COMPSCI 514: ALGORITHMS FOR DATA SCIENCE

Cameron Musco
University of Massachusetts Amherst. Spring 2020.
Lecture 6

LOGISTICS

- Problem Set 1 is due Friday at 8pm in Gradescope.

- Problem Set 1 is due Friday at 8pm in Gradescope.
- Thanks for your feedback in Piazza on lecture pace.
  - 6% a bit too slow.
  - 22% just right.
  - 63% a bit too fast.
  - 8% way too fast.

Last Class:

Last Class:

- Distinct Elements via Hashing:
  - Distinct elements algorithm using min-of-hashes approach.
  - Analysis using averaging and the ‘median trick’.
  - Practical implementations (HyperLogLog) and applications.

Last Class:

- Distinct Elements via Hashing:
  - Distinct elements algorithm using min-of-hashes approach.
  - Analysis using averaging and the 'median trick'.
  - Practical implementations (HyperLogLog) and applications.
- Jaccard Similarity:
  - Jaccard similarity as a similarity metric between sets and binary strings. Applications in document comparison and audio fingerprinting.

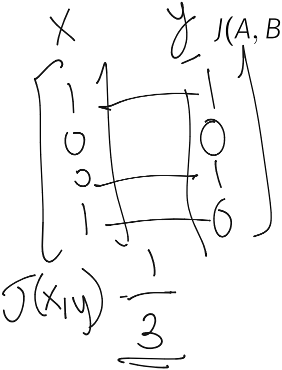
Last Class:

- Distinct Elements via Hashing:
  - Distinct elements algorithm using min-of-hashes approach.
  - Analysis using averaging and the 'median trick'.
  - Practical implementations (HyperLogLog) and applications.
- Jaccard Similarity:
  - Jaccard similarity as a similarity metric between sets and binary strings. Applications in document comparison and audio fingerprinting.

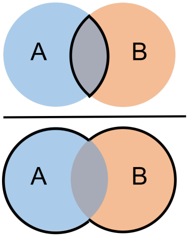
This Class:

- See how a min-of-hashes approach (MinHash) is used to estimate the Jaccard similarity.
- Application of MinHash to fast similarity search.
- Locality sensitive hashing.

Jaccard Index: A similarity measure between two sets.



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



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

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

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.

SEARCH WITH JACCARD SIMILARITY

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

Want Fast Implementations For:

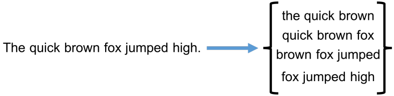
- **Near Neighbor Search:** Have a database of n sets/bit strings and given a set A, want to find if it has high Jaccard similarity to anything in the database. $\Omega(n)$ time with a linear scan.
- **All-pairs Similarity Search:** Have n different sets/bit strings and want to find all pairs with high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

Will speed up via randomized locality sensitive hashing.

Why can't we just use e.g. binary search or regular hash tables to speed up these search problems?

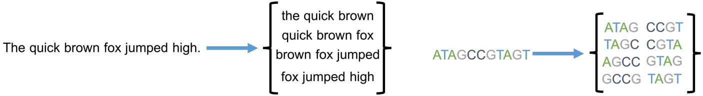
Document Similarity:

- E.g., to detect plagiarism, copyright infringement, duplicate webpages, spam.
- Use Shingling + Jaccard similarity.



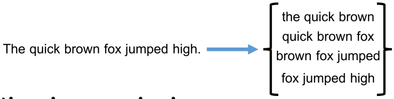
Document Similarity:

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



Document Similarity:

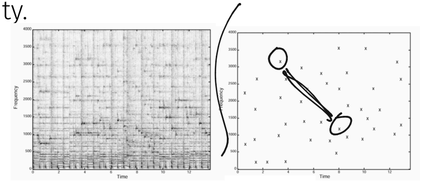
- E.g., to detect plagiarism, copyright infringement, duplicate webpages, spam.
- Use Shingling + Jaccard similarity. (n-grams, k-mers)



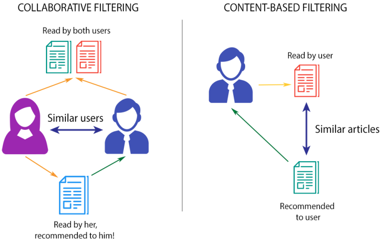
Audio Fingerprinting:

