COMPSCI 514: Algorithms for Data Science

Cameron Musco

University of Massachusetts Amherst. Fall 2022.

Lecture 9

Logistics

- Problem Set 2 is due next Friday, 10/14 at 11:59pm.
- The midterm is the following Thursday, 10/20 in class.
- Many students want some more time to go over Distinct Elements/Median trick/LogLog algorithm.
- I will plan to cover less material on high dimension geometry before the midterm and review this material instead.
- If we have time, I'll also go over some more practical use cases of distinct elements counting. Also see Lecture 9 slides.

Summary

Last Class:

δ

- Analysis of distinct elements counting vis MinHashing.
- The Median Trick to boost success probability.
- High-level overview of practical distinct elements algorithms (see posted slides for more info if you are interested).

Summary

Last Class:

- Analysis of distinct elements counting vis MinHashing.
- The Median Trick to boost success probability.
- High-level overview of practical distinct elements algorithms (see posted slides for more info if you are interested).

This Class:

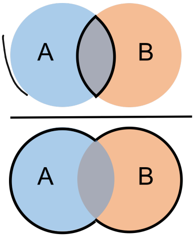
- Introduction of Jaccard similarity and the similarity search problem.
- Locality sensitive hashing for fast similarity search.
- MinHashing for Jaccard similarity search.

Jaccard Index: A similarity measure between two sets.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{\text{\# shared elements}}{\text{\# total elements}}. \qquad \in [0, 1]$$

A, B

$\cup$



Natural measure for similarity between bit strings – interpret an n bit string as a set, containing the elements corresponding the positions of its ones. $J(x, y) = \frac{\text{\# shared ones}}{\text{total ones}}$.

$[0, \overset{2}{1}, 0, \overset{4}{1}, 0]$

$\{2, 4\}$

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{\#\text{ shared elements}}{\#\text{ total elements}}.$$

Want Fast Implementations For:

- **Near Neighbor Search:** Have a database of n sets/bit strings and given a set A, want to find if it has high Jaccard similarity to anything in the database. $\Omega(n)$ time with a linear scan.
- **All-pairs Similarity Search:** Have n different sets/bit strings and want to find all pairs with high Jaccard similarity. $\Omega(n^2)$ time if we check all pairs explicitly.

Will speed up via randomized locality sensitive hashing.

Application: Document Similarity

Document Similarity:

- E.g., to detect plagiarism, copyright infringement, duplicate webpages, spam.
- Use Shingling + Jaccard similarity. (n-grams, k-mers)

The quick brown fox jumped high. → { the quick brown, quick brown fox, brown fox jumped, fox jumped high }

Application: Document Similarity

Document Similarity:

- E.g., to detect plagiarism, copyright infringement, duplicate webpages, spam.
- Use Shingling + Jaccard similarity. (n-grams, k-mers)

3

$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{2}{6} = \frac{1}{3}$

A

B

The slow brown fox jumped high

The quick brown fox jumped high. → { the quick brown, quick brown fox, brown fox jumped, fox jumped high }

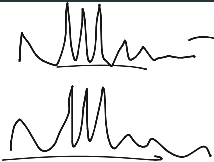
the slow brown
slow brown fox
⋮

ATAGCCGTAGT → { ATAG, TAGC, AGCC, GCCG, CCGT, CGTA, GTAG, TAGT }

Application: Audio Search

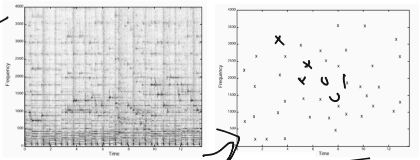
$\rightarrow S_1$

$\rightarrow S_2$



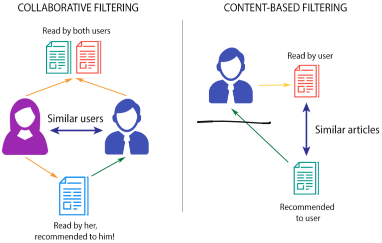
Audio Fingerprinting:

- E.g., in audio search (Shazam), Earthquake detection.
- Represent sound clip via a binary ‘fingerprint’ then compare with Jaccard similarity.



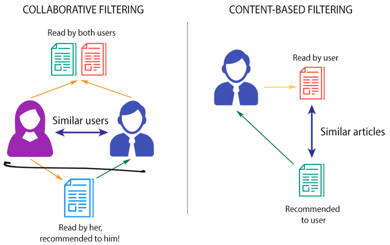
Application: Collaborative Filtering

Online recommendation systems are often based on **collaborative filtering**. Simplest approach: find similar users and make recommendations based on those users.



Application: Collaborative Filtering

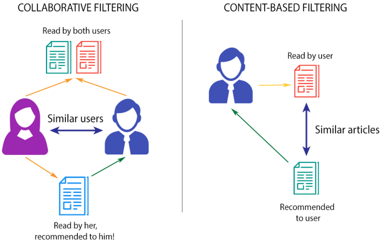
Online recommendation systems are often based on **collaborative filtering**. Simplest approach: find similar users and make recommendations based on those users.



