DATA MINING
LECTURE 5
Similarity and Distance
Sketching, Locality Sensitive Hashing
SIMILARITY AND DISTANCE

Thanks to:
Tan, Steinbach, and Kumar, “Introduction to Data Mining”
Rajaraman and Ullman, “Mining Massive Datasets”
Similarity and Distance

For many different problems we need to quantify how close two objects are.

Examples:
- For an item bought by a customer, find other similar items
- Group together the customers of site so that similar customers are shown the same ad.
- Group together web documents so that you can separate the ones that talk about politics and the ones that talk about sports.
- Find all the near-duplicate mirrored web documents.
- Find credit card transactions that are very different from previous transactions.

To solve these problems we need a definition of similarity, or distance.
- The definition depends on the type of data that we have
Similarity

- Numerical measure of how alike two data objects are.
  - A function that maps pairs of objects to real values
  - Higher when objects are more alike.
- Often falls in the range $[0,1]$, sometimes in $[-1,1]$

Desirable properties for similarity

1. $s(p, q) = 1$ (or maximum similarity) only if $p = q$. (*Identity*)
2. $s(p, q) = s(q, p)$ for all $p$ and $q$. (*Symmetry*)
Similarity between sets

- Consider the following documents

  apple releases new ipod
  apple releases new ipad
  new apple pie recipe

- Which ones are more similar?

- How would you quantify their similarity?
Similarity: Intersection

- Number of words in common

  apple releases new ipod
  apple releases new ipad
  new apple pie recipe

- Sim(D,D) = 3, Sim(D,D) = Sim(D,D) = 2

- What about this document?

  Vefa rereases new book with apple pie recipes

- Sim(D,D) = Sim(D,D) = 3
Jaccard Similarity

- The Jaccard similarity (Jaccard coefficient) of two sets $S_1$, $S_2$ is the size of their intersection divided by the size of their union.
  - $\text{JSim} (C_1, C_2) = \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|}$.

- Extreme behavior:
  - $\text{Jsim}(X,Y) = 1$, iff $X = Y$
  - $\text{Jsim}(X,Y) = 0$ iff $X,Y$ have not elements in common
  - JSim is symmetric
Similarity: Intersection

- Number of words in common

apple releases new ipod  apple releases new ipad  new apple pie recipe

\[ \text{JSim}(D,D) = 3/5 \]
\[ \text{JSim}(D,D) = \text{JSim}(D,D) = 2/6 \]
\[ \text{JSim}(D,D) = \text{JSim}(D,D) = 3/9 \]
Similarity between vectors

Documents (and sets in general) can also be represented as vectors.

<table>
<thead>
<tr>
<th>document</th>
<th>Apple</th>
<th>Microsoft</th>
<th>Obama</th>
<th>Election</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>10</td>
<td>20</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D2</td>
<td>30</td>
<td>60</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D2</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>20</td>
</tr>
</tbody>
</table>

How do we measure the similarity of two vectors?

How well are the two vectors aligned?
Documents D1, D2 are in the “same direction”
Document D3 is orthogonal to these two
Cosine Similarity

- \( \text{Sim}(X,Y) = \cos(X,Y) \)
  - The cosine of the angle between \( X \) and \( Y \)

- If the vectors are aligned (correlated) angle is zero degrees and \( \cos(X,Y)=1 \)
- If the vectors are orthogonal (no common coordinates) angle is 90 degrees and \( \cos(X,Y) = 0 \)

- Cosine is commonly used for comparing documents, where we assume that the vectors are normalized by the document length.

Figure 2.16. Geometric illustration of the cosine measure.
Cosine Similarity - math

- If $d_1$ and $d_2$ are two vectors, then
  \[
  \cos( d_1, d_2 ) = \frac{d_1 \cdot d_2}{\|d_1\| \|d_2\|},
  \]
  where $\cdot$ indicates vector dot product and $\| d \|$ is the length of vector $d$.

