Finding Similar Items: Locality Sensitive Hashing

CS246: Mining Massive Datasets
Jure Leskovec, Stanford University
http://cs246.stanford.edu
High Dimensional Data

- Many real-world problems
  - Web Search and Text Mining
    - Billions of documents, millions of terms
  - Product Recommendations
    - Millions of customers, millions of products
  - Scene Completion, other graphics problems
    - Image features
  - Online Advertising, Behavioral Analysis
    - Customer actions e.g., websites visited, searches
Many problems can be expressed as finding “similar” sets:

- Find near-neighbors in high-D space

Examples:

- Pages with similar words
  - For duplicate detection, classification by topic
- Customers who purchased similar products
  - NetFlix users with similar tastes in movies
  - Products with similar customer sets
- Images with similar features
- Users who visited the similar websites
Scene Completion Problem

[Hays and Efros, SIGGRAPH 2007]
Scene Completion Problem

[Hays and Efros, SIGGRAPH 2007]
Scene Completion Problem

10 nearest neighbors from a collection of 20,000 images
Scene Completion Problem

10 nearest neighbors from a collection of 2 million images
We formally define “near neighbors” as points that are a “small distance” apart.

For each use case, we need to define what “distance” means.

Two major classes of distance measures:

- A *Euclidean distance* is based on the locations of points in such a space.
- A *Non-Euclidean distance* is based on properties of points, but not their “location” in a space.
Some Euclidean Distances

- **$L_2$ norm:** $d(p, q) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$
  
  The most common notion of “distance”

- **$L_1$ norm:** sum of the absolute differences in each dimension
  
  - **Manhattan distance** = distance if you had to travel along coordinates only

\[ d_1(p, q) = \| p - q \|_1 = \sum_{i=1}^{n} |p_i - q_i|, \]
Non-Euclidean Distances: Cosine

- Think of a point as a vector from the origin (0,0,...,0) to its location
- Two vectors make an angle, whose cosine is normalized dot-product of the vectors:

\[ d(A,B) = \theta = \arccos \left( \frac{A \cdot B}{\|A\| \cdot \|B\|} \right) \]

- **Example:** A = 00111; B = 10011
  - A·B = 2; \|A\| = \|B\| = \sqrt{3}
  - \cos(\theta) = 2/3; \theta is about 48 degrees

**Note:** if A,B>0 then we can simplify the expression to

\[ d(A,B) = 1 - \frac{A \cdot B}{\|A\| \cdot \|B\|} \]
The Jaccard Similarity of two sets is the size of their intersection / the size of their union:

\[ \text{Sim}(C_1, C_2) = \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|} \]

The Jaccard Distance between sets is 1 minus their Jaccard similarity:

\[ d(C_1, C_2) = 1 - \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|} \]

3 in intersection
8 in union
Jaccard similarity = 3/8
Jaccard distance = 5/8
Finding Similar Items
**Finding Similar Documents**

- **Goal:** Given a large number (N in the millions or billions) of text documents, find pairs that are “near duplicates”

- **Applications:**
  - Mirror websites, or approximate mirrors
    - Don’t want to show both in a search
  - Similar news articles at many news sites
    - Cluster articles by “same story”

- **Problems:**
  - Many small pieces of one doc can appear out of order in another
  - Too many docs to compare all pairs
  - Docs are so large or so many that they cannot fit in main memory
3 Essential Steps for Similar Docs

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

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

3. **Locality-sensitive hashing**: Focus on pairs of signatures likely to be from similar documents
The set of strings of length $k$ that appear in the document

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

**Candidate pairs:** those pairs of signatures that we need to test for similarity.
Step 1: **Shingling:** Convert documents, emails, etc., to sets

**Simple approaches:**
- Document = set of words appearing in doc
- Document = set of “important” words
- Don’t work well for this application. Why?

Need to account for ordering of words

A different way: **Shingles**
Define: Shingles

- A \( k \)-shingle (or \( k \)-gram) for a document is a sequence of \( k \) tokens that appears in the document.
  - Tokens can be characters, words or something else, depending on application.
  - Assume tokens = characters for examples.
- **Example:** \( k=2; D_1=abcab \)
  - Set of 2-shingles: \( S(D_1)=\{ab, bc, ca\} \)
  - **Option:** Shingles as a bag, count \( ab \) twice.
- Represent a doc by the set of hash values of its \( k \)-shingles.
Compressing Shingles

- 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.

- **Idea:** Two documents could (rarely) appear to have shingles in common, when in fact only the hash-values were shared.

