Similarity Search

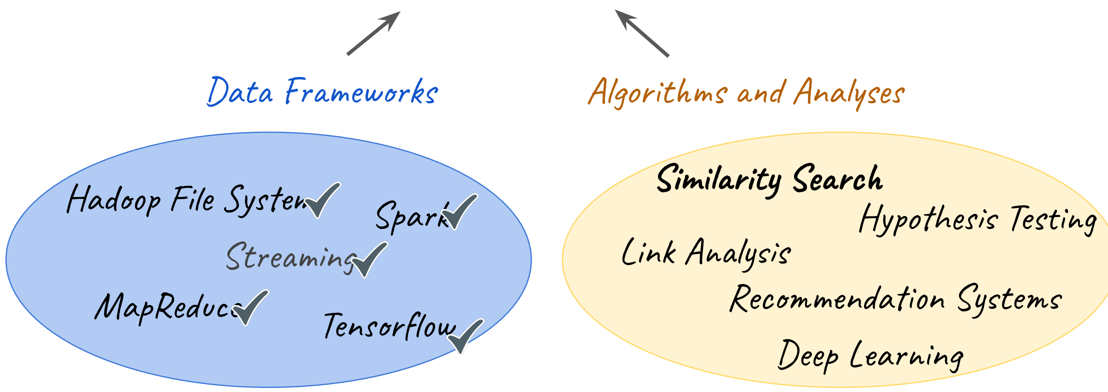
CSE545 - Spring 2020
Stony Brook University

H. Andrew Schwartz

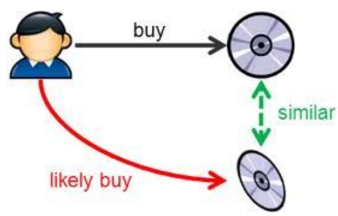
$A \cap B$

Big Data Analytics, The Class

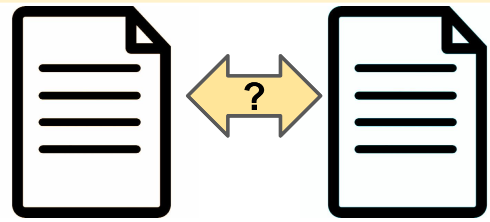
Goal: Generalizations
A *model* or *summarization* of the data.

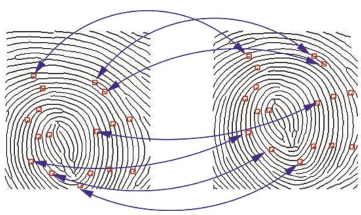


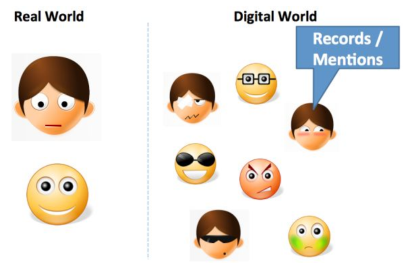
Finding Similar Items



(http://blog.soton.ac.uk/hive/2012/05/10/recommendation-system-of-hive/)







(http://www.datacommunitydc.org/blog/2013/08/entity-resolution-for-big-data)

- There are many applications where we desire finding similar items to a given example.
- For example:
    - Document Similarity:
        - Mirrored web-pages
        - Plagiarism; Similar News
    - Recommendations:
        - Online purchases
        - Movie ratings
    - Entity Resolution: matching one instance of a person with another
    - Fingerprint Matching: finding the most likely matches in a larg dataset of matches.

Finding Similar Items: Topics

- Shingling
- Minhashing
- Locality-sensitive hashing
- Distance Metrics

We will cover the following methods for finding similar items.

The first 3 make up a pipeline of techniques, culminating in LSH for rapidly matching items over a large search space. Similarity in these cases all comes down to a jaccard set similarity.

Distance metrics introduces a different set of common approaches to assessing similarity between items, assuming one has some features (quantities describing describing them).

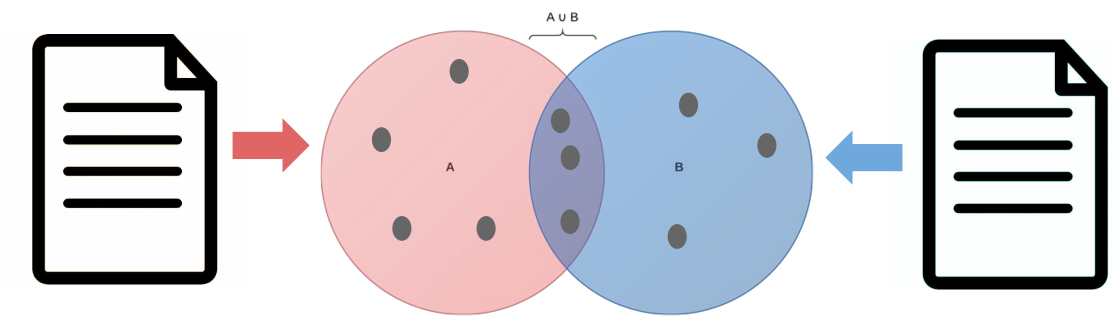
Document Similarity

Challenge: How to represent the document in a way that can be efficiently encoded and compared?

The first challenge for efficiently searching for similar items is simply how to represent an item.

Shingles

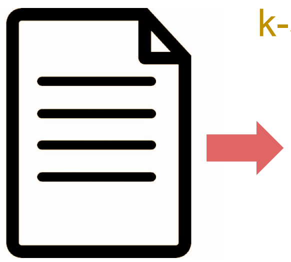
Goal: Convert documents to sets



If we can represent an item (a document in this case) simply as a set, a very simple representation, then we can look at overlap in sets as similarity.

Shingles

Goal: Convert documents to sets



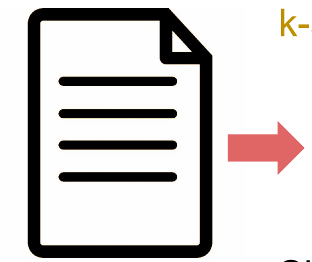
k-shingles (aka "character n-grams")
- sequence of *k* characters

E.g. k=2 doc="abcdabd"
singles(doc, 2) = {ab, bc, cd, da, bd}

A very easy way to get sets from all documents and many other file types is simply shingles. Take sequences of k characters in a row.

Shingles

Goal: Convert documents to sets



k-shingles (aka “character n-grams”)
- sequence of *k* characters

E.g. *k*=2 doc=”abcdabd”
singles(doc, 2) = {ab, bc, cd, da, bd}

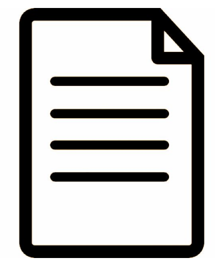
- Similar documents have many common shingles
- Changing words or order has minimal effect.
- In practice use 5 < k < 10

We would expect similar document to have similar shingles.

In practice using shingles of size 5 to 10 is more ideal to make it less likely to randomly match shingles between 2 documents.

Shingles

Goal: Convert documents to sets



Large enough that any given shingle appearing a document is highly unlikely
(e.g. < .1% chance)

Can hash large shingles to smaller
(e.g. 9-shingles into 4 bytes)

Can also use words (aka n-grams).

- Similar documents have many common shingles
- Changing w ds or order has minimal effect.
- **In practice use 5 < k < 10**

Generally, we want elements in our sets (i.e. shingles) to match with about 1 in 1000 probability.

The larger generally the better for this purpose and we can even hash shingles to reduce their size a bit.

Shingles

Problem: Even if hashing, sets of shingles are large
(e.g. 4 bytes => 4x the size of the document).

However, such a representation, even when hashed, still enlarges the document rather than reduces it and we want to be able to search over millions to billions of these quickly. If you consider a character as a byte then even hashing 9grams (9 bytes) down to 4 bytes has the potential to make a document 4x its original size.

Minhashing

Goal: Convert sets to shorter ids, signatures

While shingles gives us a simple way to turn a document into a set, we need a way to make that set representation smaller. This is where minhashing comes in.

