*** (originally row 1) is the first row whose column is 1 ***for now***

(Ex) Signature Matrix (4/7)

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

- Next we consider row 2
  - $h_1(2) = 3$, $h_2(2) = 2$

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

$\Rightarrow$

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

(Ex) Signature Matrix (5/7)

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

- Next we consider row 3
  - $h_1(3) = 4$, $h_2(3) = 0$

	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

(Ex) Signature Matrix (6/7)

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

- Next we consider row 4
  - $h_1(3) = 0$, $h_2(3) = 3$

	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

(Ex) Signature Matrix (7/7)

- Estimating Jaccard similarities from the signature matrix

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

	$SIM(S_1, S_2)$	$SIM(S_1, S_3)$	$SIM(S_1, S_4)$
True	0	1/4	2/3
Estimated	0	1/2	1

- The estimates become close as the number of hash functions increases

Locality-Sensitive Hashing

- Minhashing
  - Compresses large documents into small signatures, while preserving the expected similarity of any pair of documents
- Still, the number of pairs of documents may be too large
  - (ex) $1{,}000{,}000$ documents → ${}_{1{,}000{,}000}C_2 \approx 500{,}000{,}000{,}000$ pairs
- **Locality-sensitive hashing (LSH)**
  - Allows us to focus our attention ***only*** on pairs that are likely to be similar, without investigating every pair

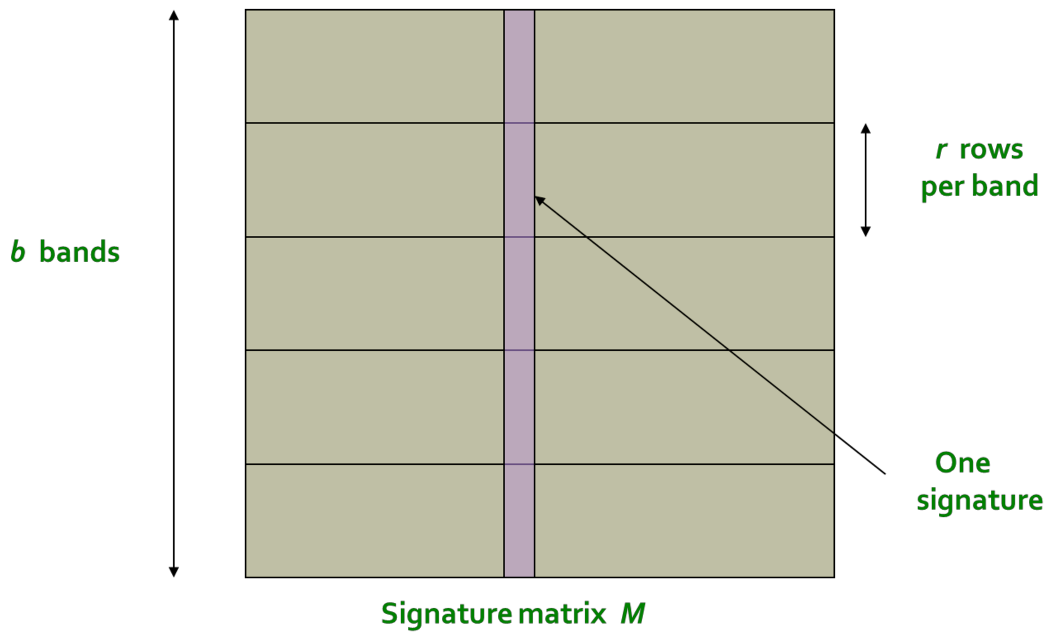
General Approach to LSH

- Big idea
  - Hash items ***several times***
  - We call a pair that hashed to the same bucket for any of the hashings to be a ***candidate pair***
  - We check ***only*** the candidate pair for similarity

- The hope is that most of the dissimilar pairs will never hash to the same bucket
  - ***False positives*** will be only a small fraction of all pairs
    - Those dissimilar pairs that do hash to the same bucket
  - ***False negatives*** will be only a small fraction of the truly similar pairs
    - Those similar pairs that do not hash to the same bucket under at least one of the hash functions