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

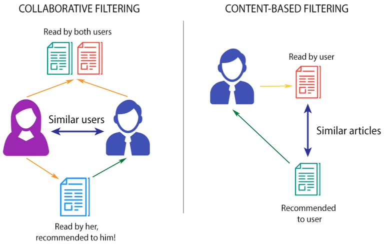
- not robust to shifting



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

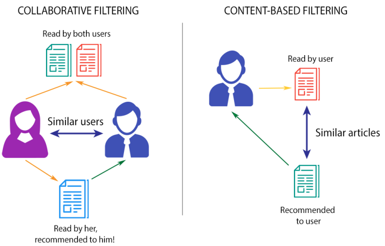


Online recommendation systems are often based on **collaborative filtering**. Simplest approach: find similar users and make recommendations based on those users.



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

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 similar 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.

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	...
...	...	...
⋮	⋮	⋮

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!

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.

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.

Why use Jaccard similarity over other metrics like: Hamming distinct (bit strings), correlation (sound waves, seismograms), edit distance (text, genome sequences, etc.)?

WHY JACCARD SIMILARITY?

[a ~~dog~~ walked]   [the cat walked]
cat

Why use Jaccard similarity over other metrics like: Hamming distinct (bit strings), correlation (sound waves, seismograms), edit distance (text, genome sequences, etc.)?

Two Reasons:

- Depending on the application, often is a very good measure.
- Even when not ideal, very efficient to compute and (as we will see today) implement near neighbor search and all-pairs similarity search with.

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

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

Strategy: Use random hashing to map each set to a very compressed representation. Jaccard similarity can be estimated from these representations.

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

Strategy: Use random hashing to map each set to a very compressed representation. Jaccard similarity can be estimated from these representations.

MinHash(A): A random hash function, built on top of a random hash function! [Andrei Broder, 1997 at Altavista]

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

Strategy: Use random hashing to map each set to a very compressed representation. Jaccard similarity can be estimated from these representations.

MinHash(A): A random hash function, built on top of a random hash function! [Andrei Broder, 1997 at Altavista]

- Let $\mathrm{h} : U \rightarrow [0, 1]$ be a random hash function

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

Strategy: Use random hashing to map each set to a very compressed representation. Jaccard similarity can be estimated from these representations.

MinHash(A): A random hash function, built on top of a random hash function! [Andrei Broder, 1997 at Altavista]

- Let $\mathbf{h} : U \rightarrow [0, 1]$ be a random hash function
- $\mathbf{s} := 1$

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

Strategy: Use random hashing to map each set to a very compressed representation. Jaccard similarity can be estimated from these representations.

MinHash(A): A random hash function, built on top of a random hash function! [Andrei Broder, 1997 at Altavista]

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

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

Strategy: Use random hashing to map each set to a very compressed representation. Jaccard similarity can be estimated from these representations.

MinHash(A): A random hash function, built on top of a random hash function! [Andrei Broder, 1997 at Altavista]

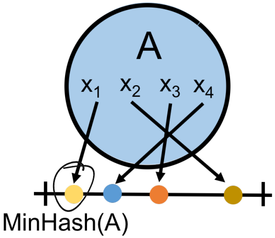
- Let $\mathbf{h} : U \rightarrow [0, 1]$ be a random hash function
- $\mathbf{s} := 1$
- For $x_1, \ldots, x_{|A|} \in A$
  - $\mathbf{s} := \min(\mathbf{s}, \mathbf{h}(x_k))$
- Return $\mathbf{s}$:= minHash(A)

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

Strategy: Use random hashing to map each set to a very compressed representation. Jaccard similarity can be estimated from these representations.

MinHash(A): A random hash function, built on top of a random hash function! [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 $\mathbf{s}$



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

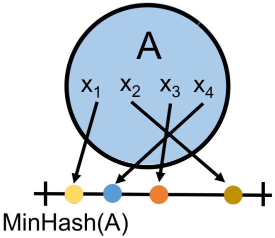
Strategy: Use random hashing to map each set to a very compressed representation. Jaccard similarity can be estimated from these representations.

MinHash(A): A random hash function, built on top of a random hash function! [Andrei Broder, 1997 at Altavista]

- Let $\mathbf{h} : U \rightarrow [0, 1]$ be a random hash function

Same h?

- $\mathbf{s} := 1$
- 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!