Minhashing

Goal: Convert sets to shorter ids, “signatures”

Characteristic Matrix, X:

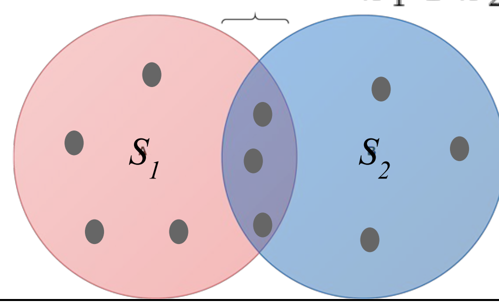
$Element$	S_1	S_2	S_3	S_4	....
a	1	0	0	1	
b	0	0	1	0	
c	0	1	0	1	
d	1	0	1	1	
e	0	0	1	0	

(Leskovec at al., 2014; http://www.mmds.org/)

often very sparse! (lots of zeros)

Jaccard Similarity:

$$sim(S_1, S_2) = \frac{S_1 \cap S_2}{S_1 \cup S_2}$$



Let’s go ahead and define how we will compute similarity based on a set:
We can use Jaccard Similarity: The amount of overlap divided by the total elements of the union.
In this way, similarity is basically a percentage of the total number of elements that are shared.
It has intuitive properties such as if one document is larger and thus has more elements in its set that will have the effect of shrinking the amount of similarity unless they other document contains many of the same elements.

We will call “characteristic matrix” the actual type of data structure we use to represent these sets. It’s simply a binary matrix with sets (i.e. documents) as columns and shingles (i.e. elements) as rows.
In practice, the characteristic matrix will be very sparse -- remember we want about a 1 in 1000 chance of a particular shingle to appear.

Minhashing

Characteristic Matrix:

	S_1	S_2
ab	1	1
bc	0	1
de	1	0
ah	1	1
ha	0	0
ed	1	1
ca	0	1

Jaccard Similarity:

$$sim(S_1, S_2) = \frac{S_1 \cap S_2}{S_1 \cup S_2}$$

Latex equation: sim(S_1, S_2) = \frac{S_1 \cap S_2 }{S_1 \cup S_2}

Let’s start to work with an example charactertistic matrix of two documents.

What would be the similarity?

Minhashing

Characteristic Matrix:

	S_1	S_2	
ab	1	1	* *
bc	0	1	*
de	1	0	*
ah	1	1	**
ha	0	0	
ed	1	1	**
ca	0	1	*

Jaccard Similarity:

$$sim(S_1, S_2) = \frac{S_1 \cap S_2}{S_1 \cup S_2}$$

sim(S_1, S_2) = \frac{S_1 \cap S_2 }{S_1 \cup S_2}

One way to quick algorithm to calculate is simply to sum the rows.

Minhashing

Characteristic Matrix:

	S_1	S_2	
ab	1	1	* *
bc	0	1	*
de	1	0	*
ah	1	1	**
ha	0	0	
ed	1	1	**
ca	0	1	*

Jaccard Similarity:

$$sim(S_1, S_2) = \frac{S_1 \cap S_2}{S_1 \cup S_2}$$

$sim(S_{1,} S_2) = 3 / 6$

\# both have / # at least one has

and divide the number of 2s by the number of 1s. (i.e. 3/6 in this case)

Notice we only care about when one of them is 1.

Minhashing

Problem: Even if hashing shingle contents,
sets of shingles are large

e.g. 4 byte integer per shingle: assume all unique shingles,
=> 4x the size of the document
(since there are as many shingles as characters and 1byte per char).

So, keeping Jaccard Similarity in mind, how do we get this characteristic matrix smaller?

Minhashing

Goal: Convert sets to shorter ids, "signatures"

Characteristic Matrix: X

	S_1	S_2	S_3	S_4
ab	1	0	1	0
bc	1	0	0	1
de	0	1	0	1
ah	0	1	0	1
ha	0	1	0	1
ed	1	0	1	0
ca	1	0	1	0

(Leskovec at al., 2014; http://www.mmds.org/)

We want to create a shorter id a "signature" from the larger characteristic matrix

Minhashing

Goal: Convert sets to shorter ids, "signatures"

Characteristic Matrix: X

	S_1	S_2	S_3	S_4
ab	1	0	1	0
bc	1	0	0	1
de	0	1	0	1
ah	0	1	0	1
ha	0	1	0	1
ed	1	0	1	0
ca	1	0	1	0

(Leskovec at al., 2014; http://www.mmds.org/)

Approximate Approach:

1) Instead of keeping whole characteristic matrix, just keep first row where 1 is encountered.

2) Shuffle and repeat to get a "signature" for each set.

Well let's take an extreme approach. What if we only represented the Set by a single integer?

We could just keep the row number where the first element was non-zero.

Minhashing

Goal: Convert sets to shorter ids, "signatures"

Characteristic Matrix: X

	1	3	1	2
	S_1	S_2	S_3	S_4
ab	1	0	1	0
bc	1	0	0	1
de	0	1	0	1
ah	0	1	0	1
ha	0	1	0	1
ed	1	0	1	0
ca	1	0	1	0

(Leskovec at al., 2014; http://www.mmds.org/)

Approximate Approach:

1) Instead of keeping whole characteristic matrix, just keep first row where 1 is encountered.

2) Shuffle and repeat to get a "signature" for each set.

Here is what we would get: set 1 and set 3 woudl actually get the same integer, while 2 and 4 would each have a different.

Well set 1 and set 3 do happen to be quite similar: Their Sim is ¾

In fact, if you think about it, given a random ordering of the rows, what is the probability that both of their first non-zero row happens to be the same? ¾ in 3 of the 4 possible rows that have at least a 1 (ab, bv, ed, and ca) only 1 of them being first wouldn't be a match (bc).

Minhashing

Goal: Convert sets to shorter ids, "signatures"

Characteristic Matrix: X

	1	3	1	2
	S_1	S_2	S_3	S_4
ab	1	0	1	0
bc	1	0	0	1
de	0	1	0	1
ah	0	1	0	1
ha	0	1	0	1
ed	1	0	1	0
ca	1	0	1	0

(Leskovec at al., 2014; http://www.mmds.org/)

Approximate Approach:

1) Instead of keeping whole characteristic matrix, just keep first row where 1 is encountered.

2) Shuffle and repeat to get a "signature".

	2	1	2	1
	S_1	S_2	S_3	S_4
ah	0	1	0	1
ca	1	0	1	0
ed	1	0	1	0
de	0	1	0	1
ab	1	0	1	0
bc	1	0	0	1

...

In reality of course, a single integer is not going to be enough but we can repeat this a few times. Here's an example after we shuffle.
Now both pairs S1 - S3 AND S2 S4 match. S2 and S4 also have a sim of ¾ . If we just asked at this point how much did these 2-integer signatures match, we'd find 100% for S1-S3 and 50% for S2-S4… one overestimates; one underestimates…
This can continue in order to make a more and more accurate signature that matches with the same probability as the Jaccard Similarity.

Minhashing

Goal: Convert sets to shorter ids, "signatures"

Characteristic Matrix: X

1 3 1 2

	S_1	S_2	S_3	S_4
ab	1	0	1	0
bc	1	0	0	1
de	0	1	0	1
ah	0	1	0	1
ha	0	1	0	1
ed	1	0	1	0
ca	1	0	1	0

(Leskovec at al., 2014; http://www.mmds.org/)

Approximate Approach:

1) Instead of keeping whole characteristic matrix, just keep first row where 1 is encountered.

2) Shuffle and repeat to get a "signature".

2 1 2 1

	S_1	S_2	S_3	S_4
ah	0	1	0	1
ca	1	0	1	0
ed	1	0	1	0
de	0	1	0	1
ab	1	0	1	0
bc	1	0	0	1

...

signatures

S_1	S_2	S_3	S_4
1	3	1	2
2	1	2	1
...	...	...	...

Here is what the signatures look like so far.

We're going to try to produce a "signature matrix" as the output of minhashing, where each column is a signature.

