Finding Similar Sets

Applications
Shingling
Minhashing
Locality-Sensitive Hashing

Mining of Massive Datasets

Leskovec, Rajaraman, and Ullman
Stanford University

Applications of Set-Similarity

Many data-mining problems can be expressed as finding "similar" sets:

1. Pages with similar words, e.g., for classification by topic.
2. NetFlix users with similar tastes in movies, for recommendation systems.
3. Dual: movies with similar sets of fans.
4. Entity resolution.

定义：不同的数据提供方对同一个事物即实体 (Entity)可能会有不同的描述 (这 里的描述包括数据格式 、表示方法 等)，每一个对实体的描述称为该实体的一个引用。实体解析，是指从一个“ 引用集合 ”中解析并映射到现实世界中的“ 实体 ”过程 。

实体解析(Entity Resolution)又被称为记录链接(Record Linkage) 、对象识别(object Identification) 、个体识别(Individual Identification) 、重复检测(Duplicate Detection)
https://www.cnblogs.com/nolonely/p/5399695.html

Similar Documents

- Given a body of documents, e.g., the Web, find pairs of documents with a lot of text in common, such as:
  - Mirror sites, or approximate mirrors.
    - Application: Don't want to show both in a search.
  - Plagiarism, including large quotations.
  - Similar news articles at many news sites.
    - Application: Cluster articles by "same story."

核心思想就是从一些修改或者被污染的数据中找出来和原来的东西一样的或者是镜像：比如抄袭，或者是相同类容的新闻或者文章筛选

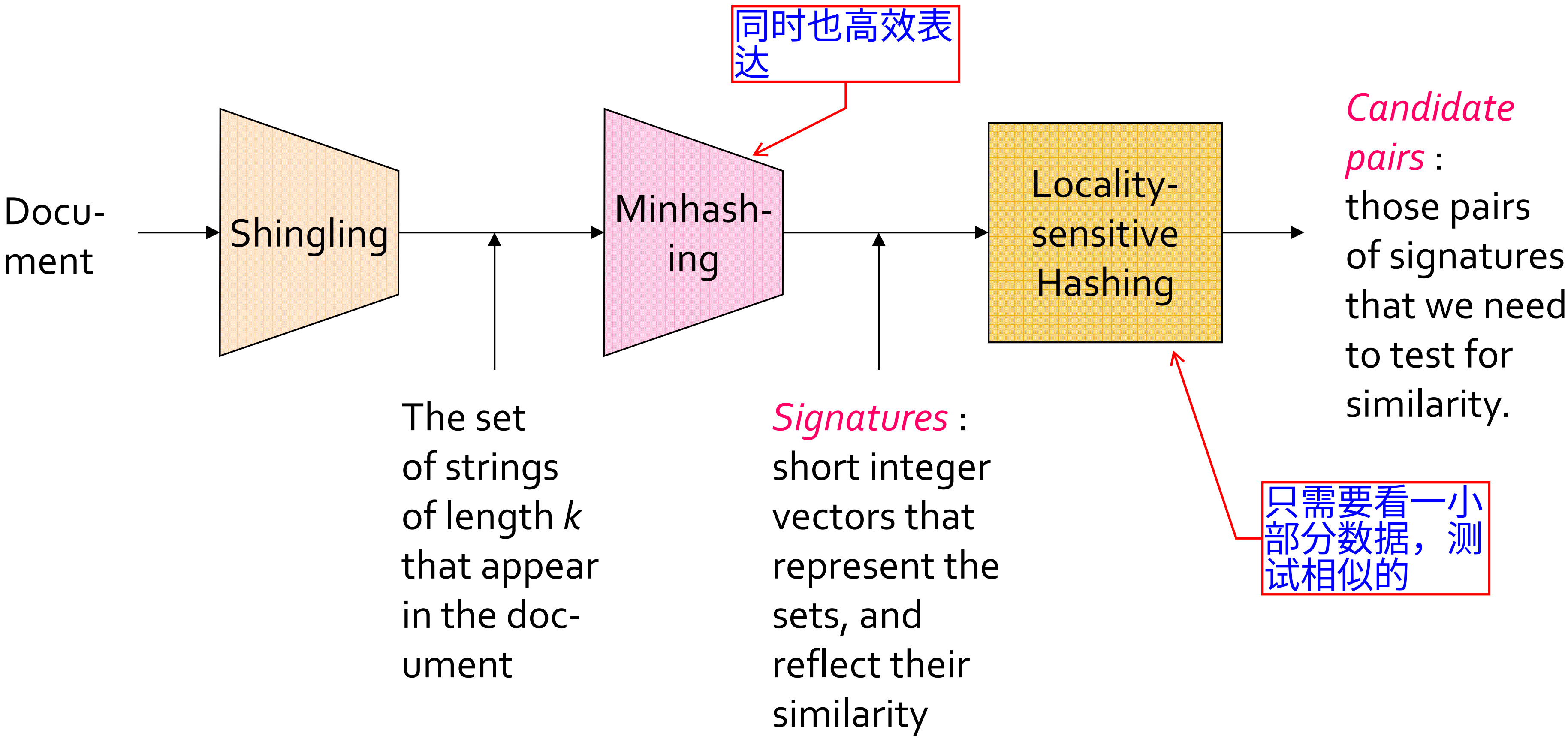
Three Essential Techniques for Similar Documents

还可以告诉我们，shingling搞出来的东西究竟有多相似

1. *Shingling* : convert documents, emails, etc., to sets. a lot of text in common
2. *Minhashing* : convert large sets to short signatures, while preserving similarity.
3. *Locality-sensitive hashing* : focus on pairs of signatures likely to be similar.