- Example:

  \[
  d_1 = 3 2 0 5 0 0 0 2 0 0 \\
  d_2 = 1 0 0 0 0 0 0 1 0 2
  \]

  \[
  d_1 \cdot d_2 = 3*1 + 2*0 + 0*0 + 5*0 + 0*0 + 0*0 + 0*0 + 2*1 + 0*0 + 0*2 = 5
  \]

  \[
  \|d_1\| = (3^2+2^2+0^2+0^2+5^2+0^2+0^2+0^2+0^2+0^2)^{0.5} = (42)^{0.5} = 6.481
  \]

  \[
  \|d_2\| = (1^2+0^2+0^2+0^2+0^2+0^2+0^2+0^2+1^2+0^2+2^2)^{0.5} = (6)^{0.5} = 2.245
  \]

  \[
  \cos( d_1, d_2 ) = .3150
  \]
Similarity between vectors

\[
\cos(D1,D2) = 1 \\
\cos(D1,D3) = \cos(D2,D3) = 0
\]

<table>
<thead>
<tr>
<th>document</th>
<th>Apple</th>
<th>Microsoft</th>
<th>Obama</th>
<th>Election</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>10</td>
<td>20</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D2</td>
<td>30</td>
<td>60</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D2</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>20</td>
</tr>
</tbody>
</table>
Distance

- Numerical measure of how different two data objects are
  - A function that maps pairs of objects to real values
  - Lower when objects are more alike
- Minimum distance is 0, when comparing an object with itself.
- Upper limit varies
Distance Metric

A distance function $d$ is a distance metric if it is a function from pairs of objects to real numbers such that:

1. $d(x,y) \geq 0$. (non-negativity)
2. $d(x,y) = 0$ iff $x = y$. (identity)
3. $d(x,y) = d(y,x)$. (symmetry)
4. $d(x,y) \leq d(x,z) + d(z,y)$ (triangle inequality).
Triangle Inequality

- Triangle inequality guarantees that the distance function is well-behaved.
  - The direct connection is the shortest distance

- It is useful also for proving properties about the data
  - For example, suppose I want to find an object that minimizes the sum of distances to all points in my dataset
  - If I select the best point from my dataset, the sum of distances I get is at most twice that of the optimal point.
Distances for real vectors

- Vectors $x = (x_1, ..., x_d)$ and $y = (y_1, ..., y_d)$

- $L_p$ norms or Minkowski distance:
  $$L_p(x, y) = \left[ |x_1 - y_1|^p + \cdots + |x_d - y_d|^p \right]^{1/p}$$

- $L_2$ norm: Euclidean distance:
  $$L_2(x, y) = \sqrt{|x_1 - y_1|^2 + \cdots + |x_d - y_d|^2}$$

- $L_1$ norm: Manhattan distance:
  $$L_1(x, y) = |x_1 - y_1| + \cdots + |x_d - y_d|$$

- $L_\infty$ norm:
  $$L_\infty(x, y) = \max\{|x_1 - y_1|, \ldots, |x_d - y_d|\}$$

- The limit of $L_p$ as $p$ goes to infinity.
Example of Distances

L₂-norm:
\[ \text{dist}(x, y) = \sqrt{4^2 + 3^2} = 5 \]

L₁-norm:
\[ \text{dist}(x, y) = 4 + 3 = 7 \]

Lₘₐₓ-norm:
\[ \text{dist}(x, y) = \max\{3, 4\} = 4 \]
Example

\[ x = (x_1, \ldots, x_n) \]

Green: All points y at distance \( L_1(x, y) = r \) from point x

Blue: All points y at distance \( L_2(x, y) = r \) from point x

Red: All points y at distance \( L_\infty(x, y) = r \) from point x
L_p distances for sets

- We can apply all the L_p distances to the cases of sets of attributes, with or without counts, if we represent the sets as vectors
  - E.g., a transaction is a 0/1 vector
  - E.g., a document is a vector of counts.
Similarities into distances

- Jaccard distance:
  \[ JDist(X,Y) = 1 - JSim(X,Y) \]