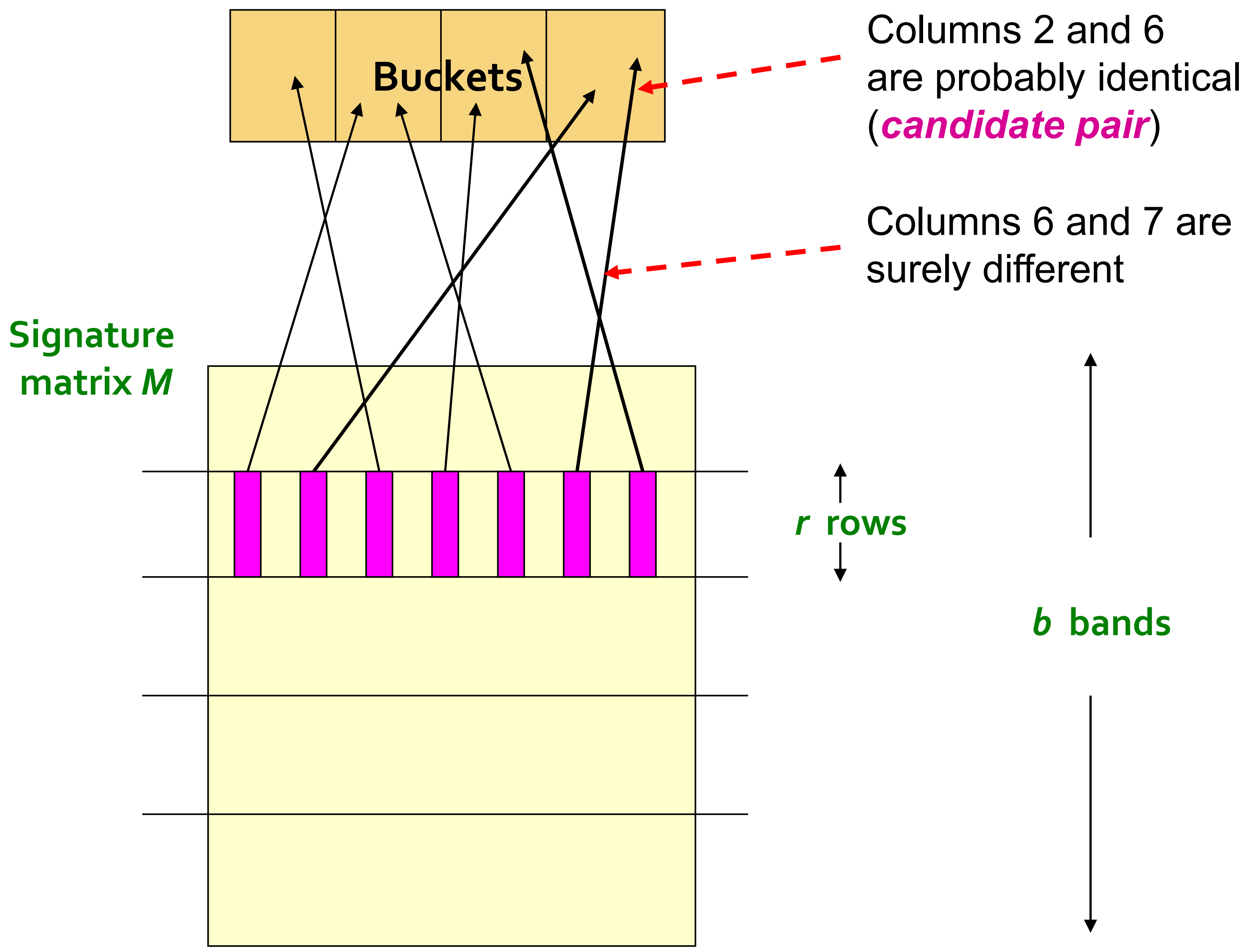
LSH for Minhash Signatures

- Divide the signature matrix into b bands of r rows



- For each band, hash its portion of each column to k buckets
  - Make k as large as possible (so there are almost no collisions)
- ***Candidate column pairs*** are those that hash to the same bucket for one or more bands

(Ex) LSH for Minhash Signatures



Simplifying Assumption

- There are ***enough*** buckets that columns are unlikely to hash to the same bucket unless they are ***identical*** in a particular band

- Hereafter, we assume that "***same*** bucket" means "***identical*** in that band"

- Observation
  - The more similar two columns are, the more likely they will be identical in some band
  - Thus, the banding strategy makes similar columns much more likely to be candidate pairs than dissimilar pairs

Analysis of the Banding Technique (1/2)

- Let (d_1, d_2) be a pair of documents that have Jaccard similarity s
- Can we compute the ***probability*** that (d_1, d_2) becomes a candidate pair when we use LSH?
  - Obviously, this probability depends on their Jaccard similarity s
- Example
  - If the Jaccard similarity between d_1 and d_2 is 0.8, the probability of (d_1, d_2) becoming a candidate pair is 99.965%
  - If the Jaccard similarity between d_1 and d_2 is 0.3, the probability of (d_1, d_2) becoming a candidate pair is 4.74%