LSH最根本的作用，就是能高效处理海量高维数据的最近邻问题
Locality-sensitive hashing (LSH) reduces the dimensionality of high-dimensional data. LSH hashes input items so that similar items map to the same "buckets" with high probability (the number of buckets being much smaller than the universe of possible input items).
Locality-sensitive hashing has much in common with data clustering and nearest neighbor search.

The Big Picture



Shingles

- A *k* -shingle (or *k* -gram) for a document is a sequence of *k* characters that appears in the document. k=5,10经常用
- Example: k=2; doc = abcab. Set of 2-shingles = {ab, bc, ca}.
- Represent a doc by its set of *k*-shingles.

Shingles and Similarity

- Documents that are intuitively similar will have many shingles in common.
- Changing a word only affects k-shingles within distance k from the word.
- Reordering paragraphs only affects the 2k shingles that cross paragraph boundaries.
- Example: k=3, "The dog which chased the cat" versus "The dog that chased the cat".
  7个shingles
  6个shingles
  - Only 3-shingles replaced are g_w, _wh, whi, hic, ich, ch_, and h_c.

Shingles: Compression Option

http://ethen8181.github.io/machine-learning/clustering_old/text_similarity/text_similarity.html

- To compress long shingles, we can hash them to (say) 4 bytes.
  - Called *tokens*.
- Represent a doc by its tokens, that is, the set of hash values of its *k*-shingles.
- Two documents could (rarely) appear to have shingles in common, when in fact only the hash-values were shared.

Minhashing

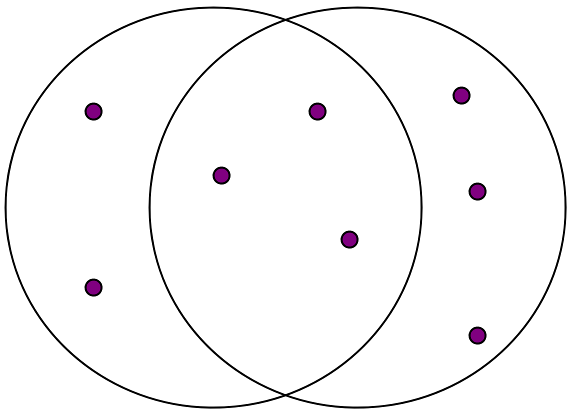
Jaccard Similarity Measure
Constructing Signatures

Jaccard Similarity

- The *Jaccard similarity* of two sets is the size of their intersection divided by the size of their union.
- $Sim(C_1, C_2) = |C_1 \cap C_2| / |C_1 \cup C_2|$.

雅卡尔指数（英语：Jaccard index），又称为并交比（Intersection over Union）、雅卡尔相似系数（Jaccard similarity coefficient），是用于比较样本集的相似性与多样性的统计量。雅卡尔系数能够量度有限样本集合的相似度，其定义为两个集合交集大小与并集大小之间的比例：

Example: Jaccard Similarity



3 in intersection.
8 in union.
Jaccard similarity
= 3/8

From Sets to Boolean Matrices

- Rows = elements of the universal set.
  - Example: the set of all k-shingles.
- Columns = sets.
- 1 in row *e* and column *S* if and only if *e* is a member of *S*.
- Column similarity is the Jaccard similarity of the sets of their rows with 1.
- Typical matrix is sparse.

Example: Column Similarity

C_1	C_2		
0	1		*
1	0		*
1	1	*	*
0	0		
1	1	*	*
0	1		*

可以想象成是顾客买了亚马逊的书，row是书的种类，col是不同的顾客。矩阵稀疏，因为每个人只会买少部分的书。Column Similarity就可以找出顾客的相似性

$$\text{Sim}(C_1, C_2) = 2/5 = 0.4$$

Four Types of Rows

- Given columns C_1 and C_2, rows may be classified as:

	C_1	C_2
a	1	1
b	1	0
c	0	1
d	0	0

- Also, *a* = # rows of type *a* , etc.
- Note $Sim(C_1, C_2) = a/(a + b + c)$.

Minhashing

https://my.oschina.net/keyven/blog/628898
在经过随机行打乱后，两个集合的最小哈希值相等的概率等于这两个集合的Jaccard相似度

- Imagine the rows permuted randomly.
- Define *minhash function* $h(C)$ = the number of the first (in the permuted order) row in which column C has 1.
- Use several (e.g., 100) independent hash functions to create a signature for each column.
- The signatures can be displayed in another matrix – the *signature matrix* – whose columns represent the sets and the rows represent the minhash values, in order for that column.

https://blog.csdn.net/liujan511536/article/details/47729721
　为了计算最小哈希，首先对特征矩阵的行进行打乱（也即随机调换行与行之间的位置），这个打乱是随机的。然后某一列的最小哈希值就等于打乱后的这一列第一个值为1的行所在的行号（不明白的直接看例子），行号从0开始。

Minhashing Example

签名矩阵比特征矩阵小很多 https://blog.csdn.net/liujan511536/article/details/47729721

被打乱的顺序，按照这个顺序找第一个不为零的数

Input matrix

1	4	3	1	0	1	0
3	2	4	1	0	0	1
7	1	7	0	1	0	1
6	3	6	0	1	0	1
2	6	1	0	1	0	1
5	7	2	1	0	1	0
4	5	5	1	0	1	0

Signature matrix *M*

2	1	2	1
2	1	4	1
1	2	1	2

明显可以看出来，矩阵已经被压缩了

Surprising Property

- The probability (over all permutations of the rows) that $h(C_1) = h(C_2)$ is the same as $Sim(C_1, C_2)$.
- Both are $a/(a+b+c)$!
- Why?
  - Look down the permuted columns C_1 and C_2 until we see a 1.
  - If it's a type-*a* row, then $h(C_1) = h(C_2)$. If a type-*b* or type-*c* row, then not.