Minhashing

Goal: Convert sets to shorter ids, "signatures"

Characteristic Matrix: X

	S_1	S_2	S_3	S_4
ab	1	0	1	0
bc	1	0	0	1
de	0	1	0	1
ah	0	1	0	1
ha	0	1	0	1
ed	1	0	1	0
ca	1	0	1	0

(Leskovec at al., 2014; http://www.mmds.org/)

Approximate Approach:

1) Instead of keeping whole characteristic matrix, just keep first row where 1 is encountered.

2) Shuffle and repeat to get a "signature" for each set.

Idea: We don't need to actually shuffle we can just use hash functions.

One downside of how we've discuss this is the time it woudl take to keep reshuffling rows, but there's really no need to do that.
Shuffle is just the conceptual way to think about this when in fact we can use hash functions to give us a random order of rows to look at.

Minhashing

Characteristic Matrix:

	S_1	S_2	S_3	S_4
ab	1	0	1	0
bc	1	0	0	1
de	0	1	0	1
ah	0	1	0	1
ha	0	1	0	1
ed	1	0	1	0
ca	1	0	1	0

(Leskovec at al., 2014; http://www.mmds.org/)

Minhash function: h

- Based on permutation of rows in the characteristic matrix, h maps sets to first row where set appears.

Minhashing

Characteristic Matrix:

	S_1	S_2	S_3	S_4
ab	1	0	1	0
bc	1	0	0	1
de	0	1	0	1
ah	0	1	0	1
ha	0	1	0	1
ed	1	0	1	0
ca	1	0	1	0

permuted order
1 ha
2 ed
3 ab
4 bc
5 ca
6 ah
7 de

Minhash function: h

- Based on permutation of rows in the characteristic matrix, h maps sets to first row where set appears.

(Leskovec at al., 2014; http://www.mmds.org/)

sim(S_1, S_2) = \frac{S_1 \cap S_2 }{S_1 \cup S_2}

Minhashing

Characteristic Matrix:

		S_1	S_2	S_3	S_4
3	ab	1	0	1	0
4	bc	1	0	0	1
7	de	0	1	0	1
6	ah	0	1	0	1
1	ha	0	1	0	1
2	ed	1	0	1	0
5	ca	1	0	1	0

permuted order
1 ha
2 ed
3 ab
4 bc
5 ca
6 ah
7 de

Minhash function: h

- Based on permutation of rows in the characteristic matrix, h maps sets to first row where set appears.

(Leskovec at al., 2014; http://www.mmds.org/)

sim(S_1, S_2) = \frac{S_1 \cap S_2 }{S_1 \cup S_2}

Minhashing

Characteristic Matrix:

		S_1	S_2	S_3	S_4
3	ab	1	0	1	0
4	bc	1	0	0	1
7	de	0	1	0	1
6	ah	0	1	0	1
1	ha	0	1	0	1
2	ed	1	0	1	0
5	ca	1	0	1	0

permuted order
1 ha
2 ed
3 ab
4 bc
5 ca
6 ah
7 de

(Leskovec at al., 2014; http://www.mmds.org/)

Minhash function: h

- Based on permutation of rows in the characteristic matrix, h maps sets to first row where set appears.

$h(S_1)$ = ed #permuted row 2
$h(S_2)$ = ha #permuted row 1
$h(S_3)$ =

sim(S_1, S_2) = \frac{S_1 \cap S_2 }{S_1 \cup S_2}

Minhashing

Characteristic Matrix:

		S_1	S_2	S_3	S_4
3	ab	1	0	1	0
4	bc	1	0	0	1
7	de	0	1	0	1
6	ah	0	1	0	1
1	ha	0	1	0	1
2	ed	1	0	1	0
5	ca	1	0	1	0

permuted order
1 ha
2 ed
3 ab
4 bc
5 ca
6 ah
7 de

(Leskovec at al., 2014; http://www.mmds.org/)

Minhash function: h

- Based on permutation of rows in the characteristic matrix, h maps sets to first row where set appears.

$h(S_1)$ = ed #permuted row 2
$h(S_2)$ = ha #permuted row 1
$h(S_3)$ = ed #permuted row 2
$h(S_4)$ =

sim(S_1, S_2) = \frac{S_1 \cap S_2 }{S_1 \cup S_2}

Minhashing

Characteristic Matrix:

		S_1	S_2	S_3	S_4
3	ab	1	0	1	0
4	bc	1	0	0	1
7	de	0	1	0	1
6	ah	0	1	0	1
1	ha	0	1	0	1
2	ed	1	0	1	0
5	ca	1	0	1	0

permuted order
1 ha
2 ed
3 ab
4 bc
5 ca
6 ah
7 de

(Leskovec at al., 2014; http://www.mmds.org/)

Minhash function: h

- Based on permutation of rows in the characteristic matrix, h maps sets to first row where set appears.

$h(S_1)$ = ed #permuted row 2
$h(S_2)$ = ha #permuted row 1
$h(S_3)$ = ed #permuted row 2
$h(S_4)$ = ha #permuted row 1

sim(S_1, S_2) = \frac{S_1 \cap S_2 }{S_1 \cup S_2}

Minhashing

Characteristic Matrix:

		S_1	S_2	S_3	S_4
3	ab	1	0	1	0
4	bc	1	0	0	1
7	de	0	1	0	1
6	ah	0	1	0	1
1	ha	0	1	0	1
2	ed	1	0	1	0
5	ca	1	0	1	0

(Leskovec at al., 2014; http://www.mmds.org/)

Minhash function: h

- Based on permutation of rows in the characteristic matrix, h maps sets to rows.

Signature matrix: *M*

- Record first row where each set had a 1 in the given permutation

	S_1	S_2	S_3	S_4
h_1	2	1	2	1

$h_1(S_1)$ = ed #permuted row 2
$h_1(S_2)$ = ha #permuted row 1
$h_1(S_3)$ = ed #permuted row 2
$h_1(S_4)$ = ha #permuted row 1

sim(S_1, S_2) = \frac{S_1 \cap S_2 }{S_1 \cup S_2}

Minhashing

Characteristic Matrix:

		S_1	S_2	S_3	S_4
3	ab	1	0	1	0
4	bc	1	0	0	1
7	de	0	1	0	1
6	ah	0	1	0	1
1	ha	0	1	0	1
$\boxed{2}$	ed	$\boxed{1}$	0	1	0
5	ca	1	0	1	0

(Leskovec at al., 2014; http://www.mmds.org/)

Minhash function: h

- Based on permutation of rows in the characteristic matrix, h maps sets to rows.

Signature matrix: M

- Record first row where each set had a 1 in the given permutation

	S_1	S_2	S_3	S_4
h_1	$\boxed{2}$	1	2	1

$\boxed{h_1(S_1) = \text{ed} \ \ \#\text{permuted row}}$

2

$h_1(S_2) = \text{ha} \ \ \#\text{permuted row}$

1

$h_1(S_3) = \text{ed} \ \ \#\text{permuted row}$

sim(S_1, S_2) = \frac{S_1 \cap S_2 }{S_1 \cup S_2}

Minhashing

Characteristic Matrix:

		S_1	S_2	S_3	S_4
3	ab	1	0	1	0
4	bc	1	0	0	1
7	de	0	1	0	1
6	ah	0	1	0	1
1	ha	0	1	0	1
2	ed	1	0	1	0
5	ca	1	0	1	0

(Leskovec at al., 2014; http://www.mmds.org/)

Minhash function: h

- Based on permutation of rows in the characteristic matrix, h maps sets to rows.

Signature matrix: M

- Record first row where each set had a 1 in the given permutation

	S_1	S_2	S_3	S_4
h_1	2	1	2	1

$h_1(S_1)$ = ed #permuted row 2

$h_1(S_2)$ = ha #permuted row 1

$h_1(S_3)$ = ed #permuted row

sim(S_1, S_2) = \frac{S_1 \cap S_2 }{S_1 \cup S_2}

Minhashing

Characteristic Matrix:

			S_1	S_2	S_3	S_4
4	3	ab	1	0	1	0
2	4	bc	1	0	0	1
1	7	de	0	1	0	1
3	6	ah	0	1	0	1
6	1	ha	0	1	0	1
7	2	ed	1	0	1	0
5	5	ca	1	0	1	0

(Leskovec at al., 2014; http://www.mmds.org/)

Minhash function: h

- Based on permutation of rows in the characteristic matrix, h maps sets to rows.

