COMPSCI 514: ALGORITHMS FOR DATA SCIENCE

Andrew McGregor
Lecture 8
Jaccard Similarity: \[ J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{\text{# shared elements}}{\text{# total elements}}. \]

Two Common Use Cases:

- **Near Neighbor Search**: Have a database of \( n \) sets/bit strings and given a set \( A \), want to find if it has high similarity to anything in the database. Naively \( \Omega(n) \) time.

- **All-pairs Similarity Search**: Have \( n \) different sets/bit strings. Want to find all pairs with high similarity. Naively \( \Omega(n^2) \) time.
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.
**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 \( \binom{1000000}{2} \approx 500 \text{ billion pairs of customers to check!} \)
Many applications to spam/fraud detection. E.g.
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.
In 1997, Andrei Broder at Altavista proposed MinHash:

\[ \text{MinHash}(A) = \min_{a \in A} h(a) \text{ where } h : U \rightarrow [0, 1] \text{ is random.} \]

For two sets \( A \) and \( B \), what is \( \Pr(\text{MinHash}(A) = \text{MinHash}(B)) \)?
In 1997, Andrei Broder at Altavista proposed MinHash:

\[ MinHash(A) = \min_{a \in A} h(a) \text{ where } h : U \to [0, 1] \text{ is random}. \]

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

\( \text{MinHash}(A) = \text{MinHash}(B) \) iff an item \( A \cap B \) takes min value.
In 1997, Andrei Broder at Altavista proposed MinHash:

\[ \text{MinHash}(A) = \min_{a \in A} h(a) \text{ where } h : U \rightarrow [0, 1] \text{ is random.} \]

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

\( \text{MinHash}(A) = \text{MinHash}(B) \) iff an item \( A \cap B \) takes min value.

\[ \Pr(\text{MinHash}(A) = \text{MinHash}(B)) = ? \]
In 1997, Andrei Broder at Altavista proposed MinHash:

$$MinHash(A) = \min_{a \in A} h(a) \text{ where } h : U \to [0, 1] \text{ is random.}$$

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

$MinHash(A) = MinHash(B)$ iff an item $A \cap B$ takes min value.

$$\Pr(\text{MinHash}(A) = \text{MinHash}(B)) = \frac{|A \cap B|}{\text{total \# items hashed}}$$
In 1997, Andrei Broder at Altavista proposed MinHash:

\[ \text{MinHash}(A) = \min_{a \in A} h(a) \text{ where } h : U \rightarrow [0, 1] \text{ is random.} \]

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

\( \text{MinHash}(A) = \text{MinHash}(B) \) iff an item \( A \cap B \) takes min value.

\[
\Pr(\text{MinHash}(A) = \text{MinHash}(B)) = \frac{|A \cap B|}{\text{total } \# \text{ items hashed}} = \frac{|A \cap B|}{|A \cup B|}
\]
**Upshot:** MinHash reduces estimating the Jaccard similarity to checking equality of a *single number*.

\[
\Pr(\text{MinHash}(A) = \text{MinHash}(B)) = J(A, B).
\]
**Upshot:** MinHash reduces estimating the Jaccard similarity to checking equality of a *single number*.

\[
\Pr(\text{MinHash}(A) = \text{MinHash}(B)) = J(A, B).
\]

- An instance of **locality sensitive hashing** (LSH).
**Upshot:** MinHash reduces estimating the Jaccard similarity to checking equality of a *single number*.

\[
\Pr(\text{MinHash}(A) = \text{MinHash}(B)) = J(A, B).
\]

- 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*.

\[
Pr(\text{MinHash}(A) = \text{MinHash}(B)) = J(A, B).
\]

- 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.*

\[
\text{Pr}(\text{MinHash}(A) = \text{MinHash}(B)) = J(A, B).
\]

- 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.)
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.

We will use $h(x) = g(\text{MinHash}(x))$ where $g: [0,1] \rightarrow \mathbb{N}$ is a random hash function. Why?
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.
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.
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.
- We will use \( h(x) = g(\text{MinHash}(x)) \) where \( g : [0, 1] \rightarrow [n] \) 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 $g : [0, 1] \rightarrow [m]$, and insert every item $x$ into bucket $g(\text{MinHash}(x))$. Search for items similar to $y$ in bucket $g(\text{MinHash}(y))$. 

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

- What is $\Pr [g(\text{MinHash}(x)) = g(\text{MinHash}(y))]$ assuming $J(x, y) = 1/2$ and $g$ is collision free?
**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 $g : [0, 1] \rightarrow [m]$, and insert every item $x$ into bucket $g(\text{MinHash}(x))$. Search for items similar to $y$ in bucket $g(\text{MinHash}(y))$.