Similarity for Signatures

- The *similarity of signatures* is the fraction of the minhash functions in which they agree.
  - Thinking of signatures as columns of integers, the similarity of signatures is the fraction of rows in which they agree.
- Thus, the expected similarity of two signatures equals the Jaccard similarity of the columns or sets that the signatures represent.
  - And the longer the signatures, the smaller will be the expected error.

几百次的minhash之后，这个ture jaccard和two signature 来衡量相似度的值是一样的，with 非常小的error.见17页ppt,最上

Min Hashing – Example

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

Input matrix

1	0	1	0
1	0	0	1
0	1	0	1
0	1	0	1
0	1	0	1
1	0	1	0
1	0	1	0

Signature matrix *M*

2	1	2	1
2	1	4	1
1	2	1	2

	1-3	2-4	1-2
Col/Col	0.75	0.75	0
Sig/Sig	0.67	1.00	0

就是2/3

Implementation of Minhashing

- Suppose 1 billion rows.
- Hard to pick a random permutation of 1…billion.
- Representing a random permutation requires 1 billion entries.
- Accessing rows in permuted order leads to thrashing.

Implementation – (2)

- A good approximation to permuting rows: pick, say, 100 hash functions.
- For each column c and each hash function h_i, keep a "slot" $M(i, c)$.
- Intent: $M(i, c)$ will become the smallest value of $h_i(r)$ for which column c has 1 in row r.
  - I.e., $h_i(r)$ gives order of rows for i^{th} permutation.

Implementation – (3)

for each row r **do begin**
  for each hash function h_i **do**
    compute $h_i(r)$;
  for each column c
    if c has 1 in row r
      for each hash function h_i **do**
        if $h_i(r)$ is smaller than $M(i, c)$ **then**
          $M(i, c) := h_i(r)$;
end;

https://blog.csdn.net/liujan511536/article/details/47729721

Example

变化

Row	C1	C2
1	1	0
2	0	1
3	1	1
4	1	0
5	0	1

	Sig1	Sig2
$h(1) = 1$	1	∞
$g(1) = 3$	3	∞
$h(2) = 2$	1	2
$g(2) = 0$	3	0
$h(3) = 3$	1	2
$g(3) = 2$	2	0
$h(4) = 4$	1	2
$g(4) = 4$	2	0
$h(5) = 0$	1	0
$g(5) = 1$	2	0

最小的才替换

$h(x) = x \bmod 5$

$g(x) = (2x+1) \bmod 5$

Implementation – (4)

- Often, data is given by column, not row.
  - Example: columns = documents, rows = shingles.
- If so, sort matrix once so it is by row.

Start with a list of row column pairs where the ones are.
Initially sort it by column, and sort these pairs by row.

Locality-Sensitive Hashing

Focusing on Similar Minhash Signatures
Other Applications Will Follow

Locality-Sensitive Hashing

- General idea: Generate from the collection of all elements (signatures in our example) a small list of *candidate pairs*: pairs of elements whose similarity must be evaluated.
- For signature matrices: Hash columns to many buckets, and make elements of the same bucket candidate pairs.

这样做的缺点是当文档数量很多时，要比较的次数会非常大。那么我们可不可以只比较那些相似度可能会很高的文档，而直接忽略过那些相似度很低的文档。

Candidate Generation From Minhash Signatures

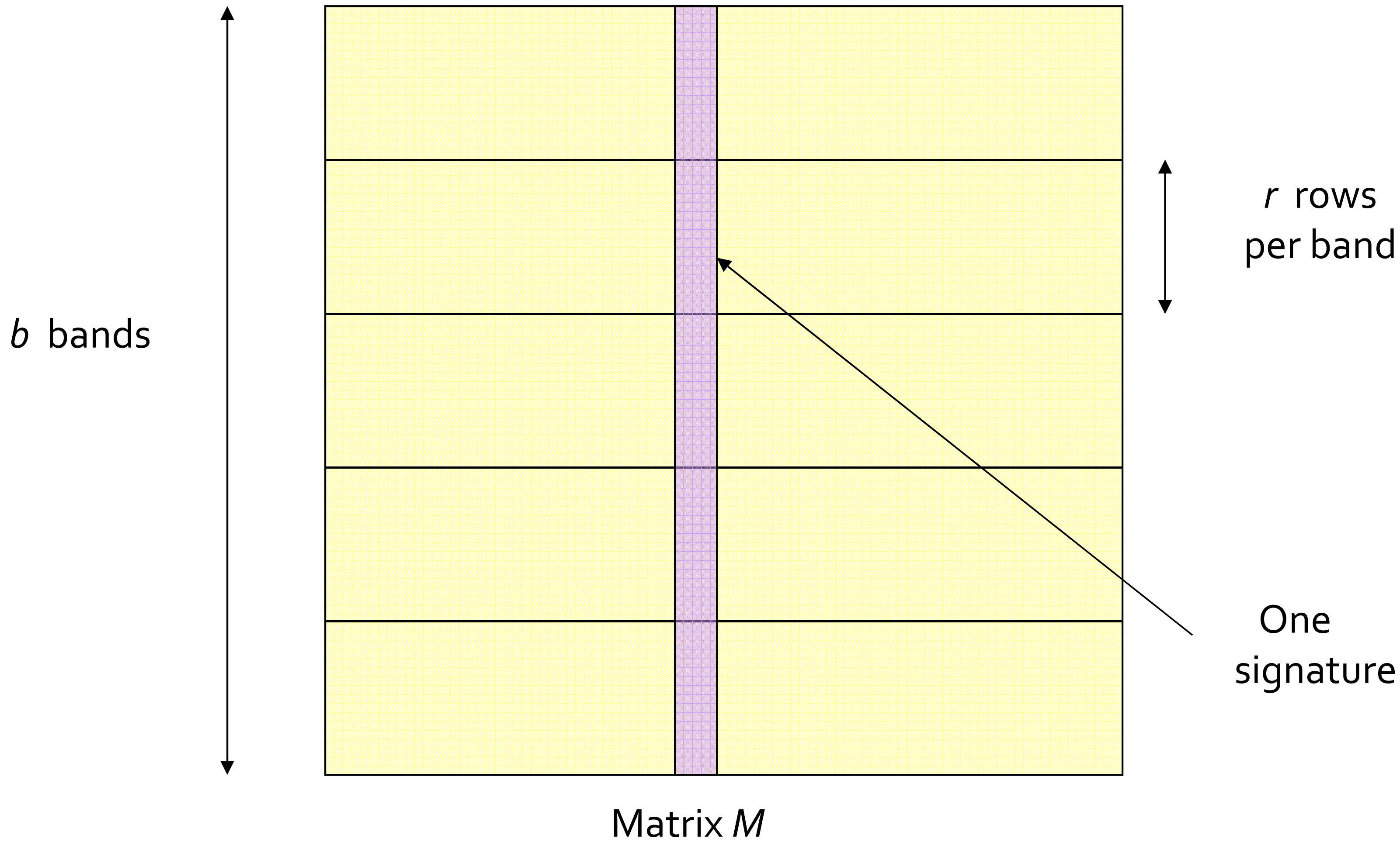
https:// https://blog.csdn.net/liujan511536/article/details/47729721