Analysis of the Banding Technique (2/2)

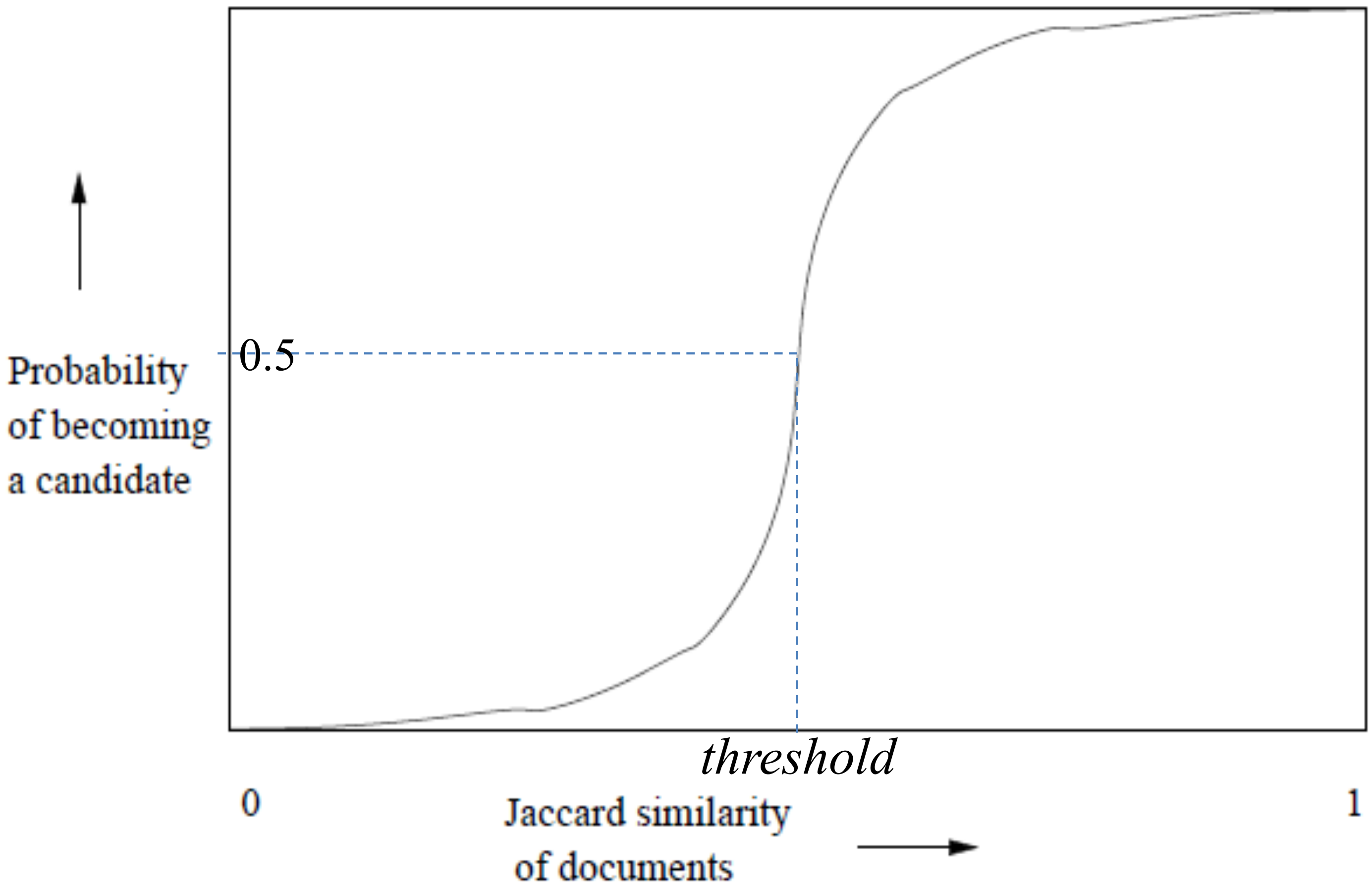
- Suppose we use b bands of r rows each

- Let (d_1, d_2) be a pair of documents that have Jaccard similarity s
  - Thus, the probability the minhash signatures for d_1 and d_2 agree in any one particular row of the signature matrix is s

