Finding similar items I

Krzysztof Dembczyński

Intelligent Decision Support Systems Laboratory (IDSS)
Poznań University of Technology, Poland

Software Development Technologies
Master studies, second semester
Academic year 2018/19 (winter course)
Review of the previous lectures

• Mining of massive datasets.
• Classification and regression.
• Evolution of database systems.
• MapReduce
• MapReduce in Apache Spark
Outline

1. Motivation
2. Shingling of Documents
3. Similarity-Preserving Summaries of Sets
4. Locality-Sensitive Hashing for Documents
5. Summary
Outline

1 Motivation

2 Shingling of Documents

3 Similarity-Preserving Summaries of Sets

4 Locality-Sensitive Hashing for Documents

5 Summary
Nearest neighbor search

- Find similar elements to the query element.
Applications of nearest neighbor search

• Similarity of documents
  ▶ Plagiarism
  ▶ Mirror pages
  ▶ Articles from the same source

• Machine learning
  ▶ k-nearest neighbors
  ▶ Collaborative filtering

• Computational geometry

• Computer vision

• Geographic Information Systems (GIS)
Nearest neighbor search

- Brute force search:

  - Given a query point $q$ scan through each of $n$ data points in database.
  - Computational complexity for 1-NN query: $O(n)$.
  - Computational complexity for k-NN query: $O(n \log k)$ or $O(n + k)$.

- With large databases linear complexity can be too costly.
- Can we do better?
- Data structures for exact search: not robust to curse of dimensionality.
- Approximate algorithms.
Nearest neighbor search

• Brute force search:
  ▶ Given a query point \( q \) scan through each of \( n \) data points in database

\[ \text{Computational complexity for 1-NN query: } O(n) \]

\[ \text{Computational complexity for k-NN query: } O(n \log k) \text{ or } O(n + k) \]

• With large databases linear complexity can be too costly.

• Can we do better?

• Data structures for exact search: not robust to curse of dimensionality

• Approximate algorithms
Nearest neighbor search

- Brute force search:
  - Given a query point $q$ scan through each of $n$ data points in database
  - Computational complexity for 1-NN query:
Nearest neighbor search

- Brute force search:
  - Given a query point $q$ scan through each of $n$ data points in database
  - Computational complexity for 1-NN query: $O(n)$. 
Nearest neighbor search

- **Brute force search:**
  - Given a query point $q$ scan through each of $n$ data points in database
  - Computational complexity for 1-NN query: $\mathcal{O}(n)$.
  - Computational complexity for k-NN query:
Nearest neighbor search

- **Brute force search:**
  - Given a query point $q$ scan through each of $n$ data points in database
  - Computational complexity for 1-NN query: $O(n)$.
  - Computational complexity for k-NN query: $O(n \log k)$ or
Nearest neighbor search

- **Brute force search:**
  - Given a query point $q$ scan through each of $n$ data points in database
  - Computational complexity for 1-NN query: $O(n)$.
  - Computational complexity for k-NN query: $O(n \log k)$ or $O(n + k)$
Nearest neighbor search

• Brute force search:
  ▶ Given a query point $q$ scan through each of $n$ data points in database
  ▶ Computational complexity for 1-NN query: $O(n)$.
  ▶ Computational complexity for k-NN query: $O(n \log k)$ or $O(n + k)$

• With large databases linear complexity can be too costly.
Nearest neighbor search

- Brute force search:
  - Given a query point \( q \) scan through each of \( n \) data points in database
  - Computational complexity for 1-NN query: \( \mathcal{O}(n) \).
  - Computational complexity for k-NN query: \( \mathcal{O}(n \log k) \) or \( \mathcal{O}(n + k) \)

- With large databases linear complexity can be too costly.