- Pick a similarity threshold t, a fraction < 1.
- We want a pair of columns c and d of the signature matrix M to be a *candidate pair* if and only if their signatures agree in at least fraction t of the rows.
  - I.e., $M(i, c) = M(i, d)$ for at least fraction t values of i.

LSH for Minhash Signatures

- Big idea: hash columns of signature matrix *M* several times.
- Arrange that (only) similar columns are likely to hash to the same bucket.
- Candidate pairs are those that hash at least once to the same bucket.

Partition Into Bands

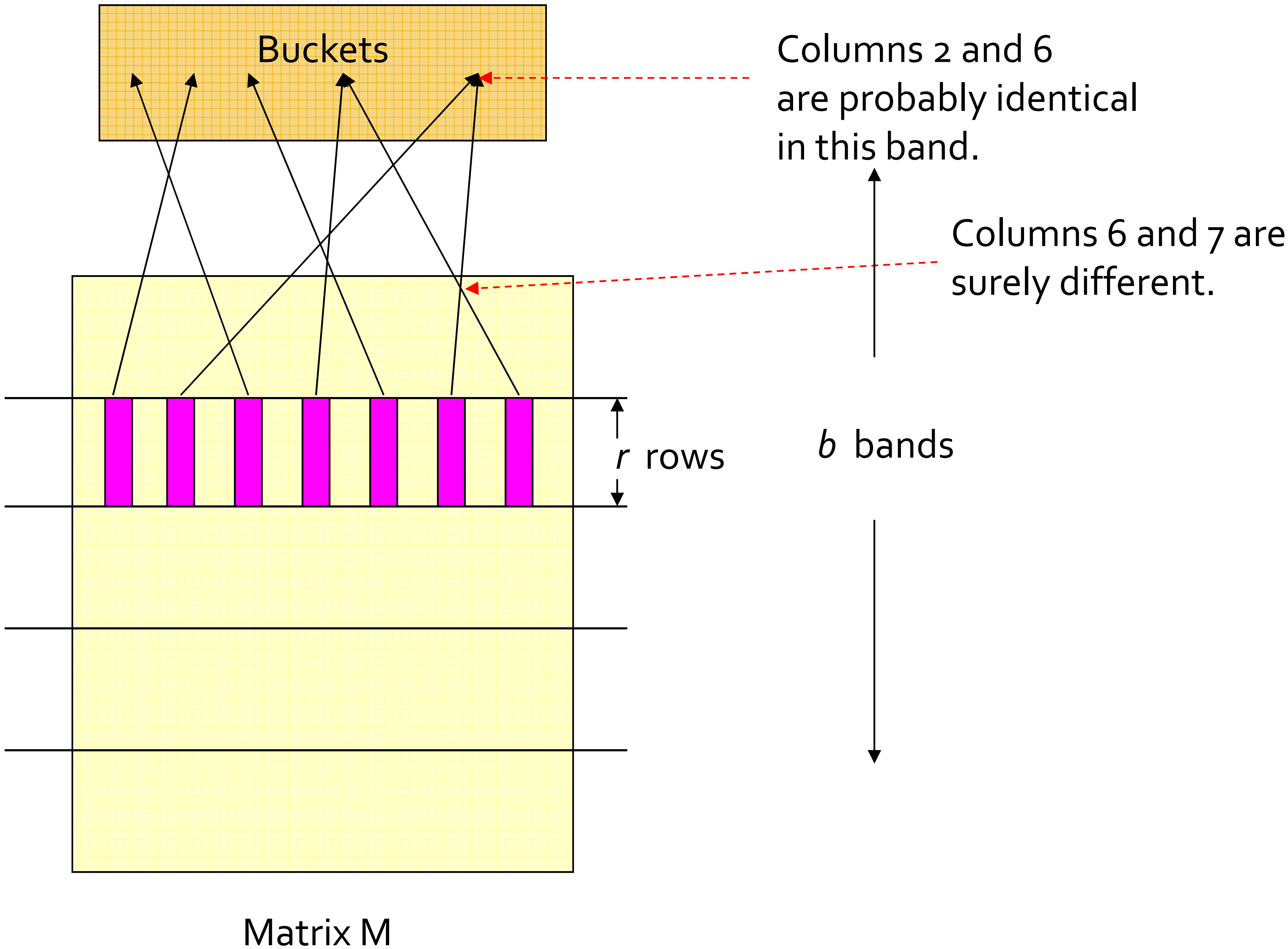


Partition into Bands – (2)