- Twitter: represent a user as the set of accounts they follow. Match users based on the Jaccard similarity of these sets. Recommend that you follow accounts followed by similar users.

Application: Collaborative Filtering

Online recommendation systems are often based on **collaborative filtering**. Simplest approach: find similar users and make recommendations based on those users.



- Twitter: represent a user as the set of accounts they follow. Match users based on the Jaccard similarity of these sets. Recommend that you follow accounts followed by similar users.
- Netflix: look at sets of movies watched. Amazon: look at products purchased, etc.

Application: Entity Resolution

Entity Resolution Problem: Want to combine records from multiple data sources that refer to the same entities.

Entity Resolution Problem: Want to combine records from multiple data sources that refer to the same entities.

- E.g. data on individuals from voting registrations, property records, and social media accounts. Names and addresses may not exactly match, due to typos, nicknames, moves, etc.
- Still want to match records that all refer to the same person using all pairs similarity search.

Customer Records

Name	Phone Number	Account #
...	...	...
...	...	...
...	...	...
Cam Musco	1-401-555-5555	...
...	...	...
⋮	⋮	⋮

resolve

Public Property Records

Name	Phone Number	Address
...	...	...
...	...	...
...	...	...
Cameron Musco	402-555-5555	...
...	...	...
⋮	⋮	⋮

Application: Entity Resolution

Entity Resolution Problem: Want to combine records from multiple data sources that refer to the same entities.

- E.g. data on individuals from voting registrations, property records, and social media accounts. Names and addresses may not exactly match, due to typos, nicknames, moves, etc.
- Still want to match records that all refer to the same person using all pairs similarity search.

See Section 3.8.2 of *Mining Massive Datasets* for a discussion of a real world example involving 1 million customers. Naively this would be $\binom{1000000}{2} \approx$ 500 billion pairs of customers to check!

Application: Spam and Fraud Detection

Many applications to spam/fraud detection. E.g.

Application: Spam and Fraud Detection

Many applications to spam/fraud detection. E.g.

- Fake Reviews: Very common on websites like Amazon. Detection often looks for (near) duplicate reviews on similar products, which have been copied. 'Near duplicate' measured with shingles + Jaccard similarity.

Application: Spam and Fraud Detection

Many applications to spam/fraud detection. E.g.

- Fake Reviews: Very common on websites like Amazon. Detection often looks for (near) duplicate reviews on similar products, which have been copied. 'Near duplicate' measured with shingles + Jaccard similarity.
- Lateral phishing: Phishing emails sent to addresses at a business coming from a legitimate email address at the same business that has been compromised.
    - One method of detection looks at the recipient list of an email and checks if it has small Jaccard similarity with any previous recipient lists. If not, the email is flagged as possible spam.

Locality Sensitive Hashing

Goal: Speed up Jaccard similarity search (near neighbor and all-pairs similarity search).

Locality Sensitive Hashing

Goal: Speed up Jaccard similarity search (near neighbor and all-pairs similarity search).

Strategy: Locality sensitive hashing (LSH).

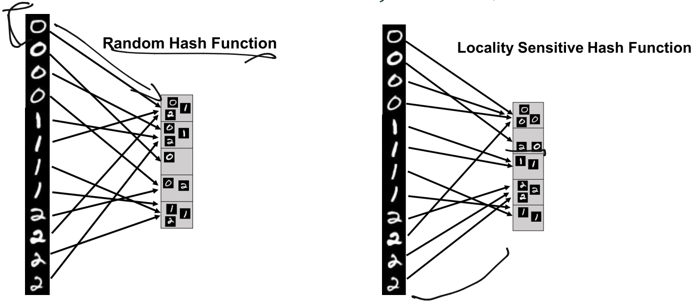
- Design a hash function where the collision probability is higher when two inputs are more similar (can design different functions for different similarity metrics.)

Locality Sensitive Hashing

Goal: Speed up Jaccard similarity search (near neighbor and all-pairs similarity search).

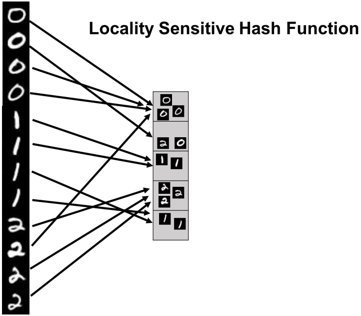
Strategy: Locality sensitive hashing (LSH).

- Design a hash function where the collision probability is higher when two inputs are more similar (can design different functions for different similarity metrics.)



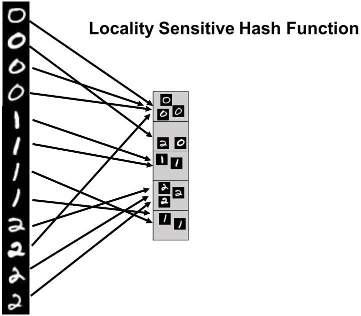
LSH For Similarity Search

How does locality sensitive hashing (LSH) help with similarity search?