- Jaccard Distance is a metric

- Cosine distance:
  \[ Dist(X,Y) = 1 - \cos(X,Y) \]

- Cosine distance is a metric
Why Jaccard Distance Is a Distance Metric

- \( \text{JDist}(x,x) = 0 \)
  - since \( \text{JSim}(x,x) = 1 \)
- \( \text{JDist}(x,y) = \text{JDist}(y,x) \)
  - by symmetry of intersection
- \( \text{JDist}(x,y) \geq 0 \)
  - since intersection of \( X,Y \) cannot be bigger than the union.
- **Triangle inequality:**
  - Follows from the fact that \( \text{JSim}(X,Y) \) is the probability of randomly selected element from the union of \( X \) and \( Y \) to belong to the intersection
Hamming Distance

- **Hamming distance** is the number of positions in which bit-vectors differ.
  - **Example**: \( p_1 = 10101 \)
    \[ p_2 = 10011. \]
    - \( d(p_1, p_2) = 2 \) because the bit-vectors differ in the 3\(^{rd}\) and 4\(^{th}\) positions.
    - The \( L_1 \) norm for the binary vectors

- **Hamming distance** between two vectors of categorical attributes is the number of positions in which they differ.
  - **Example**: \( x = (\text{married, low income, cheat}) \), \( y = (\text{single, low income, not cheat}) \)
    - \( d(x,y) = 2 \)
Why Hamming Distance Is a Distance Metric

- $d(x,x) = 0$ since no positions differ.
- $d(x,y) = d(y,x)$ by symmetry of “different from.”
- $d(x,y) \geq 0$ since strings cannot differ in a negative number of positions.
- **Triangle inequality**: changing $x$ to $z$ and then to $y$ is one way to change $x$ to $y$.

- For binary vectors it follows from the fact that $L_1$ norm is a metric.
Distance between strings

• How do we define similarity between strings?

  weird    wierd
  intelligent unintelligent
  Athena    Athina

• Important for recognizing and correcting typing errors and analyzing DNA sequences.
Edit Distance for strings

- The edit distance of two strings is the number of inserts and deletes of characters needed to turn one into the other.
- Example: $x = abcd e$; $y = bcduve$.
  - Turn $x$ into $y$ by deleting $a$, then inserting $u$ and $v$ after $d$.
  - Edit distance = 3.
- Minimum number of operations can be computed using dynamic programming.
- Common distance measure for comparing DNA sequences.
Why Edit Distance Is a Distance Metric

• $d(x, x) = 0$ because 0 edits suffice.
• $d(x, y) = d(y, x)$ because insert/delete are inverses of each other.
• $d(x, y) \geq 0$: no notion of negative edits.
• **Triangle inequality**: changing $x$ to $z$ and then to $y$ is one way to change $x$ to $y$. The minimum is no more than that
Variant Edit Distances

- Allow insert, delete, and **mutate**.
  - Change one character into another.
- Minimum number of inserts, deletes, and mutates also forms a distance measure.

- Same for any set of operations on strings.
  - **Example**: substring reversal or block transposition **OK** for DNA sequences
  - **Example**: character transposition is used for spelling
Distances between distributions

• We can view a document as a distribution over the words

<table>
<thead>
<tr>
<th>document</th>
<th>Apple</th>
<th>Microsoft</th>
<th>Obama</th>
<th>Election</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>0.35</td>
<td>0.5</td>
<td>0.1</td>
<td>0.05</td>
</tr>
<tr>
<td>D2</td>
<td>0.4</td>
<td>0.4</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>D2</td>
<td>0.05</td>
<td>0.05</td>
<td>0.6</td>
<td>0.3</td>
</tr>
</tbody>
</table>

• KL-divergence (Kullback-Leibler) for distributions $P,Q$

$$D_{KL}(P\|Q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

• KL-divergence is asymmetric. We can make it symmetric by taking the average of both sides

• JS-divergence (Jensen-Shannon)

$$JS(P, Q) = \frac{1}{2} D_{KL}(P\|Q) + \frac{1}{2} D_{KL}(Q\|P)$$
SKETCHING
AND
LOCALITY SENSITIVE HASHING

Thanks to:
Rajaraman and Ullman, “Mining Massive Datasets”
Evimaria Terzi, slides for Data Mining Course.
Why is similarity important?