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

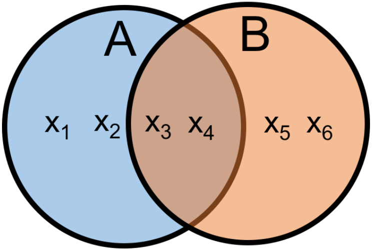
For two sets A and B, what is $\Pr(MinHash(A) = MinHash(B))$?

- 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)

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

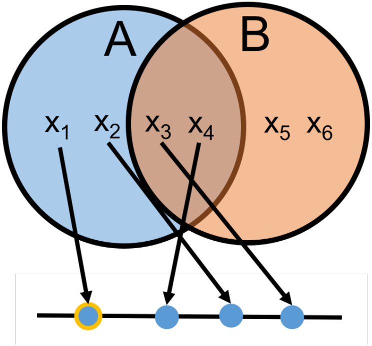
- Since we are hashing into the continuous range $[0, 1]$, we will never have $\mathbf{h}(x) = \mathbf{h}(y)$ for $x \neq y$ (i.e., no spurious collisions)

$\frac{2}{6} = \frac{1}{3}$



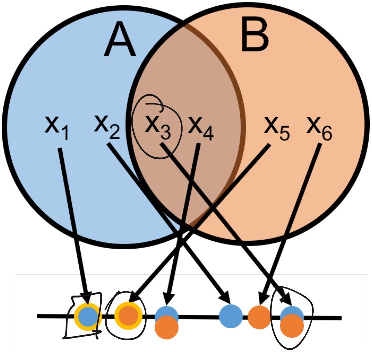
For two sets A and B, what is $\Pr(MinHash(A) = MinHash(B))$?

- 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)



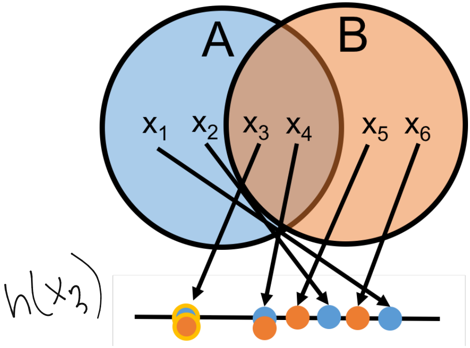
For two sets A and B, what is $\Pr(MinHash(A) = MinHash(B))$?

- 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)



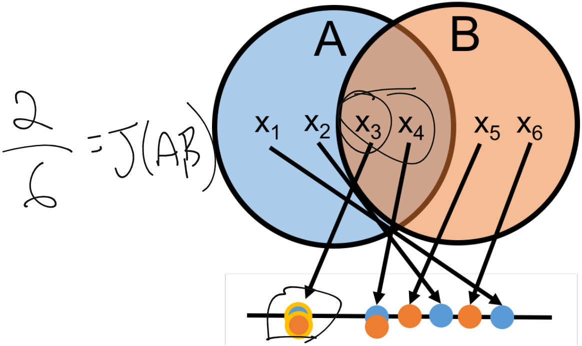
For two sets A and B, what is $\Pr(MinHash(A) = MinHash(B))$?

- 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)



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

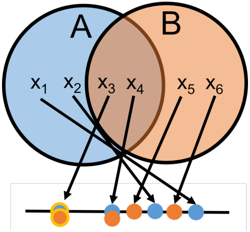
- Since we are hashing into the continuous range $[0, 1]$, we will never have $\mathbf{h}(x) = \mathbf{h}(y)$ for $x \neq y$ (i.e., no spurious collisions)



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

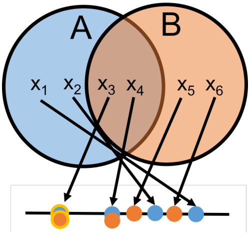
For two sets A and B, what is $\Pr(MinHash(A) = MinHash(B))$?

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



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

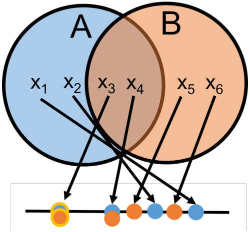
Claim: $MinHash(A) = MinHash(B)$ only if an item in $A \cap B$ has the minimum hash value in both sets.