LSH For Similarity Search

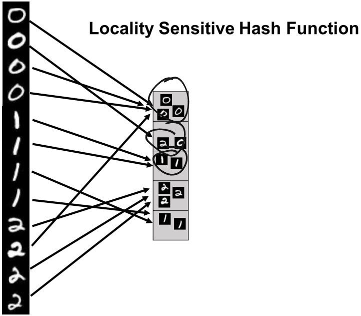
How does locality sensitive hashing (LSH) help with similarity search?



- **Near Neighbor Search:** Given item x, compute $h(x)$. Only search for similar items in the $h(x)$ bucket of the hash table.

LSH For Similarity Search

How does locality sensitive hashing (LSH) help with similarity search?



- **Near Neighbor Search:** Given item x, compute $h(x)$. Only search for similar items in the $h(x)$ bucket of the hash table.
- **All-pairs Similarity Search:** Scan through all buckets of the hash table and look for similar pairs within each bucket.

An Example: Locality sensitive hashing for Jaccard similarity.

MinHashing

An Example: Locality sensitive hashing for Jaccard similarity.

MinHash(A): [Andrei Broder, 1997 at Altavista]

.56 .73

- Let $\mathrm{h} : U \to [0, 1]$ be a random hash function
- $\mathrm{s} := 1$
- For $x_1, \ldots, x_{|A|} \in A$
  - $\mathrm{s} := \min(\mathrm{s}, \mathrm{h}(x_k))$
- Return s

the quick → .56

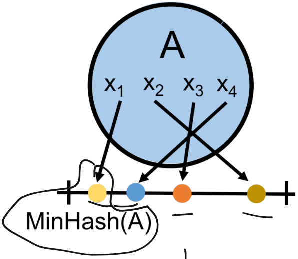
quick brown → .25

brown fox → .96

An Example: Locality sensitive hashing for Jaccard similarity.

MinHash(A): [Andrei Broder, 1997 at Altavista]

- Let $\mathbf{h} : U \rightarrow [0, 1]$ be a random hash function
- $\mathbf{s} := 1$
- For $x_1, \ldots, x_{|A|} \in A$
  - $\mathbf{s} := \min(\mathbf{s}, \mathbf{h}(x_k))$
- Return $\mathbf{s}$



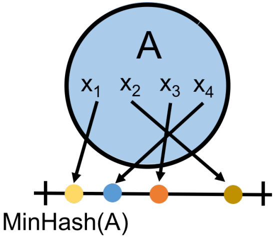
$$\mathbb{E}[\text{minHash}(A)] = \frac{1}{|A|+1}$$

MinHashing

An Example: Locality sensitive hashing for Jaccard similarity.

MinHash(A): [Andrei Broder, 1997 at Altavista]

- Let $\mathbf{h} : U \to [0, 1]$ be a random hash function
- $\mathbf{s} := 1$
- For $x_1, \ldots, x_{|A|} \in A$
  - $\mathbf{s} := \min(\mathbf{s}, \mathbf{h}(x_k))$
- Return $\mathbf{s}$



Identical to our distinct elements sketch!

MinHash Analysis

For two sets A and B, what is $\Pr(MinHash(A) = MinHash(B))$?

MinHash Analysis

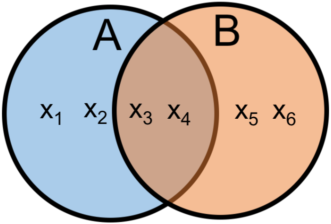
For two sets A and B, what is $\Pr(MinHash(A) = MinHash(B))$?

- Since we are hashing into the continuous range $[0, 1]$, we will never have $\mathrm{h}(x) = \mathrm{h}(y)$ for $x \neq y$ (i.e., no spurious collisions)

MinHash Analysis

For two sets A and B, what is $\Pr(MinHash(A) = MinHash(B))$?

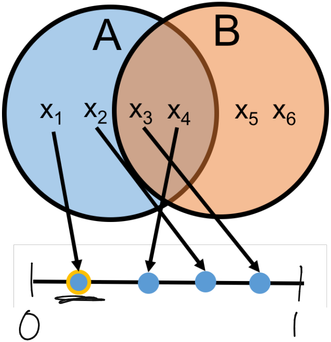
- Since we are hashing into the continuous range $[0, 1]$, we will never have $\mathbf{h}(x) = \mathbf{h}(y)$ for $x \neq y$ (i.e., no spurious collisions)



MinHash Analysis

For two sets A and B, what is $\Pr(MinHash(A) = MinHash(B))$?