- The probability that (d_1, d_2) becomes a ***candidate pair***
  - P(the signatures agree in all rows of one particular band) $= s^r$
  - P(the signatures disagree in at least one row of a particular band) $= 1 - s^r$
  - P(the signatures disagree in at least one row of each of the bands) $= (1 - s^r)^b$
  - P(the signatures agree in all the rows of at least one band) = $\mathbf{1 - (1 - s^r)^b}$
    - This is the probability that (d_1, d_2) becomes a candidate pair

$1 - (1 - s^r)^b$

- This function has the form of an ***S-curve***
  - Exactly the shape we want (i.e., pairs with similarity above threshold have a high probability of becoming a candidate, while those below the threshold have a low probability of becoming a candidate)



Threshold

- The value of similarity at which the probability of becoming a candidate is $1/2$
  - Roughly where the rise is the steepest
  - Approximately $(1/b)^{1/r}$

- For large b and r
  - Pairs with similarity above the threshold are ***very likely*** to become candidates
  - Pairs with similarity blow the threshold are ***unlikely*** to become candidates

(Ex) $1 - (1 - s^r)^b$

- When $b = 20$ and $r = 5$
  - Thus, the length of a signature $= 100$

- $1 - (1 - s^5)^{20}$

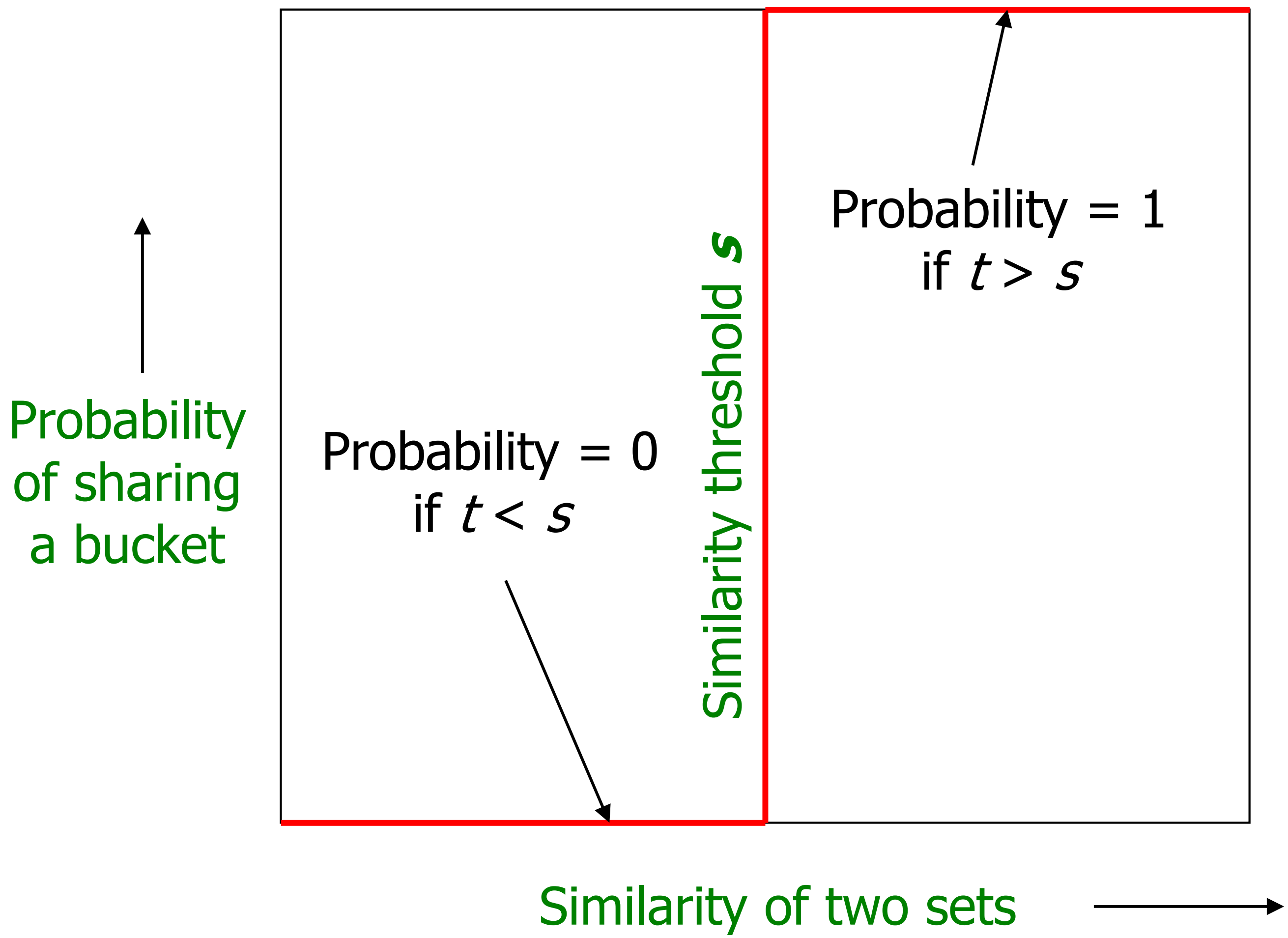
s	$1-(1-s^r)^b$
.2	.006
.3	.047
.4	.186
.5	.470
.6	.802
.7	.975
.8	.9996