• We saw many definitions of similarity and distance
• How do we make use of similarity in practice?
• What issues do we have to deal with?
An important problem

• Recommendation systems
  • When a user buys an item (initially books) we want to recommend other items that the user may like
  • When a user rates a movie, we want to recommend movies that the user may like
  • When a user likes a song, we want to recommend other songs that they may like

• A big success of data mining
• Exploits the long tail
Recommendation Systems

**Content-based:**
- Represent the items into a *feature space* and recommend items to customer C similar to previous items rated highly by C
  - Movie recommendations: recommend movies with same actor(s), director, genre, …
  - Websites, blogs, news: recommend other sites with “similar” content
Plan of action

Item profiles

likes

match

recommend

build

Red Circles Triangles

User profile
Limitations of content-based approach

- Finding the appropriate features
  - e.g., images, movies, music

- Overspecialization
  - Never recommends items outside user’s content profile
  - People might have multiple interests

- Recommendations for new users
  - How to build a profile?
Recommendation Systems (II)

- Collaborative Filtering (user – user)
  - Consider user c
  - Find set D of other users whose ratings are “similar” to c’s ratings
  - Estimate user’s ratings based on ratings of users in D
Recommendation Systems (III)

- Collaborative filtering (item-item)
  - For items, find other similar items
  - Estimate rating for item based on ratings for similar items
  - Can use same similarity metrics and prediction functions as in user-user model
- In practice, it has been observed that item-item often works better than user-user
Pros and cons of collaborative filtering

- Works for any kind of item
  - No feature selection needed
- New user problem
- New item problem
- Sparsity of rating matrix
  - Cluster-based smoothing?
Another important problem

• Find **duplicate** and **near-duplicate** documents from a web crawl.

• Why is it important:
  • Identify **mirrored web pages**, and avoid indexing them, or serving them multiple times
  • Find **replicated news stories** and cluster them under a single story.
  • Identify plagiarism

• What if we wanted exact duplicates?
Finding similar items

• Both the problems we described have a common component
  • We need a quick way to find highly similar items to a query item
  • OR, we need a method for finding all pairs of items that are highly similar.
• Also known as the Nearest Neighbor problem, or the All Nearest Neighbors problem

• We will examine it for the case of near-duplicate web documents.
Main issues

• What is the right representation of the document when we check for similarity?
  • E.g., representing a document as a set of characters will not do (why?)

• When we have billions of documents, keeping the full text in memory is not an option.
  • We need to find a shorter representation

• How do we do pairwise comparisons of billions of documents?
  • If exact match was the issue it would be ok, can we replicate this idea?
Three Essential Techniques for Similar Documents

1. **Shingling**: convert documents, emails, etc., to sets.

2. **Minhashing**: convert large sets to short signatures, while preserving similarity.

3. **Locality-Sensitive Hashing (LSH)**: focus on pairs of signatures likely to be similar.
The Big Picture

Docu-

ment

Shingling

Minhash-
ing

Locality-
sensitive Hashing

Candidate pairs: those pairs of signatures that we need to test for similarity.

The set of strings of length $k$ that appear in the doc-

ument

Signatures: short integer vectors that represent the sets, and reflect their similarity.
Shingles

- A k-shingle (or k-gram) for a document is a sequence of k characters that appears in the document.
- **Example**: document = abcab. k=2
  - Set of 2-shingles = \{ab, bc, ca\}.
  - **Option**: regard shingles as a **bag**, and count **ab** twice.