- Since we are hashing into the continuous range $[0, 1]$, we will never have $\mathbf{h}(x) = \mathbf{h}(y)$ for $x \neq y$ (i.e., no spurious collisions)



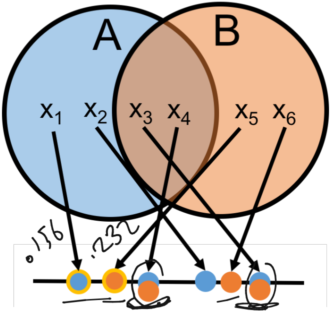
MinHash Analysis

For two sets A and B, what is $\Pr(MinHash(A) = MinHash(B))$?

- Since we are hashing into the continuous range $[0, 1]$, we will never have $\mathbf{h}(x) = \mathbf{h}(y)$ for $x \neq y$ (i.e., no spurious collisions)

random h

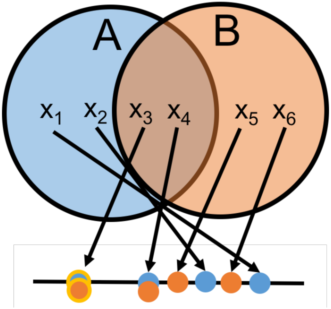
build minHash



MinHash Analysis

For two sets A and B, what is $\Pr(MinHash(A) = MinHash(B))$?

- Since we are hashing into the continuous range $[0, 1]$, we will never have $\mathbf{h}(x) = \mathbf{h}(y)$ for $x \neq y$ (i.e., no spurious collisions)

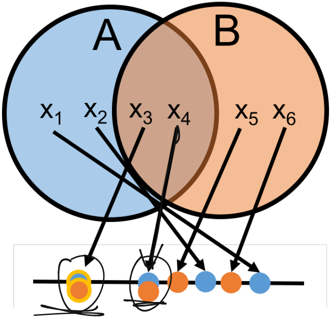


MinHash Analysis

For two sets A and B, what is $\Pr(MinHash(A) = MinHash(B))$?

- Since we are hashing into the continuous range $[0, 1]$, we will never have $\mathbf{h}(x) = \mathbf{h}(y)$ for $x \neq y$ (i.e., no spurious collisions)

[
]
0
1

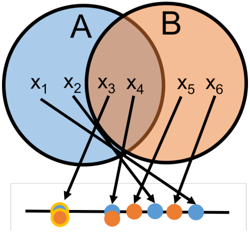


- $MinHash(A) = MinHash(B)$ only if an item in $A \cap B$ has the minimum hash value in both sets.

MinHash Analysis

For two sets A and B, what is $\Pr(MinHash(A) = MinHash(B))$?

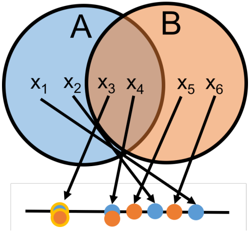
Claim: $MinHash(A) = MinHash(B)$ only if an item in $A \cap B$ has the minimum hash value in both sets.



MinHash Analysis

For two sets A and B, what is $\Pr(MinHash(A) = MinHash(B))$?

Claim: $MinHash(A) = MinHash(B)$ only if an item in $A \cap B$ has the minimum hash value in both sets.

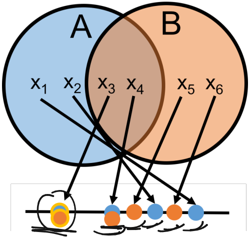


$\Pr(MinHash(A) = MinHash(B)) = ?$

MinHash Analysis

For two sets A and B, what is $\Pr(MinHash(A) = MinHash(B))$?

Claim: $MinHash(A) = MinHash(B)$ only if an item in $A \cap B$ has the minimum hash value in both sets.

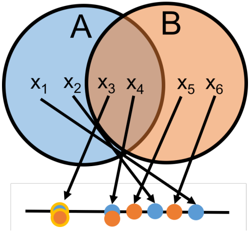


$$\Pr(MinHash(A) = MinHash(B)) = \frac{|A \cap B|}{\text{total \# items hashed}} = J(A,B)$$

MinHash Analysis

For two sets A and B, what is $\Pr(MinHash(A) = MinHash(B))$?

Claim: $MinHash(A) = MinHash(B)$ only if an item in $A \cap B$ has the minimum hash value in both sets.



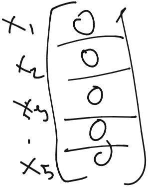
$$\Pr(MinHash(A) = MinHash(B)) = \frac{|A \cap B|}{\text{total \# items hashed}} = \frac{|A \cap B|}{|A \cup B|}$$

MinHash Analysis

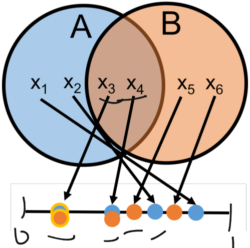
For two sets A and B, what is $\Pr(MinHash(A) = MinHash(B))$?

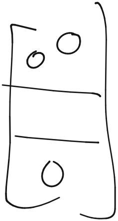
Claim: $MinHash(A) = MinHash(B)$ only if an item in $A \cap B$ has the minimum hash value in both sets.