- **Example:** $k=2$; $D_1 = abcab$
  
  Set of 2-shingles: $S(D_1) = \{ab, bc, ca\}$
  
  Hash the singles: $h(D_1) = \{1, 5, 7\}$
Similarity Metric for Shingles

- Document $D_1 = \text{set of k-shingles } C_1=S(D_1)$
- Equivalently, each document is a 0/1 vector in the space of k-shingles
  - Each unique shingle is a dimension
  - Vectors are very sparse
- A natural similarity measure is the Jaccard similarity:

  $$\text{Sim}(D_1, D_2) = \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|}$$
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
  - $k = 5$ is OK for short documents
  - $k = 10$ is better for long documents
Suppose we need to find near-duplicate documents among N=1 million documents

Naïvely, we’d have to compute pairwise Jaccard similarities for every pair of docs

- i.e, $N(N-1)/2 \approx 5 \times 10^{11}$ comparisons
- At $10^5$ secs/day and $10^6$ comparisons/sec, it would take 5 days

For $N = 10$ million, it takes more than a year...
MinHashing

Step 2: **Minhashing:** Convert large sets to short signatures, while preserving similarity

- **Document:**
  - Shingling
    - The set of strings of length $k$ that appear in the document

- **MinHashing:**
  - Signatures: short integer vectors that represent the sets, and reflect their similarity

- **Locality-Sensitive Hashing:**
  - Candidate pairs: those pairs of signatures that we need to test for similarity.
Many similarity problems can be formalized as finding subsets that have significant intersection.

- Encode sets using 0/1 (bit, boolean) vectors
  - One dimension per element in the universal set
  - Interpret set intersection as bitwise AND, and set union as bitwise OR

**Example:** \( C_1 = 10111; C_2 = 10011 \)

- Size of intersection = 3; size of union = 4, Jaccard similarity (not distance) = 3/4
- \( d(C_1, C_2) = 1 − \) (Jaccard similarity) = 1/4
From Sets to Boolean Matrices

- Rows = elements of the universal set
- Columns = sets

- 1 in row $e$ and column $s$ if and only if $e$ is a member of $s$
- Column similarity is the Jaccard similarity of the sets of their rows with 1

- Typical matrix is sparse
Example: Jaccard of Columns

- **Each document is a column:**
  - Example: $C_1 = 1100011$; $C_2 = 0110010$
    - Size of intersection = 2; size of union = 5, Jaccard similarity (not distance) = 2/5
    - $d(C_1, C_2) = 1 - \text{(Jaccard similarity)} = 3/5$

**Note:**

- We might not really represent the data by a boolean matrix
- Sparse matrices are usually better represented by the list of places where there is a non-zero value
So far:
- Documents → Sets of shingles
- Represent sets as boolean vectors in a matrix

Next Goal: Find similar columns

Approach:
1) Signatures of columns: small summaries of columns
2) Examine pairs of signatures to find similar columns
   - Essential: Similarities of signatures & columns are related
3) Optional: check that columns with similar sigs. are really similar

Warnings:
- Comparing all pairs may take too much time: job for LSH
  - These methods can produce false negatives, and even false positives (if the optional check is not made)
Key idea: “hash” each column $C$ to a small signature $h(C)$, such that:

1. $h(C)$ is small enough that the signature fits in RAM
2. $sim(C_1, C_2)$ is the same as the “similarity” of signatures $h(C_1)$ and $h(C_2)$

Goal: Find a hash function $h()$ such that:

- if $sim(C_1, C_2)$ is high, then with high prob. $h(C_1) = h(C_2)$
- if $sim(C_1, C_2)$ is low, then with high prob. $h(C_1) \neq h(C_2)$

Hash docs into buckets, and expect that “most” pairs of near duplicate docs hash into the same bucket
Goal: Find a hash function $h()$ such that:

- if $\text{sim}(C_1, C_2)$ is high, then with high prob. $h(C_1) = h(C_2)$
- if $\text{sim}(C_1, C_2)$ is low, then with high prob. $h(C_1) \neq h(C_2)$

Clearly, the hash function depends on the similarity metric:

- Not all similarity metrics have a suitable hash function

There is a suitable hash function for Jaccard similarity: \textbf{Min-hashing}
Imagine the rows of the boolean matrix permuted under random permutation $\pi$

Define a “hash” function $h_{\pi}(C) = \text{the number of the first (in the permuted order $\pi$) row in which column $C$ has value 1}$:

$$h_{\pi}(C) = \min_{\pi}(C)$$

Use several (e.g., 100) independent hash functions to create a signature of a column
Min-Hashing Example