The probability that their signatures are identical in a particular band $= 0.8^5 = 33\%$

  - The threshold is just slightly more than 0.5
  - If $s = 0.8$, the probability that their signatures are hashed into the same bucket for at least one of 20 bands is 0.9996

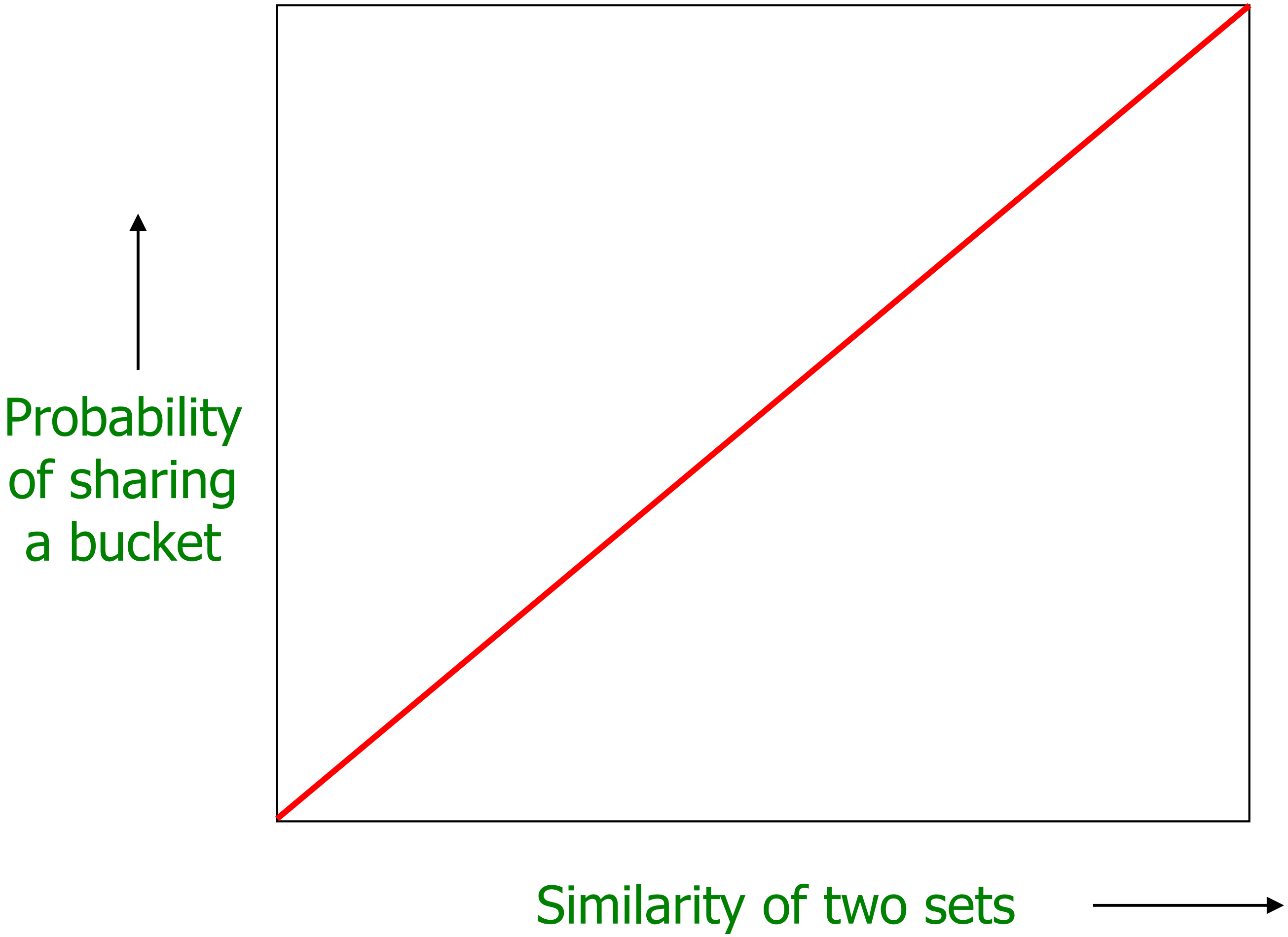
Picking *r* and *b* (1/4)