- Divide matrix M into b bands of r rows.
- For each band, hash its portion of each column to a hash table with k buckets.
  - Make k as large as possible.
- *Candidate* column pairs are those that hash to the same bucket for ≥ 1 band.
- Tune b and r to catch most similar pairs, but few nonsimilar pairs.

可以对所有行条使用相同的哈希函数，但是对于每个行条我们都使用一个独立的桶数组，因此即便是不同行条中的相同列向量，也不会被哈希到同一个桶中。这样，只要两个集合在某个行条中有落在相同桶的两列，这两个集合就被认为可能相似度比较高，作为后续计算的候选对；而那些在所有行条中都不落在同一个桶中的两列，就会被认为相似度不会很高，而被直接忽略。

Hash Function for One Bucket



Example – Bands

- Suppose 100,000 columns.
- Signatures of 100 integers.
- Therefore, signatures take 40Mb.
- Want all 80%-similar pairs of documents.
- 5,000,000,000 pairs of signatures can take a while to compare.
- Choose 20 bands of 5 integers/band.

4byte 一个

100，000取2

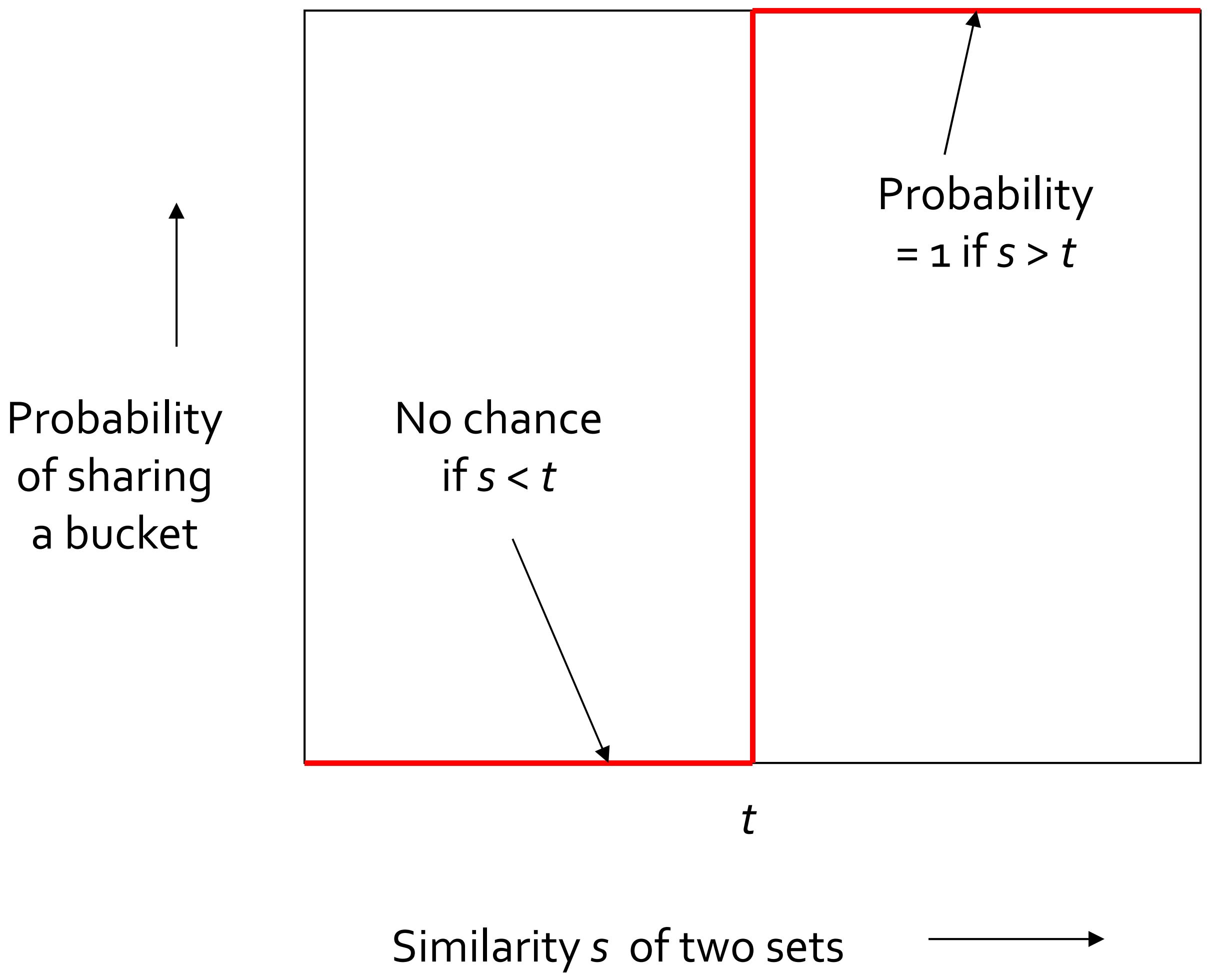
Suppose C_1, C_2 are 80% Similar

- Probability C_1, C_2 identical in one particular band: $(0.8)^5 = 0.328$.
- Probability C_1, C_2 are *not* similar in any of the 20 bands: $(1-0.328)^{20} = .00035$.
  不是candidate的概率
  - i.e., about 1/3000th of the 80%-similar underlying sets are false negatives.