$J(x_1, x_2) = .8$



$J(x_2, x_3) = .8$





$$\Pr(MinHash(A) = MinHash(B)) = \frac{|A \cap B|}{\text{total \# items hashed}}$$

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

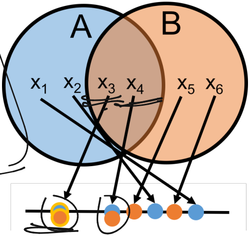
MinHash Analysis

For two sets A and B, what is $\Pr(MinHash(A) = MinHash(B))$?

Claim: $MinHash(A) = MinHash(B)$ only if an item in $A \cap B$ has the minimum hash value in both sets.

A → .05 .06 .72 .21

B → .72 .21 .45 .01



A: x_1 x_2 x_3 x_4

B: x_1 x_2 x_3 x_5

$J(A,B) = \frac{3}{5}$

$$\Pr(MinHash(A) = MinHash(B)) = \frac{|A \cap B|}{\text{total \# items hashed}}$$

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

Locality sensitive: the higher $J(A, B)$ is, the more likely $MinHash(A), MinHash(B)$ are to collide.

Similarity Search with MinHash

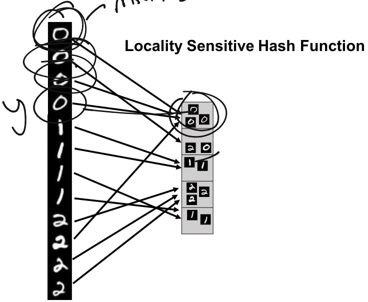
Goal: Given a document y, identify all documents x in a database with Jaccard similarity (of their shingle sets) $J(x, y) \geq 1/2$.

Goal: Given a document y, identify all documents x in a database with Jaccard similarity (of their shingle sets) $J(x, y) \geq 1/2$.

Our Approach:

- Create a hash table of size m, choose a random hash function $g : [0, 1] \to [m]$, and insert every item x into bucket $g(MinHash(x))$. Search for items similar to y in bucket $g(MinHash(y))$.

minHash(..) = .92



Similarity Search with MinHash

Goal: Given a document y, identify all documents x in a database with Jaccard similarity (of their shingle sets) $J(x, y) \geq 1/2$.

Our Approach:

- Create a hash table of size m, choose a random hash function $\mathbf{g} : [0, 1] \to [m]$, and insert every item x into bucket $\mathbf{g}(MinHash(x))$. Search for items similar to y in bucket $\mathbf{g}(MinHash(y))$.
- What is $\Pr[\mathbf{g}(MinHash(x)) = \mathbf{g}(MinHash(y))]$ assuming $J(x, y) = 1/2$ and $\mathbf{g}$ is collision free?

MinHash(x) = MinHash(y)

Goal: Given a document y, identify all documents x in a database with Jaccard similarity (of their shingle sets) $J(x, y) \geq 1/2$.

Our Approach:

- Create a hash table of size m, choose a random hash function $\mathbf{g} : [0, 1] \to [m]$, and insert every item x into bucket $\mathbf{g}(MinHash(x))$. Search for items similar to y in bucket $\mathbf{g}(MinHash(y))$.
- What is $\Pr[\mathbf{g}(MinHash(x)) = \mathbf{g}(MinHash(y))]$ assuming $J(x, y) = 1/2$ and $\mathbf{g}$ is collision free?
- For every document x in your database with $J(x, y) \geq 1/2$ what is the probability you will find x in bucket $\mathbf{g}(MinHash(y))$?

$\geq 1/2$

Reducing False Negatives

With a simple use of MinHash, we miss a match x with $J(x, y) = 1/2$ with probability $1/2$. How can we reduce this false negative rate?

Reducing False Negatives

With a simple use of MinHash, we miss a match x with $J(x, y) = 1/2$ with probability $1/2$. How can we reduce this false negative rate?

Repetition: Run MinHash t times independently, to produce hash values $MH_1(x), \ldots, MH_t(x)$. Apply random hash function $\mathbf{g}$ to map all these values to locations in t hash tables.

Reducing False Negatives

With a simple use of MinHash, we miss a match x with $J(x, y) = 1/2$ with probability $1/2$. How can we reduce this false negative rate?

Repetition: Run MinHash t times independently, to produce hash values $MH_1(x), \ldots, MH_t(x)$. Apply random hash function $\mathbf{g}$ to map all these values to locations in t hash tables.

- To search for items similar to y, look at all items in bucket $\mathbf{g}(MH_1(y))$ of the 1^{st} table, bucket $\mathbf{g}(MH_2(y))$ of the 2^{nd} table, etc.

Reducing False Negatives

With a simple use of MinHash, we miss a match x with $J(x, y) = 1/2$ with probability $1/2$. How can we reduce this false negative rate?