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

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

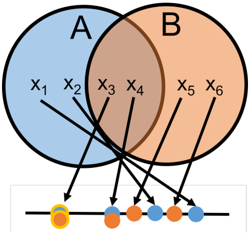
Claim: $MinHash(A) = MinHash(B)$ only if an item in $A \cap B$ has the minimum hash value in both sets.



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

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

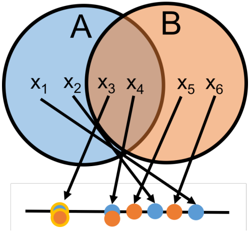
Claim: $MinHash(A) = MinHash(B)$ only if an item in $A \cap B$ has the minimum hash value in both sets.



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

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.



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

Upshot: MinHash reduces estimating the Jaccard similarity to checking equality of a *single number*.

$$\Pr(MinHash(A) = MinHash(B)) = J(A, B).$$

Upshot: MinHash reduces estimating the Jaccard similarity to checking equality of a *single number*.

$$\Pr(MinHash(A) = MinHash(B)) = J(A, B).$$

- An instance of locality sensitive hashing (LSH).

Upshot: MinHash reduces estimating the Jaccard similarity to checking equality of a *single number*.

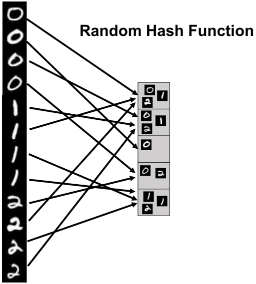
$$\Pr(MinHash(A) = MinHash(B)) = J(A, B).$$

- An instance of locality sensitive hashing (LSH).
- A hash function where the collision probability is higher when two inputs are more similar (can design different functions for different similarity metrics.)

Upshot: MinHash reduces estimating the Jaccard similarity to checking equality of a *single number*.

$$\Pr(MinHash(A) = MinHash(B)) = J(A, B).$$

- An instance of locality sensitive hashing (LSH).
- A hash function where the collision probability is higher when two inputs are more similar (can design different functions for different similarity metrics.)



$Pr(h(A) = h(B) = i) = 1/n^2 \rightarrow Pr(h(A) = h(B)) = 1/n$

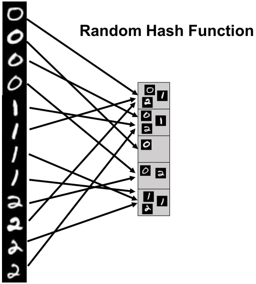
Upshot: MinHash reduces estimating the Jaccard similarity to checking equality of a *single number*.

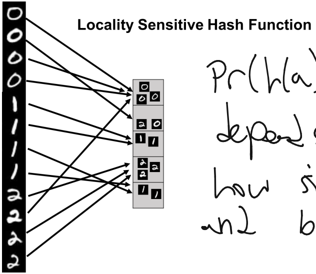
$$\Pr(MinHash(A) = MinHash(B)) = J(A, B).$$

Pairwise independent.

is it LSH?

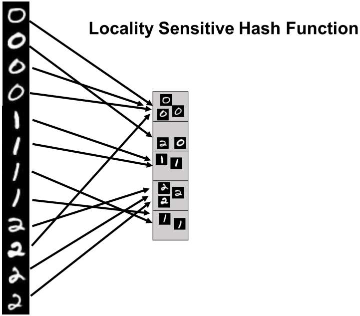
- An instance of locality sensitive hashing (LSH).
- A hash function where the collision probability is higher when two inputs are more similar (can design different functions for different similarity metrics.)



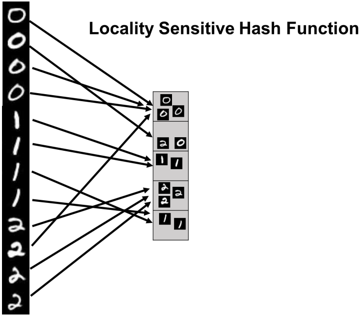


$Pr(h(a) = h(b))$ depends on how similar a and b are

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



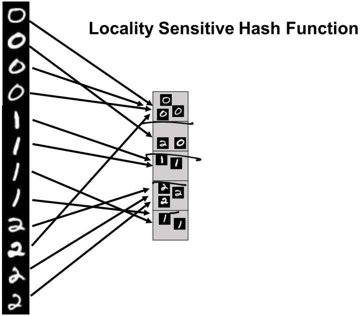
How does locality sensitive hashing (LSH) help with similarity search?