Permutation $\pi$

1 4 3
3 2 4
7 1 7
6 3 6
2 6 1
5 7 2
4 5 5

Input matrix (Shingles x Documents)

1 0 1 0
1 0 0 1
0 1 0 1
0 1 0 1
0 1 0 1
1 0 1 0
1 0 1 0

Signature matrix $M$

2 1 2 1
2 1 4 1
1 2 1 2
Surprising Property

- Choose a random permutation $\pi$
- then $\Pr[h_\pi(C_1) = h_\pi(C_2)] = \text{sim}(C_1, C_2)$
- Why?
  - Let $X$ be a set of shingles, $X \subseteq [2^{64}]$, $x \in X$
  - Then: $\Pr[\pi(y) = \min(\pi(X))] = 1/|X|$
    - It is equally likely that any $y \in X$ is mapped to the min element
  - Let $x$ be s.t. $\pi(x) = \min(\pi(C_1 \cup C_2))$
  - Then either: $\pi(x) = \min(\pi(C_1))$ if $x \in C_1$, or
    $\pi(x) = \min(\pi(C_2))$ if $x \in C_2$
  - So the prob. that both are true is the prob. $x \in C_1 \cap C_2$
  - $\Pr[\min(\pi(C_1))=\min(\pi(C_2))] = |C_1 \cap C_2|/|C_1 \cup C_2| = \text{sim}(C_1, C_2)$
Similarity for Signatures

- We know: \( \Pr[h_\pi(C_1) = h_\pi(C_2)] = \text{sim}(C_1, C_2) \)
- Now generalize to multiple hash functions
- The *similarity of two signatures* is the fraction of the hash functions in which they agree
- **Note:** Because of the minhash property, the similarity of columns is the same as the expected similarity of their signatures
### Min Hashing – Example

**Input matrix**

<table>
<thead>
<tr>
<th>1</th>
<th>4</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Signature matrix $M$**

<table>
<thead>
<tr>
<th>2</th>
<th>1</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

**Similarities:**

<table>
<thead>
<tr>
<th>1-3</th>
<th>2-4</th>
<th>1-2</th>
<th>3-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.75</td>
<td>0.75</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0.67</td>
<td>1.00</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Col/Col Similarity:**

| 0.75 | 0.75 | 0   | 0   |

**Sig/Sig Similarity:**

| 0.67 | 1.00 | 0   | 0   |
MinHash Signatures

- Pick 100 random permutations of the rows
- Think of \( \text{sig}(C) \) as a column vector
- Let \( \text{sig}(C)[i] = \) according to the \( i \)-th permutation, the index of the first row that has a 1 in column \( C \)

\[
\text{sig}(C)[i] = \min (\pi_i(C))
\]

- **Note:** The sketch (signature) of document \( C \) is small -- \( \sim 100 \) bytes!
  - We achieved our goal! We “compressed” long bit vectors into short signatures
Step 3: **Locality-sensitive hashing**: Focus on pairs of signatures likely to be from similar documents.

- **Document**: The set of strings of length $k$ that appear in the document.
- **MinHashing**: Signatures: short integer vectors that represent the sets, and reflect their similarity.
- **Locality-sensitive Hashing**: Candidate pairs: those pairs of signatures that we need to test for similarity.
**Goal:** Find documents with Jaccard similarity at least $s$ (for some similarity threshold, e.g., $s=0.8$)

**LSH – General idea:** Use a function $f(x,y)$ that tells whether $x$ and $y$ is a candidate pair: a pair of elements whose similarity must be evaluated.

**For minhash matrices:**
- Hash columns of signature matrix $M$ to many buckets
- Each pair of documents that hashes into the same bucket is a candidate pair
### Candidates from Minhash

- Pick a similarity threshold \( s \), a fraction \(< 1\)

- Columns \( x \) and \( y \) of \( M \) are a **candidate pair** if their signatures agree on at least fraction \( s \) of their rows:
  \[
  M(i, x) = M(i, y) \text{ for at least } \text{frac. } s \text{ values of } i
  \]

- We expect documents \( x \) and \( y \) to have the same similarity as their signatures
Big idea: Hash columns of signature matrix $M$ several times

Arrange that (only) similar columns are likely to hash to the same bucket, with high probability

Candidate pairs are those that hash to the same bucket
Partition $M$ into Bands

$M$ has $r$ rows per band.

$M$ has $b$ bands.

Signature matrix $M$.