Signature matrix: M

- Record first row where each set had a 1 in the given permutation

	S_1	S_2	S_3	S_4
h_1	2	1	2	1
h_2				

sim(S_1, S_2) = \frac{S_1 \cap S_2 }{S_1 \cup S_2}

Minhashing

Characteristic Matrix:

			S_1	S_2	S_3	S_4
4	3	ab	1	0	1	0
2	4	bc	1	0	0	1
1	7	de	0	1	0	1
3	6	ah	0	1	0	1
6	1	ha	0	1	0	1
7	2	ed	1	0	1	0
5	5	ca	1	0	1	0

(Leskovec at al., 2014; http://www.mmds.org/)

Minhash function: h

- Based on permutation of rows in the characteristic matrix, h maps sets to rows.

Signature matrix: *M*

- Record first row where each set had a 1 in the given permutation

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

sim(S_1, S_2) = \frac{S_1 \cap S_2 }{S_1 \cup S_2}

Minhashing

Characteristic Matrix:

				S_1	S_2	S_3	S_4
1	4	3	ab	1	0	1	0
3	2	4	bc	1	0	0	1
7	1	7	de	0	1	0	1
6	3	6	ah	0	1	0	1
2	6	1	ha	0	1	0	1
5	7	2	ed	1	0	1	0
4	5	5	ca	1	0	1	0

(Leskovec at al., 2014; http://www.mmds.org/)

Minhash function: h

- Based on permutation of rows in the characteristic matrix, h maps sets to rows.

Signature matrix: M

- Record first row where each set had a 1 in the given permutation

	S_1	S_2	S_3	S_4
h_1	2	1	2	1
h_2	2	1	4	1
h_3				

sim(S_1, S_2) = \frac{S_1 \cap S_2 }{S_1 \cup S_2}

Minhashing

Characteristic Matrix:

				S_1	S_2	S_3	S_4
1	4	3	ab	1	0	1	0
3	2	4	bc	1	0	0	1
7	1	7	de	0	1	0	1
6	3	6	ah	0	1	0	1
2	6	1	ha	0	1	0	1
5	7	2	ed	1	0	1	0
4	5	5	ca	1	0	1	0

(Leskovec at al., 2014; http://www.mmds.org/)

Minhash function: h

- Based on permutation of rows in the characteristic matrix, h maps sets to rows.

Signature matrix: M

- Record first row where each set had a 1 in the given permutation

	S_1	S_2	S_3	S_4
h_1	2	1	2	1
h_2	2	1	4	1
h_3	1	2	1	2

sim(S_1, S_2) = \frac{S_1 \cap S_2 }{S_1 \cup S_2}

Minhashing

Characteristic Matrix:

				S_1	S_2	S_3	S_4
1	4	3	ab	1	0	1	0
3	2	4	bc	1	0	0	1
7	1	7	de	0	1	0	1
6	3	6	ah	0	1	0	1
2	6	1	ha	0	1	0	1
5	7	2	ed	1	0	1	0
4	5	5	ca	1	0	1	0

(Leskovec at al., 2014; http://www.mmds.org/)

Minhash function: h

- Based on permutation of rows in the characteristic matrix, h maps sets to rows.

Signature matrix: M

- Record first row where each set had a 1 in the given permutation

	S_1	S_2	S_3	S_4
h_1	2	1	2	1
h_2	2	1	4	1
h_3	1	2	1	2
...				
...				

sim(S_1, S_2) = \frac{S_1 \cap S_2 }{S_1 \cup S_2}

Minhashing

Property of signature matrix:
The probability for any h_i (i.e. any row), that $h_i(S_1) = h_i(S_2)$ is the same as Sim(S_1, S_2)

Characteristic Matrix:

				S_1	S_2	S_3	S_4
1	4	3	ab	1	0	1	0
3	2	4	bc	1	0	0	1
7	1	7	de	0	1	0	1
6	3	6	ah	0	1	0	1
2	6	1	ha	0	1	0	1
5	7	2	ed	1	0	1	0
4	5	5	ca	1	0	1	0

	S_1	S_2	S_3	S_4
h_1	2	1	2	1
h_2	2	1	4	1
h_3	1	2	1	2
...				
...				

(Leskovec at al., 2014; http://www.mmds.org/)

sim(S_1, S_2) = \frac{S_1 \cap S_2 }{S_1 \cup S_2}

Minhashing

Property of signature matrix:
The probability for any h_i (i.e. any row), that
$h_i(S_1) = h_i(S_2)$ is the same as Sim(S_1, S_2)

Thus, similarity of signatures S_1, S_2 is the fraction of minhash functions (i.e. rows) in which they agree.

Characteristic Matrix:

				S_1	S_2	S_3	S_4
1	4	3	ab	1	0	1	0
3	2	4	bc	1	0	0	1
7	1	7	de	0	1	0	1
6	3	6	ah	0	1	0	1
2	6	1	ha	0	1	0	1
5	7	2	ed	1	0	1	0
4	5	5	ca	1	0	1	0

	S_1	S_2	S_3	S_4
h_1	2	1	2	1
h_2	2	1	4	1
h_3	1	2	1	2
...				
...				

(Leskovec at al., 2014; http://www.mmds.org/)

sim(S_1, S_2) = \frac{S_1 \cap S_2 }{S_1 \cup S_2}

Minhashing

Characteristic Matrix:

				S_1	S_2	S_3	S_4
1	4	3	ab	1	0		0
3	2						
7							
6	3	6	an				
2	6	1	ha	0	1	0	1
5	7	2	ed	1	0	1	0
4	5	5	ca	1	0	1	0

Property of signature matrix:
The probability for any h_i (i.e. any row), that $h_i(S_1) = h_i(S_2)$ is the same as Sim(S_1, S_2)

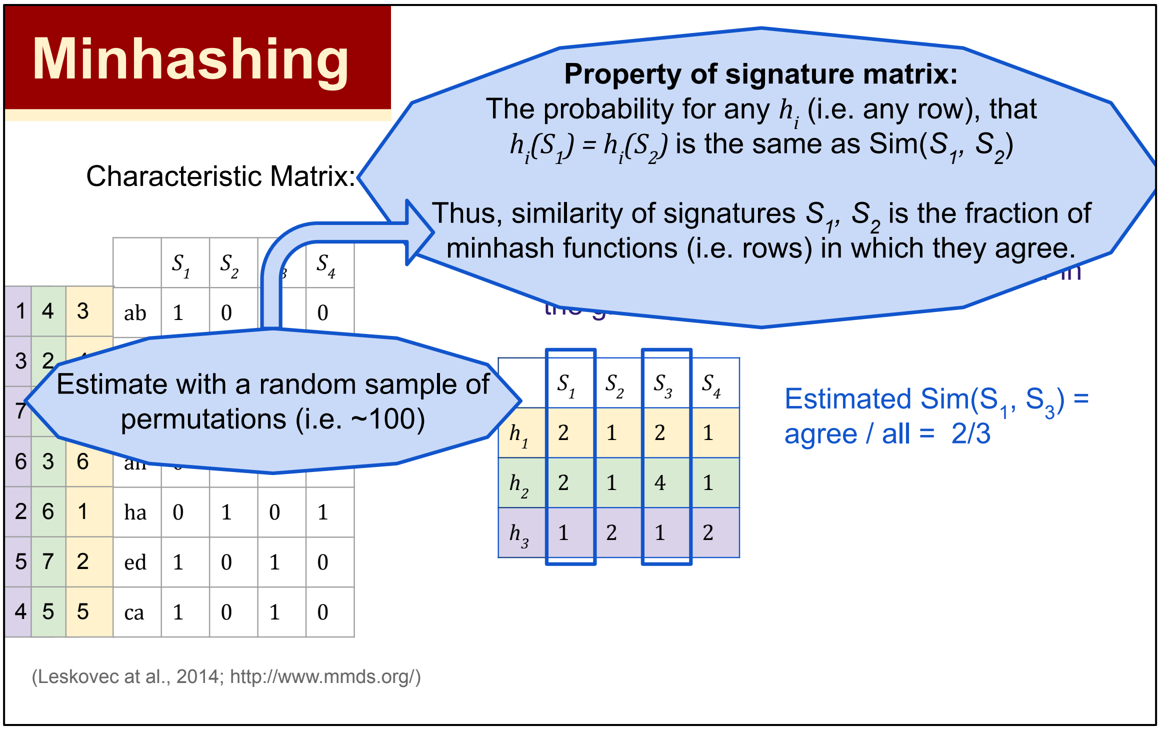
Thus, similarity of signatures S_1, S_2 is the fraction of minhash functions (i.e. rows) in which they agree.