Repetition: Run MinHash t times independently, to produce hash values $MH_1(x), \ldots, MH_t(x)$. Apply random hash function $\mathbf{g}$ to map all these values to locations in t hash tables.

- To search for items similar to y, look at all items in bucket $\mathbf{g}(MH_1(y))$ of the 1^{st} table, bucket $\mathbf{g}(MH_2(y))$ of the 2^{nd} table, etc.
- What is the probability that x with $J(x, y) = 1/2$ is in at least one of these buckets, assuming for simplicity $\mathbf{g}$ has no collisions?

Reducing False Negatives

With a simple use of MinHash, we miss a match x with $J(x, y) = 1/2$ with probability $1/2$. How can we reduce this false negative rate?

Repetition: Run MinHash t times independently, to produce hash values $MH_1(x), \ldots, MH_t(x)$. Apply random hash function $\mathbf{g}$ to map all these values to locations in t hash tables.

- To search for items similar to y, look at all items in bucket $\mathbf{g}(MH_1(y))$ of the 1^{st} table, bucket $\mathbf{g}(MH_2(y))$ of the 2^{nd} table, etc.
- What is the probability that x with $J(x, y) = 1/2$ is in at least one of these buckets, assuming for simplicity $\mathbf{g}$ has no collisions?
  $1-$ (probability in *no* buckets)

Reducing False Negatives

With a simple use of MinHash, we miss a match x with $J(x, y) = 1/2$ with probability $1/2$. How can we reduce this false negative rate?

Repetition: Run MinHash t times independently, to produce hash values $MH_1(x), \ldots, MH_t(x)$. Apply random hash function $\mathbf{g}$ to map all these values to locations in t hash tables.

- To search for items similar to y, look at all items in bucket $\mathbf{g}(MH_1(y))$ of the 1^{st} table, bucket $\mathbf{g}(MH_2(y))$ of the 2^{nd} table, etc.
- What is the probability that x with $J(x, y) = 1/2$ is in at least one of these buckets, assuming for simplicity $\mathbf{g}$ has no collisions?
  $1-$ (probability in *no* buckets) $= \underbrace{1 - \left(\frac{1}{2}\right)^t}$

Reducing False Negatives

With a simple use of MinHash, we miss a match x with $J(x, y) = 1/2$ with probability 1/2. How can we reduce this false negative rate?

Repetition: Run MinHash t times independently, to produce hash values $MH_1(x), \ldots, MH_t(x)$. Apply random hash function $\mathbf{g}$ to map all these values to locations in t hash tables.

- To search for items similar to y, look at all items in bucket $\mathbf{g}(MH_1(y))$ of the 1^{st} table, bucket $\mathbf{g}(MH_2(y))$ of the 2^{nd} table, etc.
- What is the probability that x with $J(x, y) = 1/2$ is in at least one of these buckets, assuming for simplicity $\mathbf{g}$ has no collisions?
  $1-$ (probability in *no* buckets) $= 1 - \left(\frac{1}{2}\right)^t \approx .99$ for $t = 7$.

Reducing False Negatives

With a simple use of MinHash, we miss a match x with $J(x, y) = 1/2$ with probability 1/2. How can we reduce this false negative rate?

Repetition: Run MinHash t times independently, to produce hash values $MH_1(x), \ldots, MH_t(x)$. Apply random hash function $\mathbf{g}$ to map all these values to locations in t hash tables.

- To search for items similar to y, look at all items in bucket $\mathbf{g}(MH_1(y))$ of the 1^{st} table, bucket $\mathbf{g}(MH_2(y))$ of the 2^{nd} table, etc.
- What is the probability that x with $J(x, y) = 1/2$ is in at least one of these buckets, assuming for simplicity $\mathbf{g}$ has no collisions?
  $1-$ (probability in *no* buckets) $= 1 - \left(\frac{1}{2}\right)^t \approx .99$ for $t = 7$.
- What is the probability that x with $J(x, y) = 1/4$ is in at least one of these buckets, assuming for simplicity $\mathbf{g}$ has no collisions?

$1 - \left(\frac{3}{4}\right)^t$

Reducing False Negatives

With a simple use of MinHash, we miss a match x with $J(x, y) = 1/2$ with probability 1/2. How can we reduce this false negative rate?

Repetition: Run MinHash t times independently, to produce hash values $MH_1(x), \ldots, MH_t(x)$. Apply random hash function $\mathbf{g}$ to map all these values to locations in t hash tables.

- To search for items similar to y, look at all items in bucket $\mathbf{g}(MH_1(y))$ of the 1^{st} table, bucket $\mathbf{g}(MH_2(y))$ of the 2^{nd} table, etc.
- What is the probability that x with $J(x, y) = 1/2$ is in at least one of these buckets, assuming for simplicity $\mathbf{g}$ has no collisions? $1-$ (probability in *no* buckets) $= 1 - \left(\frac{1}{2}\right)^t \approx .99$ for $t = 7$.
- What is the probability that x with $J(x, y) = 1/4$ is in at least one of these buckets, assuming for simplicity $\mathbf{g}$ has no collisions? $1-$ (probability in *no* buckets) $= 1 - \left(\frac{3}{4}\right)^t$

Reducing False Negatives

With a simple use of MinHash, we miss a match x with $J(x, y) = 1/2$ with probability 1/2. How can we reduce this false negative rate?