- Represent a document by its set of k-shingles.
Shingling

- Shingle: a sequence of $k$ contiguous characters

```
a rose is a rose is a rose
    rose is
      rose is a
        rose is a
         ose is a r
            se is a ro
              e is a ros
                is a rose
                  is a rose
                    s a rose i
                      a rose i
                        a rose is
```
Working Assumption

• Documents that have lots of shingles in common have similar text, even if the text appears in different order.

• **Careful**: you must pick $k$ large enough, or most documents will have most shingles.
  • Extreme case $k = 1$: all documents are the same
  • $k = 5$ is OK for short documents; $k = 10$ is better for long documents.

• Alternative ways to define shingles:
  • Use words instead of characters
  • Anchor on stop words (to avoid templates)
Shingles: Compression Option

• To compress long shingles, we can hash them to (say) 4 bytes.
• Represent a doc by the set of hash values of its $k$-shingles.
• From now on we will assume that shingles are integers
  • Collisions are possible, but very rare
Fingerpointing

- Hash shingles to 64-bit integers

<table>
<thead>
<tr>
<th>Set of Shingles</th>
<th>Hash function (Rabin’s fingerprints)</th>
<th>Set of 64-bit integers</th>
</tr>
</thead>
<tbody>
<tr>
<td>a rose is</td>
<td></td>
<td>1111</td>
</tr>
<tr>
<td>rose is a</td>
<td></td>
<td>2222</td>
</tr>
<tr>
<td>rose is a</td>
<td></td>
<td>3333</td>
</tr>
<tr>
<td>rose is a r</td>
<td></td>
<td>4444</td>
</tr>
<tr>
<td>se is a ro</td>
<td></td>
<td>5555</td>
</tr>
<tr>
<td>e is a ros</td>
<td></td>
<td>6666</td>
</tr>
<tr>
<td>is a rose</td>
<td></td>
<td>7777</td>
</tr>
<tr>
<td>is a rose</td>
<td></td>
<td>8888</td>
</tr>
<tr>
<td>s a rose i</td>
<td></td>
<td>9999</td>
</tr>
<tr>
<td>a rose is</td>
<td></td>
<td>0000</td>
</tr>
</tbody>
</table>
Basic Data Model: Sets

• **Document**: A document is represented as a set of shingles (more accurately, hashes of shingles)

• **Document similarity**: Jaccard similarity of the sets of shingles.
  • Common shingles over the union of shingles
  • \( \text{Sim} (C_1, C_2) = |C_1 \cap C_2|/|C_1 \cup C_2| \).

• Although we use the documents as our driving example the techniques we will describe apply to any kind of sets.
  • E.g., similar customers or items.
Signatures

• **Problem**: shingle sets are too large to be kept in memory.

• **Key idea**: “hash” each set $S$ to a small signature $\text{Sig}(S)$, such that:

1. $\text{Sig}(S)$ is **small enough** that we can fit a signature in main memory for each set.

2. $\text{Sim}(S_1, S_2)$ is **(almost) the same** as the “similarity” of $\text{Sig}(S_1)$ and $\text{Sig}(S_2)$. (signature preserves similarity).

• **Warning**: This method can produce **false negatives**, and **false positives** (if an additional check is not made).
  • **False negatives**: Similar items deemed as non-similar
  • **False positives**: Non-similar items deemed as similar
From Sets to Boolean Matrices

- Represent the data as a boolean matrix $M$
  - **Rows** = the universe of all possible set elements
    - In our case, shingle fingerprints take values in $[0...2^{64}-1]$  
  - **Columns** = the sets
    - In our case, documents, sets of shingle fingerprints
  - $M(r,S) = 1$ in row $r$ and column $S$ if and only if $r$ is a member of $S$.