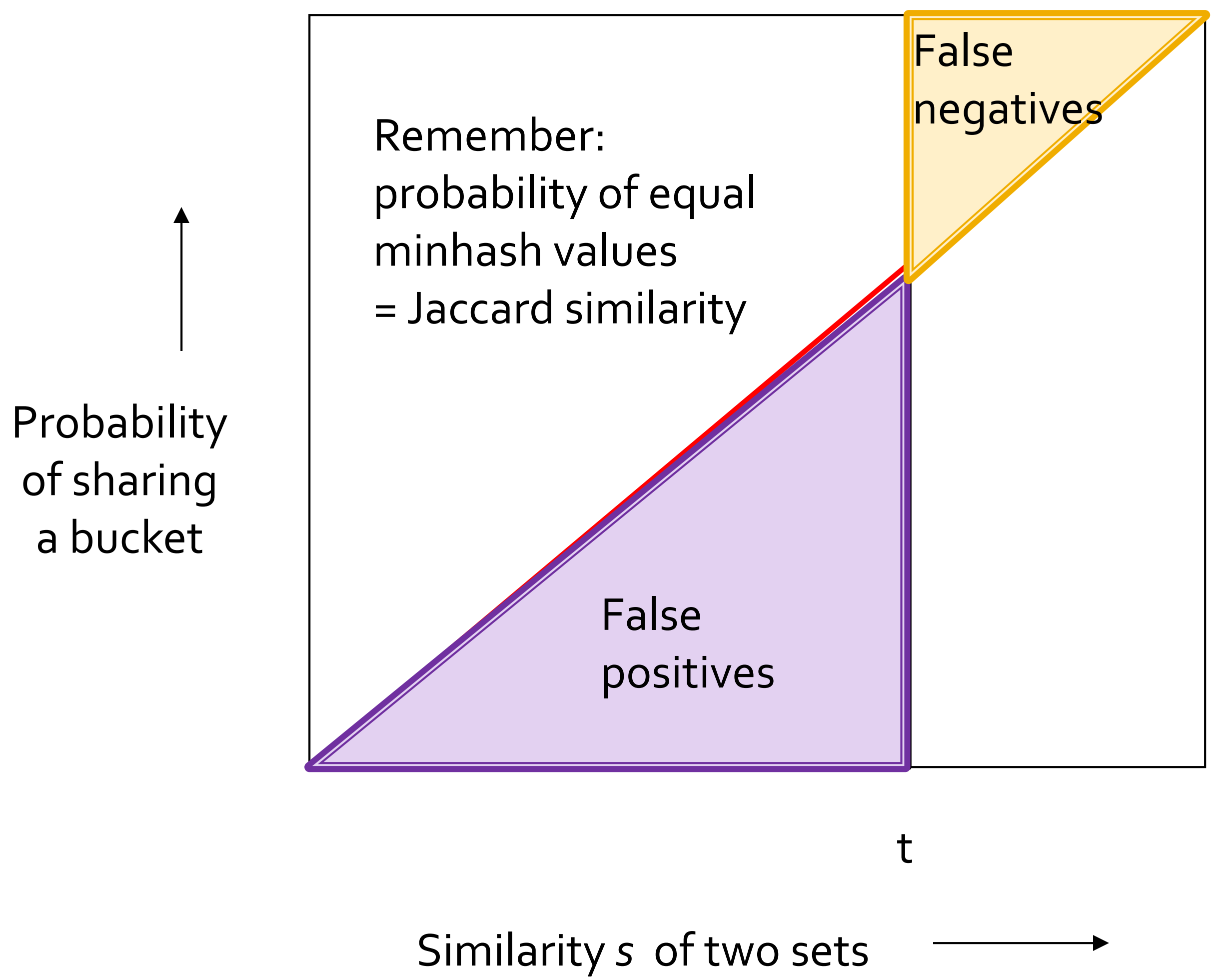
Suppose C_1, C_2 Only 40% Similar

- Probability C_1, C_2 identical in any one particular band: $(0.4)^5 = 0.01$.
- Probability C_1, C_2 identical in $\geq$ 1 of 20 bands: $\leq 20 * 0.01 = 0.2$.
- But false positives much lower for similarities << 40%.