Repetition: Run MinHash t times independently, to produce hash values $MH_1(x), \ldots, MH_t(x)$. Apply random hash function $\mathbf{g}$ to map all these values to locations in t hash tables.

- To search for items similar to y, look at all items in bucket $\mathbf{g}(MH_1(y))$ of the 1^{st} table, bucket $\mathbf{g}(MH_2(y))$ of the 2^{nd} table, etc.
- What is the probability that x with $J(x, y) = 1/2$ is in at least one of these buckets, assuming for simplicity $\mathbf{g}$ has no collisions?
  1− (probability in *no* buckets) $= 1 - \left(\frac{1}{2}\right)^t \approx .99$ for $t = 7$.
- What is the probability that x with $J(x, y) = 1/4$ is in at least one of these buckets, assuming for simplicity $\mathbf{g}$ has no collisions?
  1− (probability in *no* buckets) $= 1 - \left(\frac{3}{4}\right)^t \approx .87$ for $t = 7$.

Reducing False Negatives

With a simple use of MinHash, we miss a match x with $J(x,y) = 1/2$ with probability $1/2$. How can we reduce this false negative rate?

Repetition: Run MinHash t times independently, to produce hash values $MH_1(x), \ldots, MH_t(x)$. Apply random hash function $\mathbf{g}$ to map all these values to locations in t hash tables.

- To search for items similar to y, look at all items in bucket $\mathbf{g}(MH_1(y))$ of the 1^{st} table, bucket $\mathbf{g}(MH_2(y))$ of the 2^{nd} table, etc.
- What is the probability that x with $J(x,y) = 1/2$ is in at least one of these buckets, assuming for simplicity $\mathbf{g}$ has no collisions?
  $1-$ (probability in *no* buckets) $= 1 - \left(\frac{1}{2}\right)^t \approx .99$ for $t = 7$.
- What is the probability that x with $J(x,y) = \underline{1/4}$ is in at least one of these buckets, assuming for simplicity $\mathbf{g}$ has no collisions?
  $1-$ (probability in *no* buckets) $= 1 - \left(\frac{3}{4}\right)^t \approx .87$ for $t = 7$.

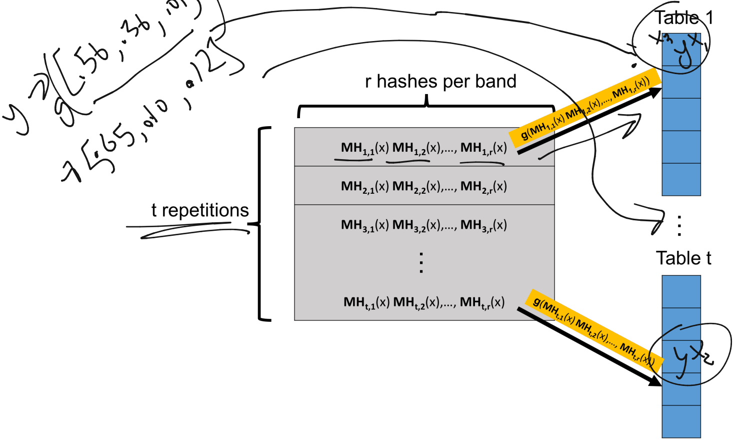
Potential for a lot of false positives! Slows down search time.

Balancing Hit Rate and Query Time

We want to balance a small probability of false negatives (a high hit rate) with a small probability of false positives (a small query time.)

Balancing Hit Rate and Query Time

We want to balance a small probability of false negatives (a high hit rate) with a small probability of false positives (a small query time.)



Create t hash tables. Each is indexed into not with a single MinHash value, but with r values, appended together. A length r signature.

Balancing Hit Rate and Query Time

Consider searching for matches in t hash tables, using MinHash signatures of length r. For x and y with Jaccard similarity $J(x, y) = s$:

Balancing Hit Rate and Query Time

Consider searching for matches in t hash tables, using MinHash signatures of length r. For x and y with Jaccard similarity $J(x, y) = s$:

- Probability that a single hash matches.
  $\Pr\left[MH_{i,j}(x) = MH_{i,j}(y)\right] = J(x, y) = s$.

Balancing Hit Rate and Query Time

Consider searching for matches in t hash tables, using MinHash signatures of length r. For x and y with Jaccard similarity $J(x,y) = s$:

- Probability that a single hash matches.
  $\Pr\left[MH_{i,j}(x) = MH_{i,j}(y)\right] = J(x,y) = s$.
- Probability that x and y having matching signatures in repetition i. $\Pr\left[MH_{i,1}(x), \ldots, MH_{i,r}(x) = MH_{i,1}(y), \ldots, MH_{i,r}(y)\right]$.