- What is $\Pr [g(\text{MinHash}(x)) = g(\text{MinHash}(y))]$ assuming $J(x, y) = 1/2$ and $g$ 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 $g(\text{MinHash}(y))$?
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 $g$ to map all these values to locations in $t$ hash tables.
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 \( g \) to map all these values to locations in \( t \) hash tables.

- To search for items similar to \( y \), look at all items in bucket \( g(MH_1(y)) \) of the 1\(^{st} \) table, bucket \( g(MH_2(y)) \) 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 $g$ to map all these values to locations in $t$ hash tables.

- To search for items similar to $y$, look at all items in bucket $g(MH_1(y))$ of the $1^{st}$ table, bucket $g(MH_2(y))$ 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 $g$ 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 $g$ to map all these values to locations in $t$ hash tables.

- To search for items similar to $y$, look at all items in bucket $g(MH_1(y))$ of the $1^{st}$ table, bucket $g(MH_2(y))$ 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 $g$ has no collisions?

$$1 - \left(\text{probability in no buckets}\right) \approx 0.99$$ for $t = 7$.

Potential for a lot of false positives! Slows down search time.
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 \( g \) to map all these values to locations in \( t \) hash tables.

- To search for items similar to \( y \), look at all items in bucket \( g(MH_1(y)) \) of the 1\(^{st} \) table, bucket \( g(MH_2(y)) \) 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 \( g \) has no collisions?

\[
1 - \left( \text{probability in no buckets} \right) = 1 - \left( \frac{1}{2} \right)^t
\]
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 \( g \) to map all these values to locations in \( t \) hash tables.

- To search for items similar to \( y \), look at all items in bucket \( g(MH_1(y)) \) of the 1\(^{st} \) table, bucket \( g(MH_2(y)) \) 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 \( g \) has no collisions?
  \[ 1 - \text{(probability in no buckets)} = 1 - \left(\frac{1}{2}\right)^t \approx .99 \text{ for } t = 7. \]
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 \( g \) to map all these values to locations in \( t \) hash tables.

- To search for items similar to \( y \), look at all items in bucket \( g(MH_1(y)) \) of the 1\(^{st} \) table, bucket \( g(MH_2(y)) \) 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 \( g \) has no collisions?
  
  \[
  1 - \left( \frac{1}{2} \right)^t \approx .99 \text{ for } t = 7.
  \]

- What is the probability that \( x \) with \( J(x, y) = 1/4 \) is in at least one of these buckets, assuming for simplicity \( g \) 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 $g$ to map all these values to locations in $t$ hash tables.

- To search for items similar to $y$, look at all items in bucket $g(MH_1(y))$ of the $1^{st}$ table, bucket $g(MH_2(y))$ 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 $g$ has no collisions? $1 – \left(\text{probability in } no \text{ buckets}\right) = 1 – \left(\frac{1}{2}\right)^t \approx .99$ 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 $g$ has no collisions? $1 – \left(\text{probability in } no \text{ buckets}\right) = 1 – \left(\frac{3}{4}\right)^t$
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 $g$ to map all these values to locations in $t$ hash tables.

- To search for items similar to $y$, look at all items in bucket $g(MH_1(y))$ of the 1$^{st}$ table, bucket $g(MH_2(y))$ 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 $g$ has no collisions? 
  \[ 1 - \text{(probability in no buckets)} = 1 - \left( \frac{1}{2} \right)^t \approx .99 \text{ for } t = 7. \]

- What is the probability that $x$ with $J(x, y) = 1/4$ is in at least one of these buckets, assuming for simplicity $g$ has no collisions? 
  \[ 1 - \text{(probability in no buckets)} = 1 - \left( \frac{3}{4} \right)^t \approx .87 \text{ for } t = 7. \]
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 \( g \) to map all these values to locations in \( t \) hash tables.