Analysis of LSH – What We Want



What One Band of One Row Gives You



What b Bands of r Rows Gives You

至少有一个是一样的，candidate，分到一个类

At least one band identical

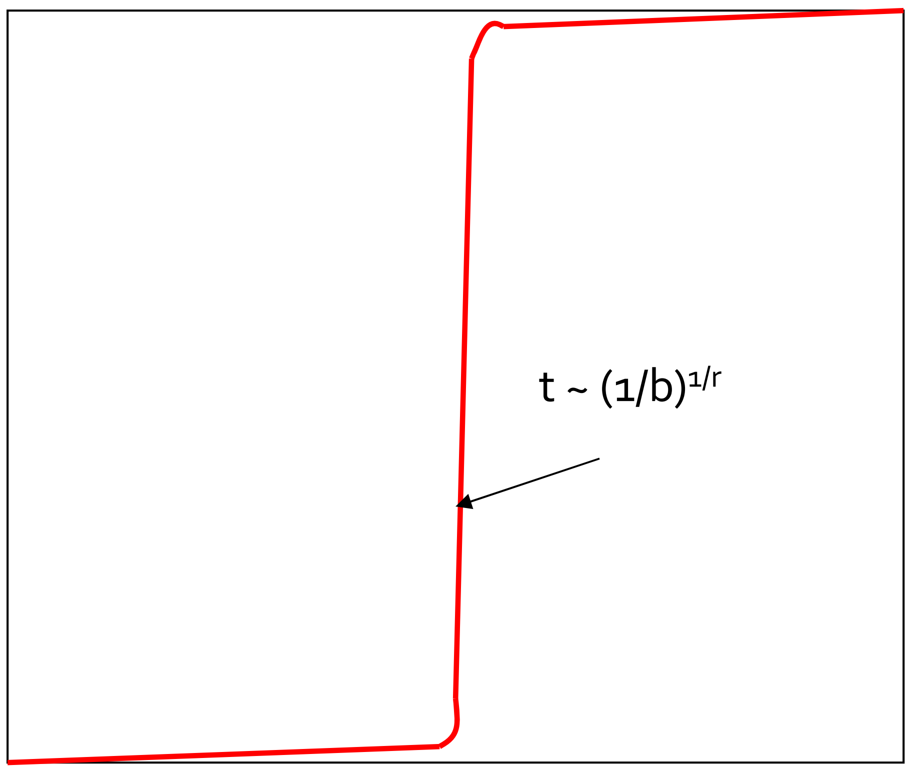
No bands identical

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

Some row of a band unequal

All rows of a band are equal

Probability of sharing a bucket



t

Similarity *s* of two sets

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

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

LSH Summary

- Tune to get almost all pairs with similar signatures, but eliminate most pairs that do not have similar signatures.
- Check that candidate pairs really do have similar signatures.
- Optional: In another pass through data, check that the remaining candidate pairs really represent similar *sets* .

By computing the Jaccard similarity of the underlying sets,
we can eliminate the false positives.
Unfortunately, we cannot eliminate false negatives this way.

Applications of LSH

Entity Resolution
Fingerprints
Similar News Articles

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University

Entity Resolution

- The *entity-resolution* problem is to examine a collection of records and determine which refer to the same entity.
  - *Entities* could be people, events, etc.
- Typically, we want to merge records if their values in corresponding fields are similar.

Matching Customer Records

- I once took a consulting job solving the following problem:
  - Company A agreed to solicit customers for Company B, for a fee.
  - They then argued over how many customers.
  - Neither recorded exactly which customers were involved.

Customer Records – (2)

- Each company had about 1 million records describing customers that might have been sent from A to B.
- Records had name, address, and phone, but for various reasons, they could be different for the same person.

Customer Records – (3)

- Step 1: Design a measure ("*score* ") of how similar records are:
  - E.g., deduct points for small misspellings ("Jeffrey" vs. "Jeffery") or same phone with different area code.
- Step 2: Score all pairs of records that the LSH scheme identified as candidates; report high scores as matches.

Customer Records – (4)

- Problem: (1 million)^2 is too many pairs of records to score.
- Solution: A simple LSH.
  - Three hash functions: exact values of name, address, phone.
    - Compare iff records are identical in at least one.
  - Misses similar records with a small differences in all three fields.

Aside: Hashing Names, Etc.

- How do we hash strings such as names so there is one bucket for each string?
- Answer: Sort the strings instead.
- Another option was to use a few million buckets, and deal with buckets that contain several different strings.

Aside: Validation of Results

- We were able to tell what values of the scoring function were reliable in an interesting way.
- Identical records had a creation date difference of 10 days.
- We only looked for records created within 90 days of each other, so bogus matches had a 45-day average.

Validation – (2)

- By looking at the pool of matches with a fixed score, we could compute the average time-difference, say x, and deduce that fraction (45-x)/35 of them were valid matches.
- Alas, the lawyers didn't think the jury would understand.

Validation – Generalized

- Any field not used in the LSH could have been used to validate, provided corresponding values were closer for true matches than false.
- Example: if records had a height field, we would expect true matches to be close, false matches to have the average difference for random people.