- Ideal curve



Picking r and b (2/4)

- When $b = 1$ and $r = 1$

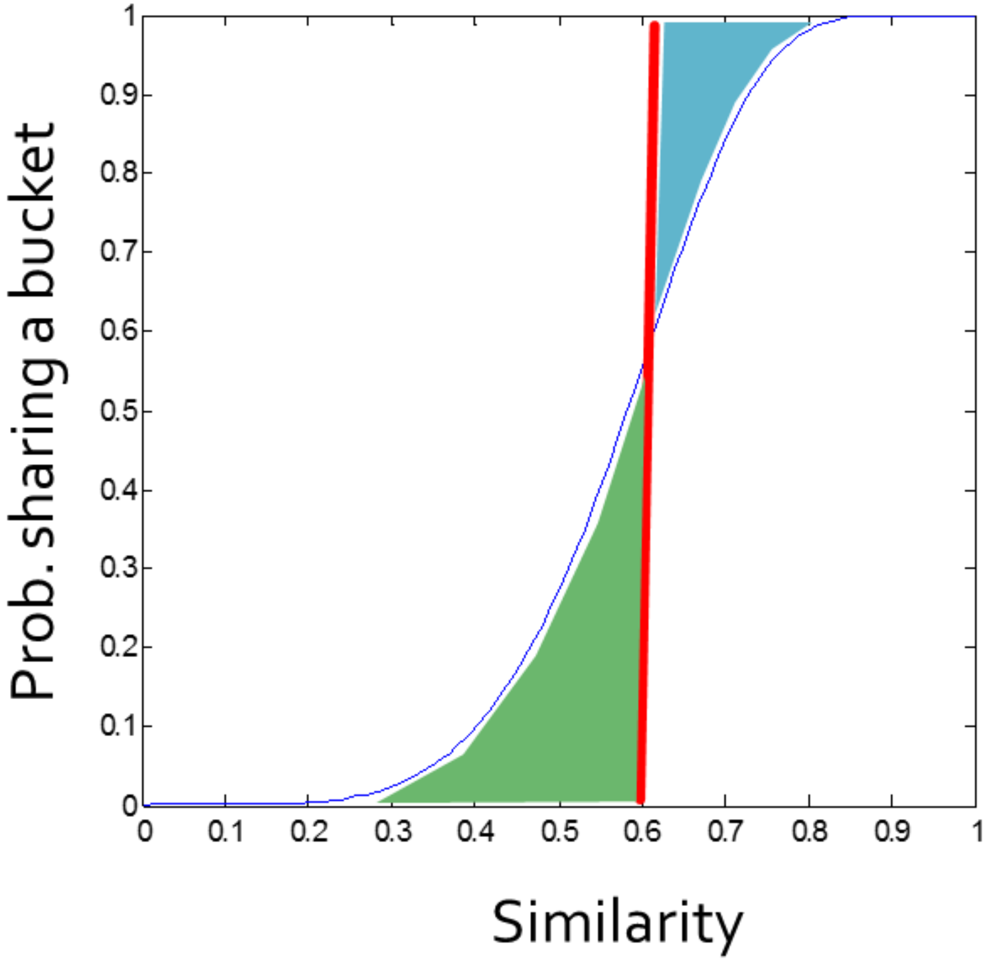


Picking r and b (3/4)

- When b and r are large

Picking r and b (4/4)

- False negative and false positive
  - When $r = 5$ and $b = 10$



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

Combining the Techniques (1/2)

✓ Note that false negatives or false positives can be produced

Step 1: Pick a value of k and construct from each document the set of k-shingles

Step 2: Pick a length n for the minhash signatures and compute the minhash signatures for all the documents

Step 3: Choose a threshold t that defines how similar documents have to be regarded as a similar pair. Pick a number of bands b and a number of rows r such that $br = n$, and the threshold t is approximately $(1/b)^{1/r}$

Combining the Techniques (2/2)

Step 5: Construct candidate pairs by applying the LSH technique

Step 6: Examine each candidate pair's signatures and determine whether the faction of components in which they agree is at least t

Step 7 (optional): If the signatures are sufficiently similar, go to the document themselves and check that they are truly similar