- To search for items similar to \( y \), look at all items in bucket \( g(MH_1(y)) \) of the 1\(^{st} \) table, bucket \( g(MH_2(y)) \) 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 \( g \) has no collisions?
  \[
  1 - \text{(probability in no buckets)} = 1 - \left( \frac{1}{2} \right)^t \approx .99 \text{ for } t = 7.
  \]

- What is the probability that \( x \) with \( J(x, y) = 1/4 \) is in at least one of these buckets, assuming for simplicity \( g \) has no collisions?
  \[
  1 - \text{(probability in no buckets)} = 1 - \left( \frac{3}{4} \right)^t \approx .87 \text{ for } t = 7.
  \]

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

Table 1

Table 2

Table t
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: 
\[ \Pr[MH_{i,j}(x) = MH_{i,j}(y)] = J(x, y) = s. \]

- Probability that $x$ and $y$ having matching signatures in repetition $i$:
\[ \Pr[MH_{i,1}(x), \ldots, MH_{i,r}(x) = MH_{i,1}(y), \ldots, MH_{i,r}(y)] = s^r. \]

- 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:
\[ \text{Hit Probability: } 1 - (1 - s^r)^t. \]
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.
  $\Pr [MH_{i,j}(x) = MH_{i,j}(y)] = J(x, y) = s$. 

• Probability that $x$ and $y$ having matching signatures in repetition $i$.
  $\Pr [MH_{i,1}(x), \ldots, MH_{i,r}(x) = MH_{i,1}(y), \ldots, MH_{i,r}(y)] = s^r$.

• 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:
  Hit Probability: $1 - (1 - s^r)^t$. 

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

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

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

- Probability that $x$ and $y$ having matching signatures in repetition $i$.
  \[
  \Pr [MH_{i,1}(x), \ldots, MH_{i,r}(x) = MH_{i,1}(y), \ldots, MH_{i,r}(y)] = s^r.
  \]

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

- Probability that $x$ and $y$ having matching signatures in repetition $i$.
  \[ \Pr [MH_{i,1}(x), \ldots, MH_{i,r}(x) = MH_{i,1}(y), \ldots, MH_{i,r}(y)] = s^r. \]

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

- Probability that $x$ and $y$ having matching signatures in repetition $i$. 
  $$\Pr [MH_{i,1}(x), \ldots, MH_{i,r}(x) = MH_{i,1}(y), \ldots, MH_{i,r}(y)] = s^r.$$ 

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

- Probability that \( x \) and \( y \) having matching signatures in repetition \( i \).
  \[
  \Pr [MH_{i,1}(x), \ldots, MH_{i,r}(x) = MH_{i,1}(y), \ldots, MH_{i,r}(y)] = s^r.
  \]

- 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: \( (1 - s^r)^t \).
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.
  \[
  \Pr[\text{MH}_{i,j}(x) = \text{MH}_{i,j}(y)] = J(x, y) = s.
  \]

- Probability that $x$ and $y$ having matching signatures in repetition $i$.
  \[
  \Pr[\text{MH}_{i,1}(x), \ldots, \text{MH}_{i,r}(x) = \text{MH}_{i,1}(y), \ldots, \text{MH}_{i,r}(y)] = s^r.
  \]

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

- Probability that \( x \) and \( y \) having matching signatures in repetition \( i \).
  \[
  \Pr [MH_{i,1}(x), \ldots, MH_{i,r}(x) = MH_{i,1}(y), \ldots, MH_{i,r}(y)] = s^r.
  \]

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

![Graph showing the s-curve with hit probability and Jaccard similarity]
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: $1 - (1 - s^r)^t$. 

![Graph](#)
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: $1 - (1 - s^r)^t$. 

![Hit Probability Graph](image)

$r = 5, t = 30$
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 \( J(x, y) \geq 0.9 \).
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 $\leq 1 - (1 - .7^{20})^{40} \approx .007$
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)

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

\[
\leq 10 + .98 \times 10,000 + .007 \times 9,989,990 \approx 80,000 \ll 10,000,000.
\]
Repetition and s-curve tuning can be used for fast similarity search with any similarity metric, given a locality sensitive hash function for that metric.

Cosine Similarity:

\[
\cos(\theta(x, y)) = \frac{\langle x, y \rangle}{\|x\|_2 \cdot \|y\|_2}.
\]

\(\cos(\theta(x, y)) = 1\) when \(\theta(x, y) = 0^\circ\) and \(\cos(\theta(x, y)) = 0\) when \(\theta(x, y) = 90^\circ\), and \(\cos(\theta(x, y)) = -1\) when \(\theta(x, y) = 180^\circ\).
Repetition and s-curve tuning can be used for fast similarity search with any similarity metric, given a locality sensitive hash function for that metric.

- LSH schemes exist for many similarity/distance measures: hamming distance, cosine similarity, etc.
Repetition and $s$-curve tuning can be used for fast similarity search with any similarity metric, given a locality sensitive hash function for that metric.

- LSH schemes exist for many similarity/distance measures: hamming distance, cosine similarity, etc.
Repetition and s-curve tuning can be used for fast similarity search with any similarity metric, given a locality sensitive hash function for that metric.