$y \rightarrow [.56, .75, .01]$

$x \rightarrow [.56, .75, .001]$

Balancing Hit Rate and Query Time

Consider searching for matches in t hash tables, using MinHash signatures of length r. For x and y with Jaccard similarity $J(x, y) = s$:

- Probability that a single hash matches. $\Pr\left[MH_{i,j}(x) = MH_{i,j}(y)\right] = J(x, y) = s$.
- Probability that x and y having matching signatures in repetition i. $\Pr\left[MH_{i,1}(x), \ldots, MH_{i,r}(x) = MH_{i,1}(y), \ldots, MH_{i,r}(y)\right] = s^r$.

Balancing Hit Rate and Query Time

Consider searching for matches in t hash tables, using MinHash signatures of length r. For x and y with Jaccard similarity $J(x, y) = s$:

- Probability that a single hash matches.
  $\Pr\left[MH_{i,j}(x) = MH_{i,j}(y)\right] = J(x, y) = s$.
- Probability that x and y having matching signatures in repetition i. $\Pr\left[MH_{i,1}(x), \ldots, MH_{i,r}(x) = MH_{i,1}(y), \ldots, MH_{i,r}(y)\right] = s^r$.
- Probability that x and y don't match in repetition i:

Balancing Hit Rate and Query Time

Consider searching for matches in t hash tables, using MinHash signatures of length r. For x and y with Jaccard similarity $J(x, y) = s$:

- Probability that a single hash matches.
  $\Pr\left[MH_{i,j}(x) = MH_{i,j}(y)\right] = J(x, y) = s$.
- Probability that x and y having matching signatures in repetition i. $\Pr\left[MH_{i,1}(x), \ldots, MH_{i,r}(x) = MH_{i,1}(y), \ldots, MH_{i,r}(y)\right] = s^r$.
- Probability that x and y don't match in repetition i: $1 - s^r$.

Balancing Hit Rate and Query Time

Consider searching for matches in t hash tables, using MinHash signatures of length r. For x and y with Jaccard similarity $J(x, y) = s$:

- Probability that a single hash matches. $\Pr\left[MH_{i,j}(x) = MH_{i,j}(y)\right] = J(x, y) = s$.
- Probability that x and y having matching signatures in repetition i. $\Pr\left[MH_{i,1}(x), \ldots, MH_{i,r}(x) = MH_{i,1}(y), \ldots, MH_{i,r}(y)\right] = s^r$.
- Probability that x and y don't match in repetition i: $1 - s^r$.
- Probability that x and y don't match in *all repetitions*:

Balancing Hit Rate and Query Time

Consider searching for matches in t hash tables, using MinHash signatures of length r. For x and y with Jaccard similarity $J(x, y) = s$:

- Probability that a single hash matches.
  $\Pr\left[MH_{i,j}(x) = MH_{i,j}(y)\right] = J(x, y) = s$.
- Probability that x and y having matching signatures in repetition i. $\Pr\left[MH_{i,1}(x), \ldots, MH_{i,r}(x) = MH_{i,1}(y), \ldots, MH_{i,r}(y)\right] = s^r$.
- Probability that x and y don't match in repetition i: $1 - s^r$.
- Probability that x and y don't match in *all repetitions*: $(1 - s^r)^t$.

Balancing Hit Rate and Query Time

Consider searching for matches in t hash tables, using MinHash signatures of length r. For x and y with Jaccard similarity $J(x, y) = s$:

- Probability that a single hash matches.
  $\Pr\left[MH_{i,j}(x) = MH_{i,j}(y)\right] = J(x, y) = s$.
- Probability that x and y having matching signatures in repetition i. $\Pr\left[MH_{i,1}(x), \ldots, MH_{i,r}(x) = MH_{i,1}(y), \ldots, MH_{i,r}(y)\right] = s^r$.
- Probability that x and y don't match in repetition i: $1 - s^r$.
- Probability that x and y don't match in *all repetitions*: $(1 - s^r)^t$.
- Probability that x and y match in at least one repetition:

Balancing Hit Rate and Query Time

Consider searching for matches in t hash tables, using MinHash signatures of length r. For x and y with Jaccard similarity $J(x,y) = s$:

- Probability that a single hash matches. $\Pr\left[MH_{i,j}(x) = MH_{i,j}(y)\right] = J(x,y) = s$.
- Probability that x and y having matching signatures in repetition i. $\Pr\left[MH_{i,1}(x), \ldots, MH_{i,r}(x) = MH_{i,1}(y), \ldots, MH_{i,r}(y)\right] = s^r$.
- Probability that x and y don't match in repetition i: $1 - s^r$.
- Probability that x and y don't match in *all repetitions*: $(1 - s^r)^t$.
- Probability that x and y match in at least one repetition:

$$\text{Hit Probability: } 1 - (1 - s^r)^t.$$