- ✓ Documents can have similar signatures by luck

Distance Measures

Distance Measures

- Measure of ***closeness***

- Example: Jaccard distance
  - 1 minus the Jaccard similarity
  - A distance measure for ***sets***
  - The closer sets are, the lower the Jaccard distance

- There are a number of other distance measures that make sense in some applications
  - Euclidean distance, cosine distance, edit distance, hamming distance

Definition of a Distance Measure

- Suppose we have a set of points, called a ***space***
- Let x and y be two points in the space
- A ***distance*** measure on this space is a function $d(x, y)$ that satisfies the following condition:
  - $d(x, y) \geq 0$ (no negative distance)
  - $d(x, y) = 0$ if and only if $x = y$ (distance is zero from a point to itself)
  - $d(x, y) = d(y, x)$ (distance is symmetric)
  - $d(x, y) \leq d(x, z) + d(z, y)$ (the ***triangle inequality***)
    - To travel from x to y, we cannot obtain any benefit if we are forced to travel via some particular third point z

Euclidean Distance (1/2)

- The most familiar distance measure
  - The one we normally think of as “distance”
- n-dimensional Euclidean space
  - A space where points are vectors of n real numbers (e.g., $[x_1, x_2, \ldots, x_n]$)
- L_2-norm

$$d([x_1, x_2, \ldots, x_n],\ [y_1, y_2, \ldots, y_n]) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

  - Note that all the requirements for a distance measure are satisfied

Euclidean Distance (2/2)

- L_r-norm (general form)

$$d([x_1, x_2, \ldots, x_n],\ [y_1, y_2, \ldots, y_n]) = (\sum_{i=1}^{n} |x_i - y_i|^r)^{1/r}$$

- L_1-norm (called ***Manhattan distance***)
  - The distance one would have to travel between points if one were constrained to travel among the streets of a city such as Manhattan

- L_∞-norm

$$d([x_1, x_2, \ldots, x_n], [y_1, y_2, \ldots, y_n]) = \max_i |x_i - y_i|$$

  - The maximum of $|x_i - y_i|$ over all dimensions i
  - As r gets larger, only the dimension with the largest difference matters

(Ex) Euclidean Distance

- Consider the two-dimensional Euclidean space and the two points $(2, 7)$ and $(6, 4)$

- L_2-norm
  - $(|2 - 6|^2 + |7 - 4|^2)^{1/2} = 5$

- L_1-norm
  - $|2 - 6| + |7 - 4| = 7$

- L_∞-norm
  - $\max(|2 - 6|, |7 - 4|) = 4$

Jaccard Distance

- $d(x, y) = 1 - SIM(x, y) = 1 - |x \cap y|/|x \cup y|$
  - In other words, the probability that a random minhash function does ***not*** send x and y to the same value

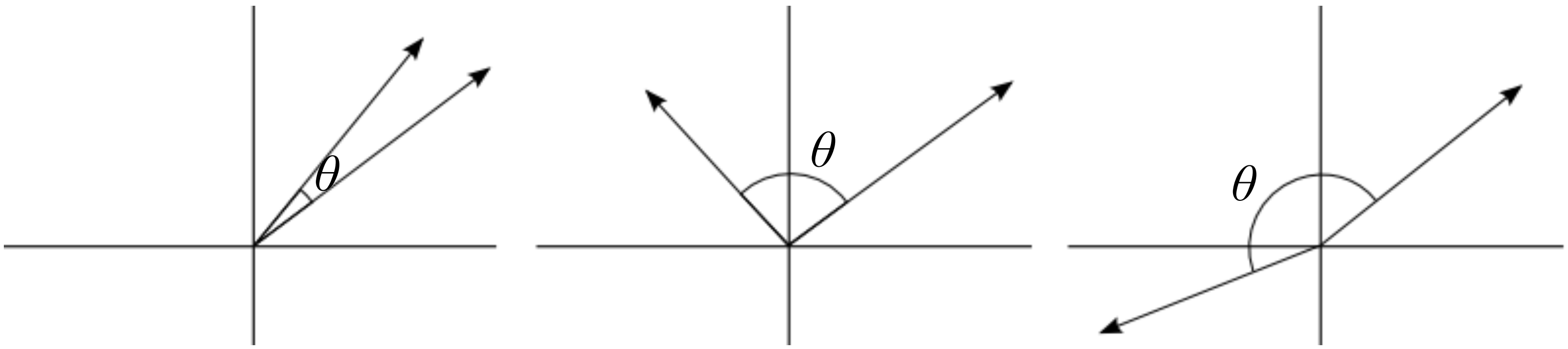
- Note that all the requirements for a distance measure are satisfied
  - $d(x, y) \geq 0$, because $|x \cap y| \leq |x \cup y|$
  - $d(x, y) = 0$ iff $x = y$, because $x \cap x = x \cup x = x$
  - $d(x, y) = d(y, x)$, because $x \cup y = y \cup x$ and $x \cap y = y \cap x$
  - $d(x, y) \leq d(x, z) + d(z, y)$
    - We show $P(h(x) \neq h(y)) \leq P(h(x) \neq h(z)) + P(h(z) \neq h(y))$
    - This is true because whenever $h(x) \neq h(y)$, at least one of $h(x)$ and $h(y)$ must be different from $h(z)$