Fingerprint Matching

Minutiae
A New Way of Bucketing

Mining of Massive Datasets

Leskovec, Rajaraman, and Ullman
Stanford University

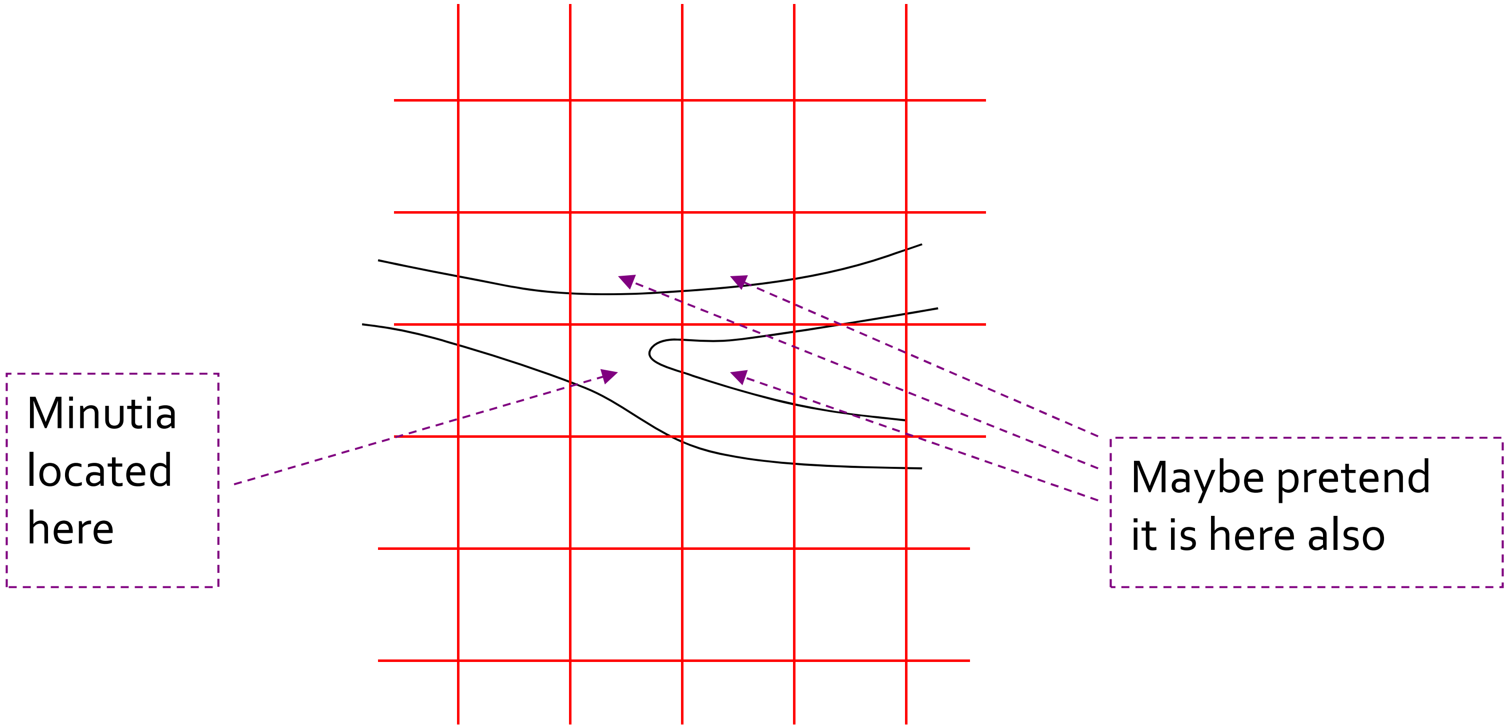
Fingerprint Comparison

- Represent a fingerprint by the set of positions of *minutiae*. 细节点
  - These are features of a fingerprint, e.g., points where two ridges come together or a ridge ends.

LSH for Fingerprints

- Place a grid on a fingerprint.
  - Normalize so identical prints will overlap.
- Set of grid squares where minutiae are located represents the fingerprint.
- Possibly, treat minutiae near a grid boundary as if also present in adjacent grid points.

Discretizing Minutiae



Applying LSH to Fingerprints

- Fingerprint = set of grid squares.
- No need to minhash, since the number of grid squares is not too large.
- Represent each fingerprint by a bit-vector with one position for each square.
  - 1 in only those positions whose squares have minutiae.

LSH/Fingerprints – (2)

- Pick 1024 (?) sets of 3 (?) grid squares (components of the bit vectors), randomly.
- For each set of three squares, two prints that each have 1 for all three squares are candidate pairs.
- Funny sort of ‘bucketization.”
  - Each set of three squares creates one bucket.
  - Prints can be in many buckets.

Example: LSH/Fingerprints

- Suppose typical fingerprints have minutiae in 20% of the grid squares.
- Suppose fingerprints from the same finger agree in at least 80% of their squares.
- Probability two random fingerprints each have minutiae in all three squares = $(0.2)^6$ = .000064.

Example: Continued

First print has has minutia in this square

Second print of the same finger also has minutia in that square

- Probability two fingerprints from the same finger each have 1's in three given squares = $(\boxed{(0.2)}\boxed{(0.8)})^3 = .004096$.
- Prob. for at least one of 1024 sets of three points = $1-(1-.004096)^{1024} = .985$.
- But for random fingerprints: $1-(1-.000064)^{1024} = .063$.

1.5% false negatives

6.3% false positives

Finding Duplicate News Articles

A New Way of Shingling
Bucketing by Length

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University

Application: Same News Article

- The Political-Science Dept. at Stanford asked a team from CS to help them with the problem of identifying duplicate, on-line news articles.
- Problem: the same article, say from the Associated Press, appears on the Web site of many newspapers, but looks quite different.

News Articles – (2)

- Each newspaper surrounds the text of the article with:
  - It’s own logo and text.
  - Ads.
  - Perhaps links to other articles.
- A newspaper may also “crop” the article (delete parts).

News Articles – (3)

- The team came up with its own solution, that included shingling, but not minhashing or LSH.
  - A special way of shingling that appears quite good for this application.
  - LSH substitute: candidates are articles of similar length.

Enter LSH

- I told them the story of minhashing + LSH.
- They implemented it and found it faster for similarities below 80%.
  - Aside: That's no surprise. When similarity is high, there are better methods, as we shall see.

Enter LSH – (2)

- Their first attempt at minhashing was very inefficient.
- They were unaware of the importance of doing the minhashing row-by-row.
- Since their data was column-by-column, they needed to sort once before minhashing.

But the problem was that I forgot to remind them to do the minhashing row by row, where you compute the hash value for each row number once and for all rather than once for each column. Remember that the rows correspond to the shingles and the columns to the web pages.

Specialized Shingling Technique

- The team observed that news articles have a lot of *stop words*, while ads do not.
  - "Buy Sudzo" vs. "I recommend that you buy Sudzo for your laundry."
- They defined a *shingle* to be a stop word and the next two following words.

Why it Works

- By requiring each shingle to have a stop word, they biased the mapping from documents to shingles so it picked more shingles from the article than from the ads.
- Pages with the same article, but different ads, have higher Jaccard similarity than those with the same ads, different articles.