One signature.
Divide matrix $M$ into $b$ bands of $r$ rows

For each band, hash its portion of each column to a hash table with $k$ buckets
- Make $k$ as large as possible

*Candidate* column pairs are those that hash to the same bucket for $\geq 1$ band

Tune $b$ and $r$ to catch most similar pairs, but few non-similar pairs
Hashing Bands

Columns 2 and 6 are probably identical (candidate pair)

Columns 6 and 7 are surely different.
There are enough buckets that columns are unlikely to hash to the same bucket unless they are identical in a particular band.

Hereafter, we assume that “same bucket” means “identical in that band”.

Assumption needed only to simplify analysis, not for correctness of algorithm.
Assume the following case:

- Suppose 100,000 columns of $M$ (100k docs)
- Signatures of 100 integers (rows)
- Therefore, signatures take 40Mb
- Choose 20 bands of 5 integers/band

**Goal:** Find pairs of documents that are at least $s = 80\%$ similar
Assume: $C_1, C_2$ are 80% similar

- Since $s=80\%$ we want $C_1, C_2$ to hash to at least one common bucket (at least one band is identical)

- Probability $C_1, C_2$ identical in one particular band: $(0.8)^5 = 0.328$

- Probability $C_1, C_2$ are not similar in all of the 20 bands: $(1-0.328)^{20} = 0.00035$
  - i.e., about 1/3000th of the 80%-similar column pairs are false negatives
  - We would find 99.965% pairs of truly similar documents
Assume: \( C_1, C_2 \) are 30% similar

- Since \( s = 80\% \) we want \( C_1, C_2 \) to hash to at **NO common buckets** (all bands should be different)

- Probability \( C_1, C_2 \) identical in one particular band: \( (0.3)^5 = 0.00243 \)

- Probability \( C_1, C_2 \) identical in at least 1 of 20 bands: \( 1 - (1 - 0.00243)^{20} = 0.0474 \)

- In other words, approximately 4.74% pairs of docs with similarity 30% end up becoming candidate pairs -- **false positives**
LSH Involves a Tradeoff

- **Pick:**
  - the number of minhashes (rows of M)
  - the number of bands $b$, and
  - the number of rows $r$ per band
  to balance false positives/negatives

- **Example:** if we had only 15 bands of 5 rows, the number of false positives would go down, but the number of false negatives would go up
Similarity $s$ of two sets

Probability of sharing a bucket

No chance if $s < t$

Probability $= 1$ if $s > t$
What 1 Band of 1 Row Gives You

Remember: Probability of equal hash-values = similarity

Probability of sharing a bucket

Similarity $s$ of two sets

$t$
Columns $C_1$ and $C_2$ have similarity $s$

Pick any band ($r$ rows)

- Prob. that all rows in band equal = $s^r$
- Prob. that some row in band unequal = $1 - s^r$

Prob. that no band identical = $(1 - s^r)^b$

Prob. that at least 1 band identical = $1 - (1 - s^r)^b$
What $b$ Bands of $r$ Rows Gives You

Probability of sharing a bucket

$\text{Similarity } s \text{ of two sets}$

$t \sim (1/b)^{1/r}$

$1 - (1 - s^r)^b$

At least one band identical

No bands identical

Some row of a band unequal

All rows of a band are equal

1/17/2012

Jure Leskovec, Stanford C246: Mining Massive Datasets
Example: $b = 20; r = 5$

- Similarity threshold $s$
- Prob. that at least 1 band identical:

<table>
<thead>
<tr>
<th>$s$</th>
<th>$1-(1-s^r)^b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>.2</td>
<td>.006</td>
</tr>
<tr>
<td>.3</td>
<td>.047</td>
</tr>
<tr>
<td>.4</td>
<td>.186</td>
</tr>
<tr>
<td>.5</td>
<td>.470</td>
</tr>
<tr>
<td>.6</td>
<td>.802</td>
</tr>
<tr>
<td>.7</td>
<td>.975</td>
</tr>
<tr>
<td>.8</td>
<td>.9996</td>
</tr>
</tbody>
</table>
Picking \( r \) and \( b \): The S-curve

- Picking \( r \) and \( b \) to get the best
  - 50 hash-functions (\( r=5, \ b=10 \))

![Graph showing the probability of sharing a bucket with similarity values and areas marked for False Negative and False Positive rates.](image-url)
**LSH Summary**

- Tune to get almost all pairs with similar signatures, but eliminate most pairs that do not have similar signatures.

- Check in main memory that candidate pairs really do have similar signatures.

- **Optional:** In another pass through data, check that the remaining candidate pairs really represent similar documents.
Summary: 3 Steps

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

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

3. **Locality-sensitive hashing**: Focus on pairs of signatures likely to be from similar documents

```
\begin{align*}
\text{Depends on the distance metric}
\end{align*}
```