Cosine Distance

- The cosine distance between two points $x = [x_1, x_2, \dots, xn]$ and $y = [y_1, y_2, \dots, yn]$ measures the ***angle*** that the vectors to those points make

$$\cos(\theta) = \frac{x \cdot y}{||x|| \cdot ||y||} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^{\,2}}\sqrt{\sum_{i=1}^{n} y_i^{\,2}}}$$

  - This angle will be in the range 0° to 180°



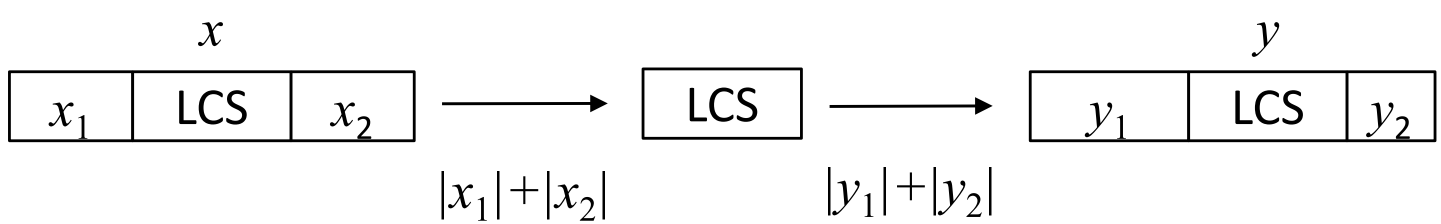
- Note that all the requirements for a distance measure are satisfied

Edit Distance (1/2)

- A distance measure for ***strings***
- The edit distance between two strings $x = x_1x_2\ldots x_n$ and $y = y_1y_2\ldots y_n$ is the ***smallest*** number of insertions and deletions of single characters that will convert x to y
- Example
  - The edit distance between $x =$ `abcde` and $y =$ `acfdeg` is 3
    - To convert x to y
      1. Delete `b`
      2. Insert `f` after `c`
      3. Insert `g` after `e`
    - No sequence of fewer than 3 insertions and/or deletions will convert x to y

Edit Distance (2/2)

- Calculation of the edit distance $d(x, y)$
  - Compute a longest common subsequence (LCS) of x and y
  - $d(x, y) = |x| + |y| - 2|\text{LCS of } x \text{ and } y|$



- Example
  - $d(\texttt{abcde}, \texttt{acfdeg}) = |\texttt{abcde}| + |\texttt{acfdeg}| - 2|\texttt{acde}| = 5 + 6 - 2{\cdot}4 = 3$
  - $d(\texttt{aba}, \texttt{bab}) = |\texttt{aba}| + |\texttt{bab}| - 2|\texttt{ab}| = 3 + 3 - 2{\cdot}2 = 2$
    - Or, $|\texttt{aba}| + |\texttt{bab}| - 2|\texttt{ba}| = 3 + 3 - 2{\cdot}2 = 2$

- Note that all the requirements for a distance measure are satisfied

Hamming Distance

- A distance measure for ***vectors***

- The Hamming distance between two vectors is the number of components in which they ***differ***
  - (ex) $d([2, \mathbf{1}, \mathbf{7}], [2, \mathbf{2}, \mathbf{3}]) = 2$, $d(1\mathbf{0}1\mathbf{01}, 1\mathbf{1}1\mathbf{10}) = 3$

- Note that all the requirements for a distance measure are satisfied
  - $d(x, y) \geq 0$ (it is clear)
  - $d(x, y) = 0$ iff $x = y$ (if the distance is zero, then the vectors are identical)
  - $d(x, y) = d(y, x)$ (the distance doesn't depend on the order of two vectors)
  - $d(x, y) \leq d(x, z) + d(z, y)$
    - If x and z differ in m components, and z and y differ in n components, then x and y cannot differ in more than $m + n$ components