MinHash(x)

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

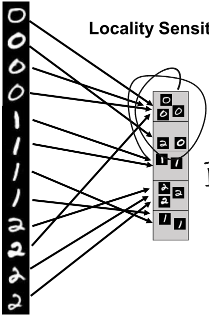
How does locality sensitive hashing (LSH) help with similarity search?



- **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.

LSH FOR SIMILARITY SEARCH

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



Is h LSH?

$Pr(h(A) = h(B))$

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

- **Near Neighbor Search:** Given item x, compute $\mathbf{h}(x)$. Only search for similar items in the $\mathbf{h}(x)$ 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.
- We will use $\mathbf{h}(x) = \mathbf{g}(MinHash(x))$ where $\mathbf{g} : [0, 1] \rightarrow [n]$ is a random hash function. Why?

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 $\mathbf{g} : [0, 1] \to [m]$, and insert every item x into bucket $\mathbf{g}(MinHash(x))$. Search for items similar to y in bucket $\mathbf{g}(MinHash(y))$.

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? $\geq 1/2$

LSH 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?
- For every document x in your database with $J(x, y) \geq 1/2$ what is the probability you will find x in bucket $\mathbf{g}(MinHash(y))$? 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?

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?

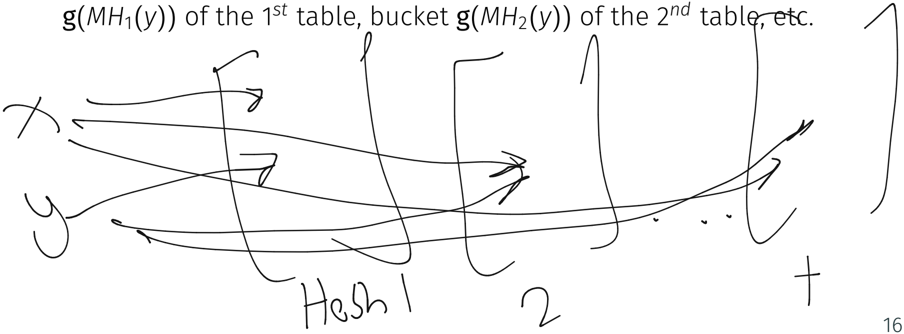
Repetition: Run MinHash t times independently, to produce hash values $MH_1(x), \ldots, MH_t(x)$. Apply random hash function $\mathbf{g}$ to map all these values to locations in t hash tables.

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.



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?

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)

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) $= 1 - \left(\frac{1}{2}\right)^t$

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.

$g_1, g_2, \ldots, g_t$

- 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?

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

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.

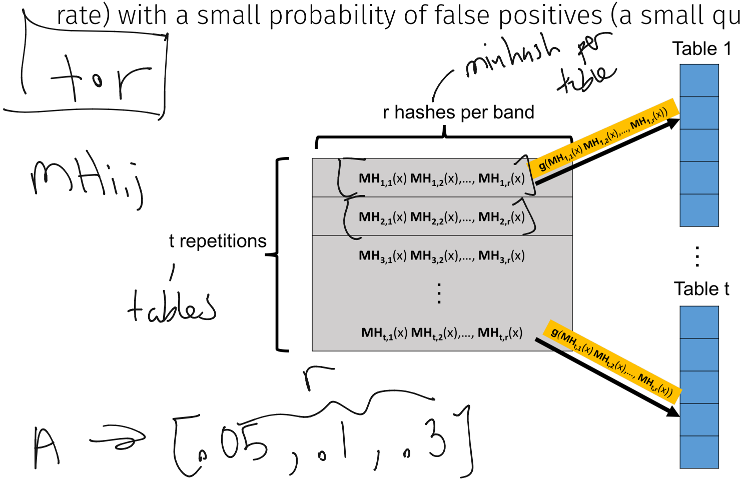
- To search for items similar to y, look at all items in bucket $\mathbf{g}(MH_1(y))$ of the 1^{st} table, bucket $\mathbf{g}(MH_2(y))$ of the 2^{nd} table, etc.
- What is the probability that x with $J(x,y) = 1/2$ is in at least one of these buckets, assuming for simplicity $\mathbf{g}$ has no collisions?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

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.

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

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.
  $\Pr\left[MH_{i,j}(x) = MH_{i,j}(y)\right] = J(x, y) = s$.