Estimate with a random sample of permutations (i.e. ~100)

	S_1	S_2	S_3	S_4
h_1	2	1	2	1
h_2	2	1	4	1
h_3	1	2	1	2
...				
...				

(Leskovec at al., 2014; http://www.mmds.org/)

sim(S_1, S_2) = \frac{S_1 \cap S_2 }{S_1 \cup S_2}



sim(S_1, S_2) = \frac{S_1 \cap S_2 }{S_1 \cup S_2}

Minhashing

Property of signature matrix:
The probability for any h_i (i.e. any row), that $h_i(S_1) = h_i(S_2)$ is the same as Sim(S_1, S_2)

Thus, similarity of signatures S_1, S_2 is the fraction of minhash functions (i.e. rows) in which they agree.

Characteristic Matrix:

				S_1	S_2	S_3	S_4
1	4	3	ab	1	0	1	0
3	2	4	bc	1	0	0	1
7	1	7	de	0	1	0	1
6	3	6	ah	0	1	0	1
2	6	1	ha	0	1	0	1
5	7	2	ed	1	0	1	0
4	5	5	ca	1	0	1	0

	S_1	S_2	S_3	S_4
h_1	2	1	2	1
h_2	2	1	4	1
h_3	1	2	1	2

Estimated Sim(S_1, S_3) = agree / all = 2/3

Real Sim(S_1, S_3) = Type a / (a + b + c) = 3/4

(Leskovec at al., 2014; http://www.mmds.org/)

sim(S_1, S_2) = \frac{S_1 \cap S_2 }{S_1 \cup S_2}

Minhashing

Property of signature matrix:
The probability for any h_i (i.e. any row), that $h_i(S_1) = h_i(S_2)$ is the same as Sim(S_1, S_2)

Thus, similarity of signatures S_1, S_2 is the fraction of minhash functions (i.e. rows) in which they agree.

Characteristic Matrix:

				S_1	S_2	S_3	S_4
1	4	3	ab	1	0	1	0
3	2	4	bc	1	0	0	1
7	1	7	de	0	1	0	1
6	3	6	ah	0	1	0	1
2	6	1	ha	0	1	0	1
5	7	2	ed	1	0	1	0
4	5	5	ca	1	0	1	0

	S_1	S_2	S_3	S_4
h_1	2	1	2	1
h_2	2	1	4	1
h_3	1	2	1	2

Estimated Sim(S_1, S_3) = agree / all = 2/3

Real Sim(S_1, S_3) = Type a / (a + b + c) = 3/4

Try Sim(S_2, S_4) and Sim(S_1, S_2)

(Leskovec at al., 2014; http://www.mmds.org/)

sim(S_1, S_2) = \frac{S_1 \cap S_2 }{S_1 \cup S_2}

Minhashing

Error Bound?

Characteristic Matrix:

				S_1	S_2	S_3	S_4
1	4	3	ab	1	0	1	0
3	2	4	bc	1	0	0	1
7	1	7	de	0	1	0	1
6	3	6	ah	0	1	0	1
2	6	1	ha	0	1	0	1
5	7	2	ed	1	0	1	0
4	5	5	ca	1	0	1	0

(Leskovec at al., 2014; http://www.mmds.org/)

	S_1	S_2	S_3	S_4
h_1	2	1	2	1
h_2	2	1	4	1
h_3	1	2	1	2

Estimated Sim(S_1, S_3) = agree / all = 2/3

Real Sim(S_1, S_3) = Type a / (a + b + c) = 3/4

Try Sim(S_2, S_4) and Sim(S_1, S_2)

sim(S_1, S_2) = \frac{S_1 \cap S_2 }{S_1 \cup S_2}

Minhashing

Characteristic Matrix:

				S_1	S_2	S_3	S_4
1	4	3	ab	1	0	1	0
3	2	4	bc	1	0	0	1
7	1	7	de	0	1	0	1
6	3	6	ah	0	1	0	1
2	6	1	ha	0	1	0	1
5	7	2	ed	1	0	1	0
4	5	5	ca	1	0	1	0

(Leskovec at al., 2014; http://www.mmds.org/)

Error Bound?

Expect error: **O(*1*/√*k*)** (k hashes)

Why? Each row is a random observation of 1 or 0 (match or not) with *P(match=1) = Sim(S1, S2).*

	S_1	S_2	S_3	S_4
h_1	2	1	2	1
h_2	2	1	4	1
h_3	1	2	1	2

Estimated Sim(S_1, S_3) = agree / all = 2/3

Real Sim(S_1, S_3) = Type a / (a + b + c) = 3/4

Try Sim(S_2, S_4) and Sim(S_1, S_2)

sim(S_1, S_2) = \frac{S_1 \cap S_2 }{S_1 \cup S_2}

Minhashing

Characteristic Matrix:

				S_1	S_2	S_3	S_4
1	4	3	ab	1	0	1	0
3	2	4	bc	1	0	0	1
7	1	7	de	0	1	0	1
6	3	6	ah	0	1	0	1
2	6	1	ha	0	1	0	1
5	7	2	ed	1	0	1	0
4	5	5	ca	1	0	1	0

(Leskovec at al., 2014; http://www.mmds.org/)

Error Bound?

Expect error: $O(1/\sqrt{k})$ (k hashes)

Why? Each row is a random observation of 1 or 0 (match or not) with *P(match=1) = Sim(S1, S2).*

N = *k* observations

Standard deviation(*std*)? < 1 (worst case is 0.5)

	S_1	S_2	S_3	S_4
h_1	2	1	2	1
h_2	2	1	4	1
h_3	1	2	1	2

Estimated Sim(S_1, S_3) = agree / all = 2/3

Real Sim(S_1, S_3) = Type a / (a + b + c) = 3/4

Try Sim(S_2, S_4) and Sim(S_1, S_2)

sim(S_1, S_2) = \frac{S_1 \cap S_2 }{S_1 \cup S_2}

Minhashing

Characteristic Matrix:

				S_1	S_2	S_3	S_4
1	4	3	ab	1	0	1	0
3	2	4	bc	1	0	0	1
7	1	7	de	0	1	0	1
6	3	6	ah	0	1	0	1
2	6	1	ha	0	1	0	1
5	7	2	ed	1	0	1	0
4	5	5	ca	1	0	1	0

(Leskovec at al., 2014; http://www.mmds.org/)

Error Bound?

Expect error: **O(*1*/√*k*)** (k hashes)

Why? Each row is a random observation of 1 or 0 (match or not) with *P(match=1) = Sim(S1, S2).*

N = *k* observations

Standard deviation(*std*)? < 1 (worst case is 0.5)

Standard Error of Mean = *std*/√N

	S_1	S_2	S_3	S_4
h_1	2	1	2	1
h_2	2	1	4	1
h_3	1	2	1	2

Estimated Sim(S_1, S_3) = agree / all = 2/3

Real Sim(S_1, S_3) = Type a / (a + b + c) = 3/4

Try Sim(S_2, S_4) and Sim(S_1, S_2)

sim(S_1, S_2) = \frac{S_1 \cap S_2 }{S_1 \cup S_2}

Minhashing

In Practice

Problem:

- Can’t reasonably do permutations (huge space)
- Can’t randomly grab rows according to an order (random disk seeks = slow!)

Minhashing

In Practice

Problem:

- Can’t reasonably do permutations (huge space)
- Can’t randomly grab rows according to an order
(random disk seeks = slow!)

Solution: Use “random” hash functions.

- Setup:
  - Pick ~100 hash functions, hashes
  - Store M[i][s] = a potential minimum $h_i(r)$
    #initialized to infinity (num hashs x num sets)

Minhashing

Solution: Use “random” hash functions.