- Typical matrix is sparse.
  - We do not really materialize the matrix
Example

- Universe: \( U = \{A, B, C, D, E, F, G\} \)
- \( X = \{A, B, F, G\} \)
- \( Y = \{A, E, F, G\} \)

\[
\text{Sim}(X, Y) = \frac{3}{5}
\]
Example

- Universe: \( U = \{A,B,C,D,E,F,G\} \)

- \( X = \{A,B,F,G\} \)

- \( Y = \{A,E,F,G\} \)

- \( \text{Sim}(X,Y) = \frac{3}{5} \)

At least one of the columns has value 1
Example

- Universe: \( U = \{A, B, C, D, E, F, G\} \)
- \( X = \{A, B, F, G\} \)
- \( Y = \{A, E, F, G\} \)
- \( \text{Sim}(X, Y) = \frac{3}{5} \)

Both columns have value 1
Minhashing

- Pick a random permutation of the rows (the universe U).
- Define “hash” function for set S
  - $h(S) = \text{the index of the first row (in the permuted order) in which column } S \text{ has } 1.$
  - OR
  - $h(S) = \text{the index of the first element of } S \text{ in the permuted order.}$
- Use $k$ (e.g., $k = 100$) independent random permutations to create a signature.
Example of minhash signatures

- **Input matrix**

<table>
<thead>
<tr>
<th></th>
<th>S₁</th>
<th>S₂</th>
<th>S₃</th>
<th>S₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

A C G F B E D

<table>
<thead>
<tr>
<th></th>
<th>S₁</th>
<th>S₂</th>
<th>S₃</th>
<th>S₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 A</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2 C</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3 G</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4 F</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5 B</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6 E</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7 D</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

1 2 1 2
Example of minhash signatures

- **Input matrix**

<table>
<thead>
<tr>
<th></th>
<th>S₁</th>
<th>S₂</th>
<th>S₃</th>
<th>S₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>S₁</th>
<th>S₂</th>
<th>S₃</th>
<th>S₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>D</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>A</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>C</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>F</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>G</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>E</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

2 1 3 1
Example of minhash signatures

- **Input matrix**

<table>
<thead>
<tr>
<th></th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

The signature for each row is computed by selecting the smallest element in each column. The resulting minhash signatures are:

- C: 0 1 0 1
- D: 0 1 0 1
- G: 1 0 1 0
- F: 1 0 1 0
- A: 1 0 1 0
- B: 0 1 0 1
- E: 0 1 0 1

The output minhash signatures are:

- 1 0 1 0
- 0 1 0 1
- 1 0 1 0
- 1 0 1 0
- 1 0 1 0
- 0 1 0 1
- 0 1 0 1
Example of minhash signatures

- **Input matrix**

<table>
<thead>
<tr>
<th></th>
<th>S₁</th>
<th>S₂</th>
<th>S₃</th>
<th>S₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- **Signature matrix**

<table>
<thead>
<tr>
<th></th>
<th>S₁</th>
<th>S₂</th>
<th>S₃</th>
<th>S₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>h₁</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>h₂</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>h₃</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

- \( \text{Sig}(S) = \text{vector of hash values} \)
  - e.g., \( \text{Sig}(S_2) = [2,1,1] \)
- \( \text{Sig}(S,i) = \text{value of the i-th hash function for set } S \)
  - E.g., \( \text{Sig}(S_2,3) = 1 \)
Hash function Property

\[ \Pr(h(S_1) = h(S_2)) = \text{Sim}(S_1, S_2) \]

- where the probability is over all choices of permutations.

- \textbf{Why?}
  - The first row where one of the two sets has value 1 belongs to the \textit{union}.
    - Recall that union contains rows with at least one 1.
  - We have equality if both sets have value 1, and this row belongs to the \textit{intersection}.
Example

• Universe: \( U = \{A, B, C, D, E, F, G\} \)
• \( X = \{A, B, F, G\} \)
• \( Y = \{A, E, F, G\} \)

• Union = \( \{A, B, E, F, G\} \)
• Intersection = \( \{A, F, G\} \)

Rows C, D could be anywhere they do not affect the probability
Example

- Universe: \( U = \{A,B,C,D,E,F,G\} \)
- \( X = \{A,B,F,G\} \)
- \( Y = \{A,E,F,G\} \)

- Union = \( \{A,B,E,F,G\} \)
- Intersection = \( \{A,F,G\} \)