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.
  $\Pr\left[MH_{i,1}(x), \ldots, MH_{i,r}(x) = MH_{i,1}(y), \ldots, MH_{i,r}(y)\right] = s^r$

[.1, .3, .05]   [.1, .3, .05]

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.
  $\Pr\left[MH_{i,1}(x), \ldots, MH_{i,r}(x) = MH_{i,1}(y), \ldots, MH_{i,r}(y)\right] = s^r$.

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:

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: $1 - s^r$.

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*:

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:

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

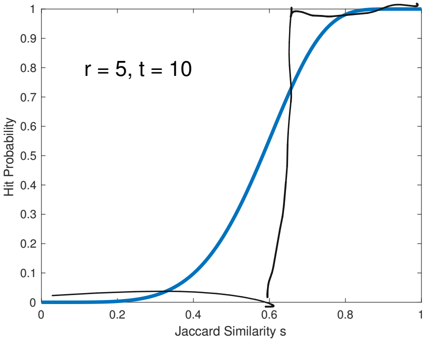
- Probability that a single hash matches.
  $\Pr\left[MH_{i,j}(x) = MH_{i,j}(y)\right] = J(x, y) = s$.
- Probability that x and y having matching signatures in repetition i.
  $\Pr\left[MH_{i,1}(x), \ldots, MH_{i,r}(x) = MH_{i,1}(y), \ldots, MH_{i,r}(y)\right] = s^r$.
- Probability that x and y don't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

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

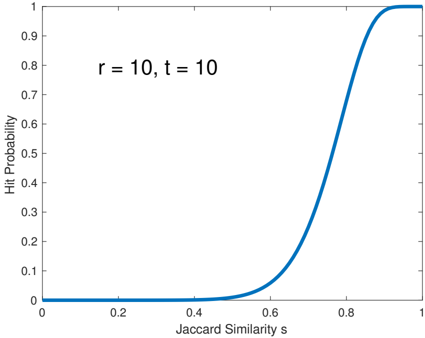
THE s-CURVE

Using t repetitions each with a signature of r MinHash values, the probability that x and y with Jaccard similarity $J(x, y) = s$ match in at least one repetition is: $1 - (1 - s^r)^t$.

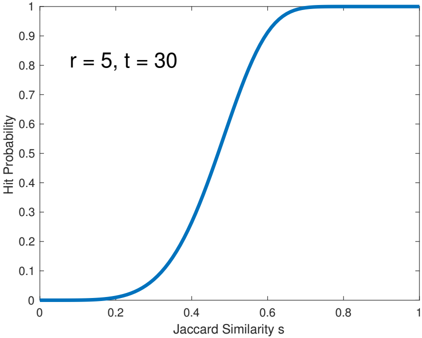
Using t repetitions each with a signature of r MinHash values, the probability that x and y with Jaccard similarity $J(x, y) = s$ match in at least one repetition is: $1 - (1 - s^r)^t$.



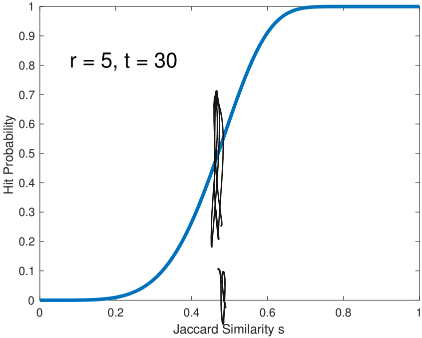
Using t repetitions each with a signature of r MinHash values, the probability that x and y with Jaccard similarity $J(x, y) = s$ match in at least one repetition is: $1 - (1 - s^r)^t$.



Using t repetitions each with a signature of r MinHash values, the probability that x and y with Jaccard similarity $J(x,y) = s$ match in at least one repetition is: $1-(1-s^r)^t$.



Using t repetitions each with a signature of r MinHash values, the probability that x and y with Jaccard similarity $J(x, y) = s$ match in at least one repetition is: $1 - (1 - s^r)^t$.



r and t are tuned depending on application. 'Threshold' when hit probability is 1/2 is $\approx (1/t)^{1/r}$. E.g., $\approx (1/30)^{1/5} = .51$ in this case.

For example: Consider a database with $10,000,000$ audio clips. You are given a clip x and want to find any y in the database with $J(x, y) \geq .9$.