Setup:

```
hashes = [getHfunc(i) for i in rand(1, num=100)]
                                  #100 hash functions, seeded random
for i in hashes: for s in sets:
   M[i][s] = np.inf #represents a potential minimum h_i(r) ; initially infinity
```

Algorithm (“efficient minhashing”):

```
for r in rows of cm: #cm is characteristic matrix
 compute h_i(r) for all i in hashes #precompute 100 values
 for each set s in sets:
   if cm[r][s] == 1:
     for i in hashes: #check which hash produces smallest value
       if h_i(r) < M[i][s]: M[i][s] = h_i(r)
```

Minhashing

Problem: Even if hashing, sets of shingles are large (e.g. 4 bytes => 4x the size of the document).

Come up with example?

Minhashing

Problem: Even if hashing, sets of shingles are large (e.g. 4 bytes => 4x the size of the document).

New Problem: Even if the size of signatures are small, it can be computationally expensive to find similar pairs.

E.g. 1m documents; 1,000,000 choose 2 = 500,000,000,000 pairs!

Come up with example?

Minhashing

Problem: Even if hashing, sets of shingles are large (e.g. 4 bytes => 4x the size of the document).

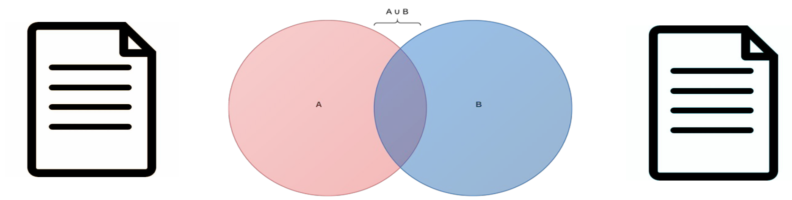
New Problem: Even if the size of signatures are small, it can be computationally expensive to find similar pairs.

E.g. 1m documents; 1,000,000 choose 2 = 500,000,000,000 pairs!

(1m documents isn't even "big data")

Come up with example?

Document Similarity



Duplicate web pages (useful for ranking

Plagiarism

Cluster News Articles

Anything similar to documents: movie/music/art tastes, product characteristics

Locality-Sensitive Hashing

Goal: find pairs of minhashes ***likely*** to be similar (in order to then test more precisely for similarity).

Candidate pairs: pairs of elements to be evaluated for similarity.

Locality-Sensitive Hashing

Goal: find pairs of minhashes ***likely*** to be similar (in order to then test more precisely for similarity).

Candidate pairs: pairs of elements to be evaluated for similarity.

If we wanted the similarity for all pairs of documents, could anything be done?

Locality-Sensitive Hashing

Goal: find pairs of minhashes likely to be similar (in order to then test more precisely for similarity).

Candidate pairs: pairs of elements to be evaluated for similarity.

Approach: Hash multiple times over subsets of data: similar items are likely in the same bucket once.

Locality-Sensitive Hashing

Goal: find pairs of minhashes likely to be similar (in order to then test more precisely for similarity).

Candidate pairs: pairs of elements to be evaluated for similarity.

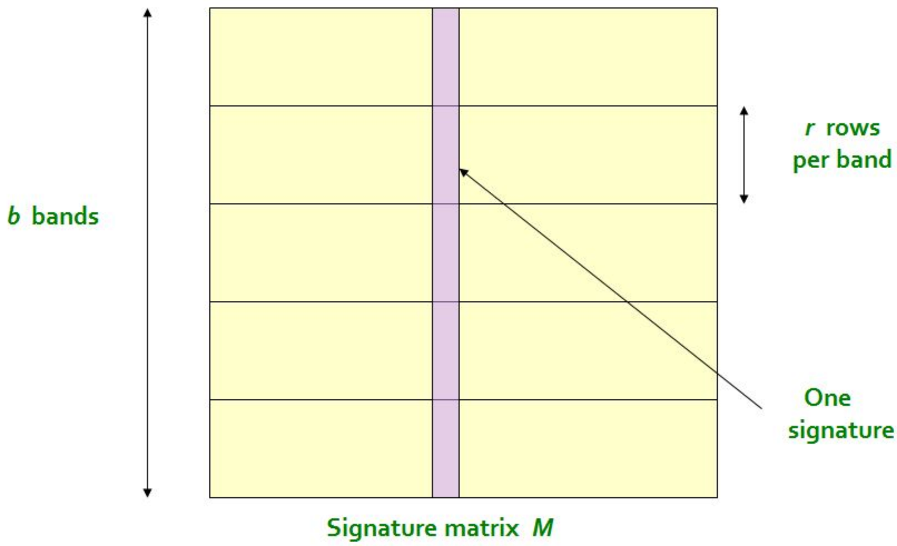
Approach: Hash multiple times over subsets of data: similar items are likely in the same bucket once.

Approach from MinHash: Hash columns of signature matrix

⇒ Candidate pairs end up in the same bucket.

Locality-Sensitive Hashing

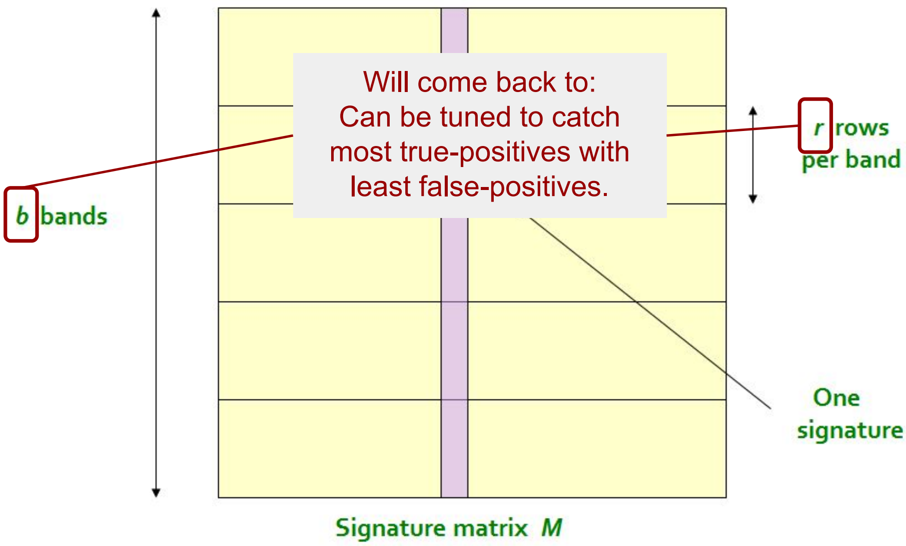
Step 1: Divide signature matrix into *b* bands



(Leskovec at al., 2014; http://www.mmds.org/)

Step 1: Divide into *b* bands

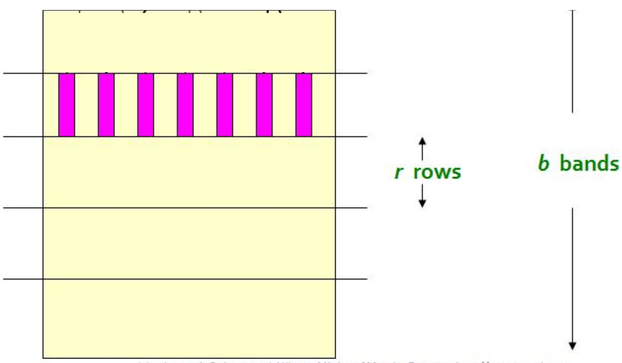
Locality-Sensitive Hashing



(Leskovec at al., 2014; http://www.mmds.org/)