- Can we do better?
Nearest neighbor search

- Brute force search:
  - Given a query point \( q \) scan through each of \( n \) data points in database
  - Computational complexity for 1-NN query: \( O(n) \).
  - Computational complexity for k-NN query: \( O(n \log k) \) or \( O(n + k) \)

- With large databases linear complexity can be too costly.
- Can we do better?
- Data structures for exact search: not robust to curse of dimensionality
Nearest neighbor search

• Brute force search:
  ▶ Given a query point \( q \) scan through each of \( n \) data points in database
  ▶ Computational complexity for 1-NN query: \( \mathcal{O}(n) \).
  ▶ Computational complexity for k-NN query: \( \mathcal{O}(n \log k) \) or \( \mathcal{O}(n + k) \)

• With large databases linear complexity can be too costly.

• Can we do better?

• Data structures for exact search: not robust to curse of dimensionality

• Approximate algorithms
Outline

1. Motivation

2. Shingling of Documents

3. Similarity-Preserving Summaries of Sets

4. Locality-Sensitive Hashing for Documents

5. Summary
Motivation

- Consider an application of finding near-duplicates of Web pages, like plagiarisms or mirrors.
Motivation

- Consider an application of finding near-duplicates of Web pages, like plagiarisms or mirrors.
- We can represent pages as sets of character $k$-grams (or $k$-shingles) and formulate a problem as finding sets with a relatively large intersection.
Motivation

• Consider an application of finding near-duplicates of Web pages, like plagiarisms or mirrors.
• We can represent pages as sets of character $k$-grams (or $k$-shingles) and formulate a problem as finding sets with a relatively large intersection.
• Storing large number of sets and computing their similarity in naive way is not sufficient.
Consider an application of finding near-duplicates of Web pages, like plagiarisms or mirrors.

We can represent pages as sets of character $k$-grams (or $k$-shingles) and formulate a problem as finding sets with a relatively large intersection.

Storing large numbers of sets and computing their similarity in a naive way is not sufficient.

We compress sets in a way that enables to deduce the similarity of the underlying sets from their compressed versions.
• We focus on similarity of sets by looking at the relative size of their intersection.

Jaccard similarity

\[ \text{SIM}(S, T) = \frac{|S \cap T|}{|S \cup T|} \]

Example: Let \( S = \{a, b, c, d\} \) and \( T = \{c, d, e, f\} \), then \( \text{SIM}(S, T) = \frac{2}{6} \).
Jaccard similarity

• We focus on similarity of sets by looking at the relative size of their intersection.
• The Jaccard similarity of sets $S$ and $T$ is defined as:

$$SIM(S, T) = \frac{|S \cap T|}{|S \cup T|}$$

- Example: Let $S = \{a, b, c, d\}$ and $T = \{c, d, e, f\}$, then $SIM(S, T) = \frac{2}{6}$. 
Jaccard similarity

- We focus on similarity of sets by looking at the relative size of their intersection.
- The Jaccard similarity of sets $S$ and $T$ is defined as:

$$SIM(S, T) = \frac{|S \cap T|}{|S \cup T|}$$

- **Example**: Let $S = \{a, b, c, d\}$ and $T = \{c, d, e, f\}$, then

$$SIM(S, T) = \frac{2}{6}.$$
• A document is a string of characters.
• A document is a string of characters.
• A $k$-shingle (or $k$-gram) for a document is any substring of length $k$ found within the document.
\(k\)-shingles

- A document is a string of characters.
- A \(k\)-shingle (or \(k\)-gram) for a document is any substring of length \(k\) found within the document.
- Each document may be represented as a set of \(k\)-shingles that appear one or more times within that document.
A document is a string of characters. A \( k \)-shingle (or \( k \)-gram) for a document is any substring of length \( k \) found within the document. Each document may be represented as a set of \( k \)-shingles that appear one or more times within that document.