For example: Consider a database with $10,000,000$ audio clips. You are given a clip x and want to find any y in the database with $J(x, y) \geq .9$.

- There are 10 true matches in the database with $J(x, y) \geq .9$.
- There are $10,000$ near matches with $J(x, y) \in [.7, .9]$.

For example: Consider a database with $10,000,000$ audio clips. You are given a clip x and want to find any y in the database with $J(x,y) \geq .9$.

- There are 10 true matches in the database with $J(x,y) \geq .9$.
- There are $10,000$ near matches with $J(x,y) \in [.7,.9]$.

With signature length $r = 25$ and repetitions $t = 50$, hit probability for $J(x,y) = s$ is $1-(1-s^{25})^{50}$.

For example: Consider a database with $10,000,000$ audio clips. You are given a clip x and want to find any y in the database with $J(x,y) \geq .9$.

- There are 10 true matches in the database with $J(x,y) \geq .9$.
- There are $10,000$ near matches with $J(x,y) \in [.7,.9]$.

With signature length $r = 25$ and repetitions $t = 50$, hit probability for $J(x,y) = s$ is $1 - (1 - s^{25})^{50}$.

- Hit probability for $J(x,y) \geq .9$ is $\geq 1 - (1 - .9^{20})^{40} \approx .98$
- Hit probability for $J(x,y) \in [.7,.9]$ is $\leq 1 - (1 - .9^{20})^{40} \approx .98$
- Hit probability for $J(x,y) \leq .7$ is $\leq 1 - (1 - .7^{20})^{40} \approx .007$

s-CURVE EXAMPLE

For example: Consider a database with $10,000,000$ audio clips. You are given a clip x and want to find any y in the database with $J(x, y) \geq .9$.

- There are 10 true matches in the database with $J(x, y) \geq .9$.
- There are $10,000$ near matches with $J(x, y) \in [.7, .9]$.

With signature length $r = 25$ and repetitions $t = 50$, hit probability for $J(x, y) = s$ is $1 - (1 - s^{25})^{50}$.

- Hit probability for $J(x, y) \geq .9$ is $\geq 1 - (1 - .9^{20})^{40} \approx .98$
- Hit probability for $J(x, y) \in [.7, .9]$ is $\leq 1 - (1 - .9^{20})^{40} \approx .98$
- Hit probability for $J(x, y) \leq .7$ is $\leq 1 - (1 - .7^{20})^{40} \approx .007$

Expected Number of Items Scanned: (proportional to query time)

$$\leq 10 + .98 * 10,000 + .007 * 9,989,990 \approx 80,000$$

For example: Consider a database with $10,000,000$ audio clips. You are given a clip x and want to find any y in the database with $J(x,y) \geq .9$.

- There are 10 true matches in the database with $J(x,y) \geq .9$.
- There are $10,000$ near matches with $J(x,y) \in [.7,.9]$.

With signature length $r = 25$ and repetitions $t = 50$, hit probability for $J(x,y) = s$ is $1-(1-s^{25})^{50}$.

- Hit probability for $J(x,y) \geq .9$ is $\geq 1-(1-.9^{20})^{40} \approx .98$
- Hit probability for $J(x,y) \in [.7,.9]$ is $\leq 1-(1-.9^{20})^{40} \approx .98$
- Hit probability for $J(x,y) \leq .7$ is $\leq 1-(1-.7^{20})^{40} \approx .007$

Expected Number of Items Scanned: (proportional to query time)

$$\leq 10 + .98 * 10,000 + .007 * 9,989,990 \approx 80,000 \ll 10,000,000.$$

HASHING FOR DUPLICATE DETECTION

	Bloom Filters	Hash Table	MinHash	Distinct Elements
Goal	Check if x is a duplicate of y in database.	Check if x is a duplicate of any y in database and return y.	Check if x is a duplicate of any y in database and return y.	Count # of items, excluding duplicates.
Approximate Duplicates?	✗	✗	✓	✗

	Bloom Filters	Hash Table	MinHash	Distinct Elements
Goal	Check if x is a duplicate of y in database.	Check if x is a duplicate of any y in database and return y.	Check if x is a duplicate of any y in database and return y.	Count # of items, excluding duplicates.
Approximate Duplicates?	✗	✗	✓	✗

All different variants of detecting duplicates/finding matches in large datasets. This is an important problem in many contexts!