Locality-Sensitive Hashing

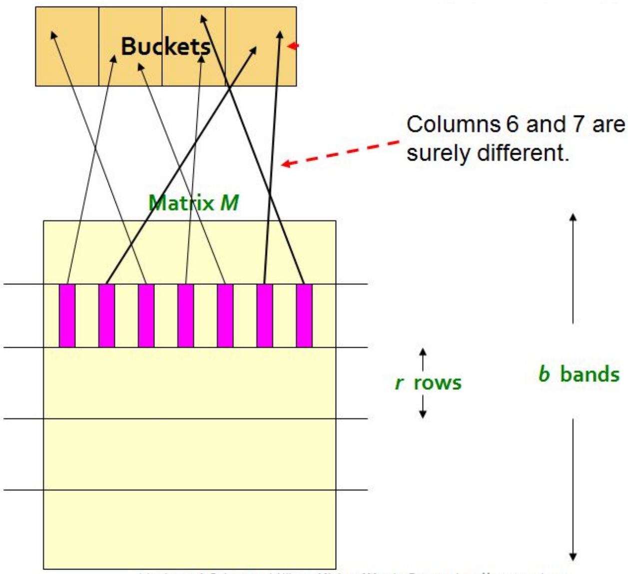
Step 1: Divide into *b* bands
Step 2: Hash columns within bands (one hash per band)



(Leskovec at al., 2014; http://www.mmds.org/)

Locality-Sensitive Hashing

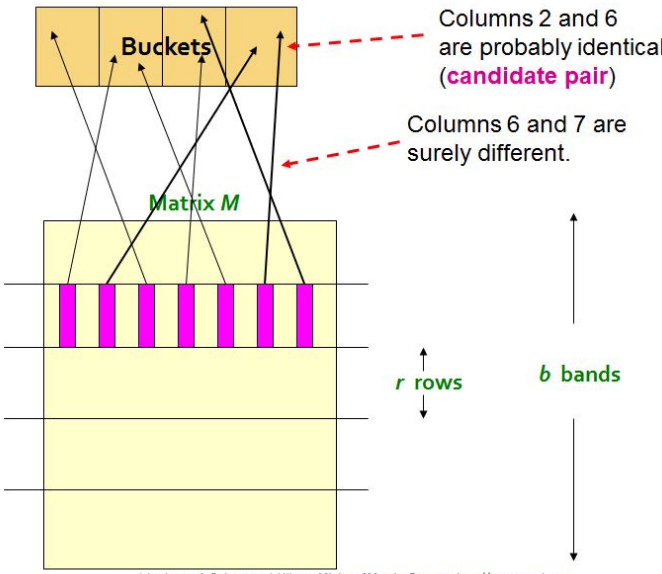
Step 1: Divide into *b* bands
Step 2: Hash columns within bands (one hash per band)



(Leskovec at al., 2014; http://www.mmds.org/)

Locality-Sensitive Hashing

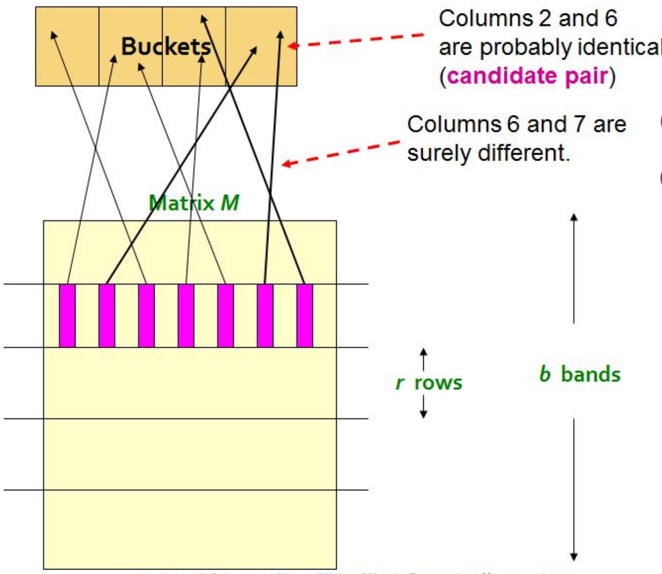
Step 1: Divide into *b* bands
Step 2: Hash columns within bands (one hash per band)



(Leskovec at al., 2014; http://www.mmds.org/)

Locality-Sensitive Hashing

Step 1: Divide into *b* bands
Step 2: Hash columns within bands (one hash per band)

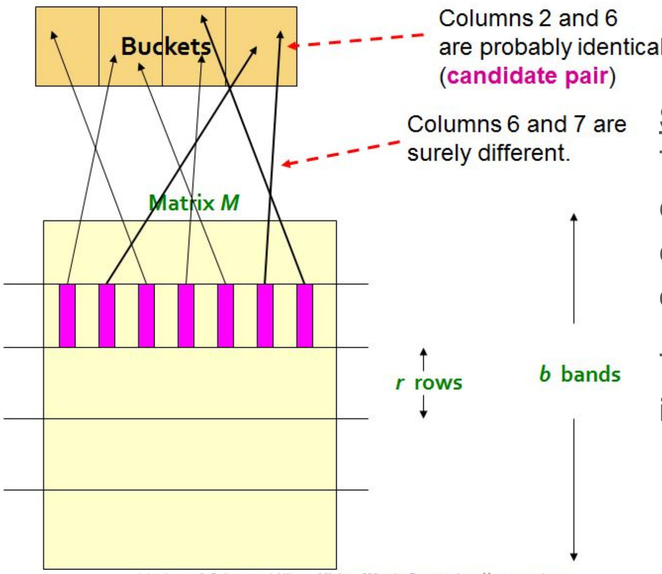


Criteria for being candidate pair:

- They end up in same bucket for at least 1 band.

(Leskovec at al., 2014; http://www.mmds.org/)

Locality-Sensitive Hashing



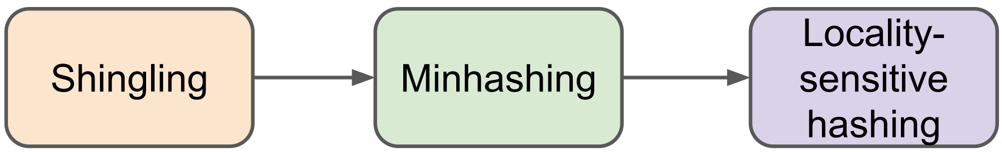
Step 1: Divide into *b* bands
Step 2: Hash columns within bands (one hash per band)

Simplification:
There are enough buckets compared to rows per band that columns must be identical in order to hash into same bucket.

Thus, we only need to check if identical within a band.

(Leskovec at al., 2014; http://www.mmds.org/)

Document Similarity Pipeline



Probabilities of agreement, Example

- 100,000 documents
- 100 random permutations/hash functions/rows

  => if 4byte integers then 40Mb to hold signature matrix

  => still 100k choose 2 is a lot (~5billion)

Probabilities of agreement, Example

- 100,000 documents
- 100 random permutations/hash functions/rows
  => if 4byte integers then 40Mb to hold signature matrix
  => still 100k choose 2 is a lot (~5billion)
- 20 bands of 5 rows
- Want 80% Jaccard Similarity ; for any row $p(S_1 == S_2) = .8$

Probabilities of agreement, Example

- 100,000 documents
- 100 random permutations/hash functions/rows

  => if 4byte integers then 40Mb to hold signature matrix

  => still 100k choose 2 is a lot (~5billion)
- 20 bands of 5 rows
- Want 80% Jaccard Similarity ; for any row $p(S_1 == S_2) = .8$

$P(S_1 == S_2 \mid b^{(5)})$: probability S1 and S2 agree within a given band

Probabilities of agreement, Example

- 100,000 documents
- 100 random permutations/hash functions/rows

  => if 4byte integers then 40Mb to hold signature matrix

  => still 100k choose 2 is a lot (~5billion)
- 20 bands of 5 rows
- Want 80% Jaccard Similarity ; for any row $p(S_1 == S_2) = .8$

$P(S_1{==}S_2 \mid b^{(5)})$: probability S1 and S2 agree within a given band

$= 0.8^5 = .328$

(Leskovec at al., 2014; http://www.mmds.org/)

Probabilities of agreement, Example

- 100,000 documents
- 100 random permutations/hash functions/rows

  => if 4byte integers then 40Mb to hold signature matrix

  => still 100k choose 2 is a lot (~5billion)
- 20 bands of 5 rows
- Want 80% Jaccard Similarity ; for any row $p(S_1 == S_2) = .8$