**Example**: The set of all 3-shingles for the first sentence on this slide:

\[
\{ \text{“A d”}, \text{“ do”}, \text{“doc”}, \text{“ocu”}, \text{“cum”}, \text{“ume”}, \text{“men”}, \ldots, \text{“ers”} \}\]
• A document is a string of characters.
• A $k$-shingle (or $k$-gram) for a document is any substring of length $k$ found within the document.
• Each document may be represented as a set of $k$-shingles that appear one or more times within that document.
• **Example**: The set of all 3-shingles for the first sentence on this slide:

$$\{\text{"A d"}, \text{" do"}, \text{"doc"}, \text{"ocu"}, \text{"cum"}, \text{"ume"}, \text{"men"}, \ldots, \text{"ers"}\}$$

• Several options regarding white spaces:
A document is a string of characters. A \( k \)-shingle (or \( k \)-gram) for a document is any substring of length \( k \) found within the document. Each document may be represented as a set of \( k \)-shingles that appear one or more times within that document.

**Example:** The set of all 3-shingles for the first sentence on this slide:

\[
\]

Several options regarding white spaces:
- Replace any sequence of one or more white spaces by a single blank.
**$k$-shingles**

- A document is a string of characters.
- A $k$-shingle (or $k$-gram) for a document is any substring of length $k$ found within the document.
- Each document may be represented as a **set** of $k$-shingles that appear one or more times within that document.
- **Example**: The set of all 3-shingles for the first sentence on this slide:

  \[
  \{\text{"A d"}, \text{" do"}, \text{"doc"}, \text{"ocu"}, \text{"cum"}, \text{"ume"}, \text{"men"}, \ldots, \text{"ers"}\}\]

- Several options regarding white spaces:
  - Replace any sequence of one or more white spaces by a single blank.
  - Remove all white spaces.
Size of shingles

- For small $k$ we would expect most sequences of $k$ characters to appear in most documents.
Size of shingles

- For small $k$, we would expect most sequences of $k$ characters to appear in most documents.
- For $k = 1$, most documents will have most of the common characters and few other characters, so almost all documents will have high similarity.

\[
\text{SIM}(\{d, o, c, u, m, e, n, t\}, \{m, o, n, u, m, e, n, t\}) = \frac{6}{8}
\]
\[
\text{SIM}(\{doc, ocu, cum, ume, men, ent\}, \{mon, onu, num, ume, men, ent\}) = \frac{3}{9}
\]
Size of shingles

• For small $k$ we would expect most sequences of $k$ characters to appear in most documents.
• For $k = 1$ most documents will have most of the common characters and few other characters, so almost all documents will have high similarity.
• $k$ should be picked large enough that the probability of any given shingle appearing in any given document is low.
Size of shingles

• For small $k$ we would expect most sequences of $k$ characters to appear in most documents.

• For $k = 1$ most documents will have most of the common characters and few other characters, so almost all documents will have high similarity.

• $k$ should be picked large enough that the probability of any given shingle appearing in any given document is low.

• **Example:** Let us check two words *document* and *monument*:

$$SIM(\{d, o, c, u, m, e, n, t\}, \{m, o, n, u, m, e, n, t\}) = \frac{6}{8}$$

$$SIM(\{doc, ocu, cum, ume, men, ent\}, \{mon, onu, num, ume, men, ent\}) = \frac{3}{9}$$
Size of shingles

• Example:
Size of shingles

- **Example:**
  - For corpus of emails setting \( k = 5 \) should be fine.
Size of shingles

- **Example:**
  - For corpus of emails setting $k = 5$ should be fine.
  - If only English letters and a general white-space character appear in emails, then there would be $27^5 = 14348907$ possible shingles.
Size of shingles

• **Example:**
  - For corpus of emails setting $k = 5$ should be fine.
  - If only English letters and a general white-space character appear in emails, then there would be $27^5 = 14348907$ possible shingles.
  - Since typical email is much smaller than 14 million characters long, this can be right value.
• **Example:**
  ▶ For corpus of emails setting $k = 5$ should be fine.
  ▶ If only English letters and a general white-space character appear in emails, then there would be $27^5 = 14348907$ possible shingles.
  ▶ Since typical email is much smaller than 14 million characters long, this can be right value.
  ▶ Since distribution of characters is not uniform, the above estimate should be corrected, for example, by assuming that there are only 20 characters.
• Instead of using substrings directly as shingles, we can pick a hash function that maps strings of length $k$ to some number of buckets.