The * rows belong to the union
Example

- Universe: $U = \{A, B, C, D, E, F, G\}$
- $X = \{A, B, F, G\}$
- $Y = \{A, E, F, G\}$

- Union = $\{A, B, E, F, G\}$
- Intersection = $\{A, F, G\}$

The question is what is the value of the first * element
Example

- **Universe:** \( U = \{A,B,C,D,E,F,G\} \)
- **\( X = \{A,B,F,G\} \)**
- **\( Y = \{A,E,F,G\} \)**

- **Union =** \( \{A,B,E,F,G\} \)
- **Intersection =** \( \{A,F,G\} \)

If it belongs to the intersection then \( h(X) = h(Y) \)
Example

- Universe: $U = \{A,B,C,D,E,F,G\}$
- $X = \{A,B,F,G\}$
- $Y = \{A,E,F,G\}$

Every element of the union is equally likely to be the * element:

$$Pr(h(X) = h(Y)) = \frac{|\{A,F,G\}|}{|\{A,B,E,F,G\}|} = \frac{3}{5} = \text{Sim}(X,Y)$$

- Union = $\{A,B,E,F,G\}$
- Intersection = $\{A,F,G\}$
Similarity for Signatures

- The similarity of signatures is the fraction of the hash functions in which they agree.

<table>
<thead>
<tr>
<th></th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Signature matrix

<table>
<thead>
<tr>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

Zero similarity is preserved
High similarity is well approximated

- With multiple signatures we get a good approximation

<table>
<thead>
<tr>
<th></th>
<th>Actual</th>
<th>Sig</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(S_1, S_2)$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$(S_1, S_3)$</td>
<td>3/5</td>
<td>2/3</td>
</tr>
<tr>
<td>$(S_1, S_4)$</td>
<td>1/7</td>
<td>0</td>
</tr>
<tr>
<td>$(S_2, S_3)$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$(S_2, S_4)$</td>
<td>3/4</td>
<td>1</td>
</tr>
<tr>
<td>$(S_3, S_4)$</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Is it now feasible?

• Assume a billion rows
• Hard to pick a random permutation of 1…billion
• **Even representing a random permutation requires 1 billion entries!!!**
• How about accessing rows in permuted order?
  • 😞
Being more practical

Approximating row permutations: pick $k=100$ hash functions $(h_1,\ldots,h_k)$

for each row $r$

for each hash function $h_i$

compute $h_i(r)$

for each column $S$ that has 1 in row $r$

if $h_i(r)$ is a smaller value than $\text{Sig}(S,i)$ then

$\text{Sig}(S,i) = h_i(r)$;

$\text{Sig}(S,i)$ will become the smallest value of $h_i(r)$ among all rows (shingles) for which column $S$ has value 1 (shingle belongs in $S$); i.e., $h_i(r)$ gives the min index for the $i$-th permutation

In practice this means selecting the function parameters

In practice only the rows (shingles) that appear in the data

$h_i(r) =$ index of shingle $r$ in permutation

$S$ contains shingle $r$

Find the shingle $r$ with minimum index
### Example

<table>
<thead>
<tr>
<th>x</th>
<th>Row</th>
<th>S1</th>
<th>S2</th>
<th>Sig1</th>
<th>Sig2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>B</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>C</td>
<td>1</td>
<td>1</td>
<td>h(1) = 2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>D</td>
<td>1</td>
<td>0</td>
<td>g(1) = 0</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>E</td>
<td>0</td>
<td>1</td>
<td>h(2) = 3</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>g(2) = 2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>h(3) = 4</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>g(3) = 4</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>h(4) = 0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>g(4) = 1</td>
<td>2</td>
</tr>
</tbody>
</table>

\[ h(x) = x + 1 \mod 5 \]
\[ g(x) = 2x + 3 \mod 5 \]
Implementation – (4)

- Often, data is given by column, not row.
  - E.g., columns = documents, rows = shingles.
- If so, sort matrix once so it is by row.
- And always compute $h_i(r)$ only once for each row.