$P(S_1{=}{=}S_2 \mid b^{(5)})$: probability S1 and S2 agree within a given band

$= 0.8^5 = .328 \quad \Rightarrow \quad P(S_1 != S_2 \mid b) = 1 - .328 = .672$

(Leskovec at al., 2014; http://www.mmds.org/)

Probabilities of agreement, Example

- 100,000 documents
- 100 random permutations/hash functions/rows

  => if 4byte integers then 40Mb to hold signature matrix

  => still 100k choose 2 is a lot (~5billion)
- 20 bands of 5 rows
- Want 80% Jaccard Similarity ; for any row $p(S_1 == S_2) = .8$

$P(S_1==S_2 \mid b^{(5)})$: probability S1 and S2 agree within a given band

$= 0.8^5 = .328 \quad \Rightarrow \quad P(S_1 != S_2 \mid b) = 1-.328 = .672$

$P(S_1 != S_2)$: probability S1 and S2 do not agree in any band

(Leskovec at al., 2014; http://www.mmds.org/)

Probabilities of agreement, Example

- 100,000 documents
- 100 random permutations/hash functions/rows

  => if 4byte integers then 40Mb to hold signature matrix

  => still 100k choose 2 is a lot (~5billion)
- 20 bands of 5 rows
- Want 80% Jaccard Similarity ; for any row $p(S_1 == S_2) = .8$

$P(S_1{=}{=}S_2 \mid b^{(5)})$: probability S1 and S2 agree within a given band

$= 0.8^5 = .328 \quad \Rightarrow \quad P(S_1 != S_2 \mid b) = 1-.328 = .672$

$P(S_1 != S_2)$: probability S1 and S2 do not agree in any band

$= .672^{20} = .00035$

(Leskovec at al., 2014; http://www.mmds.org/)

Probabilities of agreement, Example

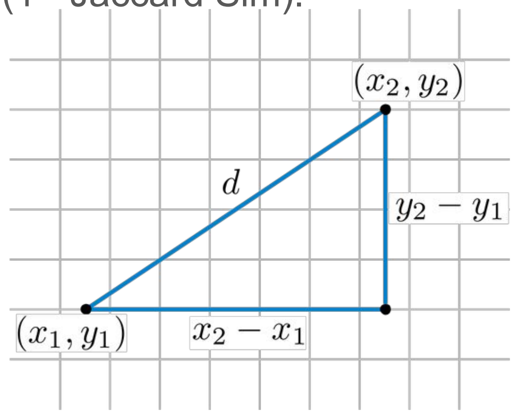
- 100,000 documents
- 100 random permutations/hash functions/rows
  => if 4byte integers then 40Mb to hold signature matrix
  => still 100k choose 2 is a lot (~5billion)
- 20 bands of 5 rows
- Want 80% Jaccard Similarity ; for any row $p(S_1 == S_2) = .8$

$P(S_1 == S_2 \mid b)$: probability S1 and S2 agree within a given band
$= 0.8^5 = .328 \Rightarrow P(S_1 != S_2 \mid b) = 1 - .328 = .672$
$P(S_1 != S_2)$: probability S1 and S2 do not agree in any band
$= .672^{20} = .00035$

What if wanting 40% Jaccard Similarity?

Distance Metrics

Pipeline gives us a way to find *near-neighbors* in *high-dimensional space* based on Jaccard Distance (1 - Jaccard Sim).

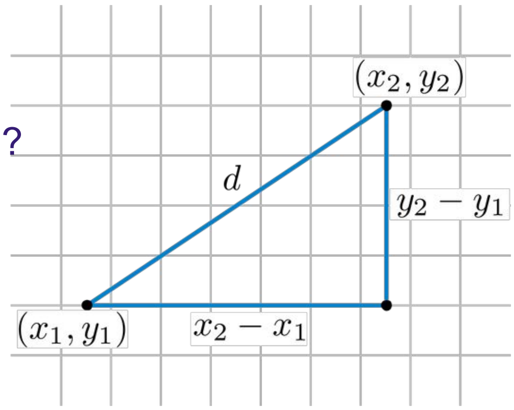


(http://rosalind.info/glossary/euclidean-distance/)

Distance Metrics

Pipeline gives us a way to find *near-neighbors* in *high-dimensional space* based on Jaccard Distance (1 - Jaccard Sim).

Typical properties of a distance metric, *d(point1,point2)*?



(http://rosalind.info/glossary/euclidean-distance/)

Distance Metrics

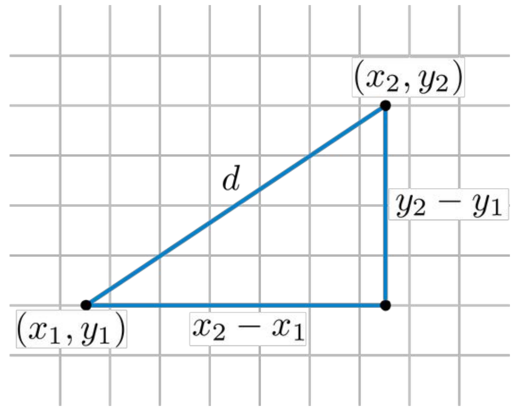
Pipeline gives us a way to find *near-neighbors* in *high-dimensional space* based on Jaccard Distance (1 - Jaccard Sim).

Typical properties of a distance metric, *d*:

$d(a, a) = 0$

$d(a, b) = d(b, a)$

$d(a, b) \leq d(a,c) + d(c,b)$



(http://rosalind.info/glossary/euclidean-distance/)

Distance Metrics

Pipeline gives us a way to find *near-neighbors* in *high-dimensional space* based on Jaccard Distance (1 - Jaccard Sim).

There are other metrics of similarity. e.g:

- Euclidean Distance
- Cosine Distance

  …

- Edit Distance
- Hamming Distance

Distance Metrics

Pipeline gives us a way to find *near-neighbors* in *high-dimensional space* based on Jaccard Distance (1 - Jaccard Sim).

There are other metrics of similarity. e.g:

- Euclidean Distance $$distance(X, Y) = \sqrt{\sum_{i}^{n}(x_i - y_i)^2}$$ ("L2 Norm")
- Cosine Distance

  …
- Edit Distance
- Hamming Distance

Distance Metrics

Pipeline gives us a way to find *near-neighbors* in *high-dimensional space* based on Jaccard Distance (1 - Jaccard Sim).

There are other metrics of similarity. e.g:

- Euclidean Distance

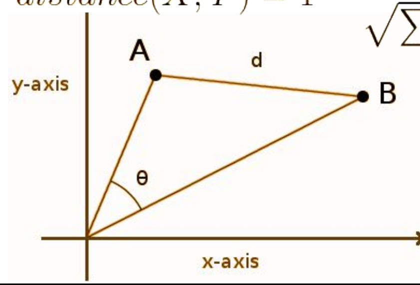
$$distance(X,Y) = \sqrt{\sum_{i}^{n}(x_i - y_i)^2}$$ ("L2 Norm")

- Cosine Distance

$$distance(X,Y) = 1 - \frac{\sum x_i y_i}{\sqrt{\sum x_i^2}\sqrt{\sum y_i^2}}$$

  …

- Edit Distance
- Hamming Distance



Locality Sensitive Hashing - Theory

LSH Can be generalized to many distance metrics by converting output to a probability and providing a lower bound on probability of being similar.

Locality Sensitive Hashing - Theory

LSH Can be generalized to many distance metrics by converting output to a probability and providing a lower bound on probability of being similar.

E.g. for euclidean distance:

- Choose random lines (analogous to hash functions in minhashing)
- Project the two points onto each line; match if two points within an interval

Side Note on Generating Hash Functions:

What hash functions to use?

Start with 2 decent hash functions

e.g. $h_a(x)$ = ascii(string) % large_prime_number
$h_b(x) = (3*$ascii(string) + $16)$ % large_prime_number

Add together multiplying the second times i:

$h_i(x) = h_a(x) + i*h_b(x) \% |BUCKETS|$
$e.g.\ h_5(x) = h_a(x) + 5*h_b(x) \ \% 100$

https://www.eecs.harvard.edu/~michaelm/postscripts/rsa2008.pdf

Popular choices: md5 (fast, predistable); mmh3 (easy to seed; fast)