Example:

▶ Each 9-shingle from a document can be mapped to a bucket number in the range from $0$ to $2^{32} - 1$.

▶ Instead of nine we use then four bytes and can manipulate (hashed) shingles by single-word machine operations.
Hashing shingles

• Instead of using substrings directly as shingles, we can pick a hash function that maps strings of length $k$ to some number of buckets.
• Then, the resulting bucket number can be treated as the shingle.
Hashing shingles

- Instead of using substrings directly as shingles, we can pick a hash function that maps strings of length $k$ to some number of buckets.
- Then, the resulting bucket number can be treated as the shingle.
- The set representing a document is then the set of integers that are bucket numbers of one or more $k$-shingles that appear in the document.
Hashing shingles

- Instead of using substrings directly as shingles, we can pick a hash function that maps strings of length $k$ to some number of buckets.
- Then, the resulting bucket number can be treated as the shingle.
- The set representing a document is then the set of integers that are bucket numbers of one or more $k$-shingles that appear in the document.
- **Example:**
Hashing shingles

• Instead of using substrings directly as shingles, we can pick a hash function that maps strings of length \( k \) to some number of buckets.
• Then, the resulting bucket number can be treated as the shingle.
• The set representing a document is then the set of integers that are bucket numbers of one or more \( k \)-shingles that appear in the document.
• **Example:**
  ▶ Each 9-shingle from a document can be mapped to a bucket number in the range from 0 to \( 2^{32} - 1 \).
• Instead of using substrings directly as shingles, we can pick a hash function that maps strings of length $k$ to some number of buckets.
• Then, the resulting bucket number can be treated as the shingle.
• The set representing a document is then the set of integers that are bucket numbers of one or more $k$-shingles that appear in the document.

**Example:**
- Each 9-shingle from a document can be mapped to a bucket number in the range from 0 to $2^{32} - 1$.
- Instead of nine we use then four bytes and can manipulate (hashed) shingles by single-word machine operations.
• Short shingles vs. hashed shingles

If we use 4-shingles, most sequences of four bytes are unlikely or impossible to find in typical documents. The effective number of different shingles is approximately $2^{20} = 160000$, much less than $2^{32}$. If we use 9-shingles, there are many more than $2^{32}$ likely shingles. When we hash them down to four bytes, we can expect almost any sequence of four bytes to be possible.
• Short shingles vs. hashed shingles
  ▶ If we use 4-shingles, most sequences of four bytes are unlikely or impossible to find in typical documents.
Hashing shingles

- Short shingles vs. hashed shingles
  - If we use 4-shingles, most sequences of four bytes are unlikely or impossible to find in typical documents.
  - The effective number of different shingles is approximately $20^4 = 160000$ much less than $2^{32}$. 

• **Short shingles vs. hashed shingles**
  ▶ If we use 4-shingles, most sequences of four bytes are unlikely or impossible to find in typical documents.
  ▶ The effective number of different shingles is approximately $20^4 = 160000$ much less than $2^{32}$.
  ▶ if we use 9-shingles, there are many more than $2^{32}$ likely shingles.
• Short shingles vs. hashed shingles
  ▶ If we use 4-shingles, most sequences of four bytes are unlikely or impossible to find in typical documents.
  ▶ The effective number of different shingles is approximately $20^4 = 160000$ much less than $2^{32}$.
  ▶ if we use 9-shingles, there are many more than $2^{32}$ likely shingles.
  ▶ When we hash them down to four bytes, we can expect almost any sequence of four bytes to be possible.
Outline