- LSH schemes exist for many similarity/distance measures: hamming distance, cosine similarity, etc.
Repetition and s-curve tuning can be used for fast similarity search with any similarity metric, given a locality sensitive hash function for that metric.

- LSH schemes exist for many similarity/distance measures: hamming distance, cosine similarity, etc.

\[ \cos(\theta(x, y)) = \begin{cases} 1 & \text{when } \theta(x, y) = 0^\circ, \\ 0 & \text{when } \theta(x, y) = 90^\circ, \\ -1 & \text{when } \theta(x, y) = 180^\circ. \end{cases} \]

**Cosine Similarity:** \( \cos(\theta(x, y)) \)
Repetition and $s$-curve tuning can be used for fast similarity search with any similarity metric, given a locality sensitive hash function for that metric.

- LSH schemes exist for many similarity/distance measures: hamming distance, cosine similarity, etc.

**Cosine Similarity**: $\cos(\theta(x, y))$

- $\cos(\theta(x, y)) = 1$ when $\theta(x, y) = 0^\circ$ and $\cos(\theta(x, y)) = 0$ when $\theta(x, y) = 90^\circ$, and $\cos(\theta(x, y)) = -1$ when $\theta(x, y) = 180^\circ$
Repetition and $s$-curve tuning can be used for fast similarity search with any similarity metric, given a locality sensitive hash function for that metric.

- LSH schemes exist for many similarity/distance measures: hamming distance, cosine similarity, etc.

**Cosine Similarity:**
\[
\cos(\theta(x, y)) = \frac{\langle x, y \rangle}{\|x\|_2 \cdot \|y\|_2}.
\]

- $\cos(\theta(x, y)) = 1$ when $\theta(x, y) = 0^\circ$ and $\cos(\theta(x, y)) = 0$ when $\theta(x, y) = 90^\circ$, and $\cos(\theta(x, y)) = -1$ when $\theta(x, y) = 180^\circ$
SimHash Algorithm: LSH for cosine similarity.
SimHash Algorithm: LSH for cosine similarity.
SimHash Algorithm: LSH for cosine similarity.
SimHash Algorithm: LSH for cosine similarity.

SimHash \( (x) = \text{sign} \langle x, t \rangle \) for a random vector \( t \).

What is \( \Pr \left[ \text{SimHash} (x) = \text{SimHash} (y) \right] \)?
SimHash Algorithm: LSH for cosine similarity.

\[
\text{SimHash}(x) = \text{sign}(\langle x, t \rangle) \quad \text{for a random vector } t.
\]

random plane

\[
\text{SimHash}(x) = 1
\]

\[
\text{SimHash}(x) = -1
\]
SimHash Algorithm: LSH for cosine similarity.

SimHash \( (x) = \text{sign}(\langle x, t \rangle) \) for a random vector \( t \).

What is \( \Pr [\text{SimHash}(x) = \text{SimHash}(y)] \)?
What is \( \Pr [\text{SimHash}(x) = \text{SimHash}(y)] \)?
What is $\Pr [\text{SimHash}(x) = \text{SimHash}(y)]$?

$\text{SimHash}(x) \neq \text{SimHash}(y)$ when the plane separates $x$ from $y$. 
What is $\Pr [\text{SimHash}(x) = \text{SimHash}(y)]$?

$\text{SimHash}(x) \neq \text{SimHash}(y)$ when the plane separates $x$ from $y$. 

SimHash(x) = 1

SimHash(y) = -1
What is $\Pr[\text{SimHash}(x) = \text{SimHash}(y)]$?

$\text{SimHash}(x) \neq \text{SimHash}(y)$ when the plane separates $x$ from $y$.

- $\Pr[\text{SimHash}(x) \neq \text{SimHash}(y)] = \frac{\theta(x,y)}{\pi}$
What is $\text{Pr} \left[ \text{SimHash}(x) = \text{SimHash}(y) \right]$?

$\text{SimHash}(x) \neq \text{SimHash}(y)$ when the plane separates $x$ from $y$.

- $\text{Pr} \left[ \text{SimHash}(x) \neq \text{SimHash}(y) \right] = \frac{\theta(x,y)}{\pi}$
- $\text{Pr} \left[ \text{SimHash}(x) = \text{SimHash}(y) \right] = 1 - \frac{\theta(x,y)}{\pi} \approx \frac{\cos(\theta(x,y))+1}{2}$
Questions on MinHash and Locality Sensitive Hashing?