1 Motivation
2 Shingling of Documents
3 Similarity-Preserving Summaries of Sets
4 Locality-Sensitive Hashing for Documents
5 Summary
Similarity-preserving summaries of sets

• Sets of shingles are large!
Similarity-preserving summaries of sets

- Sets of shingles are large!
- Even if we hash them to four bytes each, the space needed to store a set is still roughly four times the space taken by the document.
Similarity-preserving summaries of sets

- Sets of shingles are large!
- Even if we hash them to four bytes each, the space needed to store a set is still roughly four times the space taken by the document.
- If we have millions of documents, it may well not be possible to store all the shingle-sets in main memory.
• Sets of shingles are large!
• Even if we hash them to four bytes each, the space needed to store a set is still roughly four times the space taken by the document.
• If we have millions of documents, it may well not be possible to store all the shingle-sets in main memory.
• We would like to replace large sets by much smaller representations called signatures.
• Sets of shingles are large!
• Even if we hash them to four bytes each, the space needed to store a set is still roughly four times the space taken by the document.
• If we have millions of documents, it may well not be possible to store all the shingle-sets in main memory.
• We would like to replace large sets by much smaller representations called signatures.
• The signatures, however, should preserve (at least to some extent) the similarity between sets.
Matrix representation of sets

- **Characteristic matrix**

  - The columns of the matrix correspond to the sets.
  - The rows correspond to elements of the universal set from which elements of the sets are drawn.
  - There is a 1 in row $r$ and column $c$ if the element for row $r$ is a member of the set for column $c$.
  - Otherwise the value in position $(r, c)$ is 0.
Matrix representation of sets

- **Characteristic matrix**
  - The columns of the matrix correspond to the sets.
Matrix representation of sets

• **Characteristic matrix**
  - The columns of the matrix correspond to the sets.
  - The rows correspond to elements of the universal set from which elements of the sets are drawn.
• **Characteristic matrix**
  - The columns of the matrix correspond to the sets.
  - The rows correspond to elements of the universal set from which elements of the sets are drawn.
  - There is a 1 in row $r$ and column $c$ if the element for row $r$ is a member of the set for column $c$.  

Matrix representation of sets

- **Characteristic matrix**
  - The columns of the matrix correspond to the sets.
  - The rows correspond to elements of the universal set from which elements of the sets are drawn.
  - There is a 1 in row $r$ and column $c$ if the element for row $r$ is a member of the set for column $c$.
  - Otherwise the value in position $(r, c)$ is 0.
Matrix representation of sets

- **Example:**
  - Let the universal set be \{a, b, c, d, e\}.
  - Let \(S_1 = \{a, d\}\), \(S_2 = \{c\}\), \(S_3 = \{b, d, e\}\), \(S_4 = \{a, c, d\}\).

<table>
<thead>
<tr>
<th>Element</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>0</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>d</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>e</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- It is important to remember that the characteristic matrix is unlikely to be the way the data is stored, but it is useful as a way to visualize the data!
Minhashing

- The signatures we desire to construct for sets are composed of the results of some number of calculations (say several hundred) each of which is a minhash of the characteristic matrix.
Minhashing

- The signatures we desire to construct for sets are composed of the results of some number of calculations (say several hundred) each of which is a minhash of the characteristic matrix.
- To minhash a set represented by a column of the characteristic matrix, pick a permutation of the rows.
Minhashing

- The signatures we desire to construct for sets are composed of the results of some number of calculations (say several hundred) each of which is a **minhash** of the characteristic matrix.
- To minhash a set represented by a column of the characteristic matrix, pick a permutation of the rows.
- The minhash value of any column is the number of the first row, in the permuted order, in which the column has a 1 (or, the first element of the set in the given permutation).
Minhashing

• The signatures we desire to construct for sets are composed of the results of some number of calculations (say several hundred) each of which is a minhash of the characteristic matrix.

• To minhash a set represented by a column of the characteristic matrix, pick a permutation of the rows.

• The minhash value of any column is the number of the first row, in the permuted order, in which the column has a 1 (or, the first element of the set in the given permutation).

• The index of the first row is 0 in the following.
Minhashing

- **Example:**
  - Let us pick the order of rows *beadc* for the matrix from the previous example.

<table>
<thead>
<tr>
<th>Element</th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>e</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>d</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>c</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

- In this matrix, we can read off the values of minhash ($mh$) by scanning from the top until we come to a 1.
- Thus, we see that $mh(S_1) = 2$ (*a*), $mh(S_2) = 4$ (*c*), $mh(S_3) = 0$ (*b*), and $mh(S_4) = 2$ (*a*).
Minhashing and Jaccard similarity

- There is a remarkable connection between minhashing and Jaccard similarity of the sets that are minhashed:
Minhashing and Jaccard similarity

- There is a remarkable connection between minhashing and Jaccard similarity of the sets that are minhashed:
  - The probability that the minhash function for a random permutation of rows produces the same value for two sets equals the Jaccard similarity of those sets.
Minhashing and Jaccard similarity

- Let us consider two sets, i.e., two columns of the characteristic matrix.

<table>
<thead>
<tr>
<th>Element</th>
<th>$S_1$</th>
<th>$S_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>e</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>d</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>c</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
• Let us consider two sets, i.e., two columns of the characteristic matrix.

<table>
<thead>
<tr>
<th>Element</th>
<th>$S_1$</th>
<th>$S_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>e</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>d</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>c</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

• The rows can be divided into three classes:
Minhashing and Jaccard similarity

- Let us consider two sets, i.e., two columns of the characteristic matrix.

<table>
<thead>
<tr>
<th>Element</th>
<th>$S_1$</th>
<th>$S_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>e</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>d</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>c</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

- The rows can be divided into three classes:
  - Type $X$ rows have 1 in both columns,
Minhashing and Jaccard similarity

• Let us consider two sets, i.e., two columns of the characteristic matrix.

<table>
<thead>
<tr>
<th>Element</th>
<th>$S_1$</th>
<th>$S_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>e</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>d</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>c</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

• The rows can be divided into three classes:
  ▶ Type $X$ rows have 1 in both columns,
  ▶ Type $Y$ rows have 1 in one of the columns and 0 in the other,
Minhashing and Jaccard similarity

- Let us consider two sets, i.e., two columns of the characteristic matrix.

<table>
<thead>
<tr>
<th>Element</th>
<th>$S_1$</th>
<th>$S_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>e</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>d</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>c</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

- The rows can be divided into three classes:
  - Type $X$ rows have 1 in both columns,
  - Type $Y$ rows have 1 in one of the columns and 0 in the other,
  - Type $Z$ rows have 0 in both columns.
Minhashing and Jaccard similarity

• Since the matrix is sparse, most rows are of type $\mathcal{Z}$. 

• The ratio of the numbers of type $X$ and type $Y$ rows determine both $\text{SIM}(S, T)$ and the probability that $\text{mh}(S) = \text{mh}(T)$.

• Let there be $x$ rows of type $X$ and $y$ rows of type $Y$.

• Then, the Jaccard similarity is: $\text{SIM}(S, T) = \frac{x}{x+y}$.

• If we imagine the rows permuted randomly, and we proceed from the top, the probability that we shall meet a type $X$ row before we meet a type $Y$ row is, as before, $P(\text{mh}(S) = \text{mh}(T)) = \frac{x}{x+y}$. 
Minhashing and Jaccard similarity

- Since the matrix is sparse, most rows are of type $Z$.
- The ratio of the numbers of type $X$ and type $Y$ rows determine both $SIM(S, T)$ and the probability that $mh(S) = mh(T)$. 

$$SIM(S, T) = \frac{x}{x + y}$$

If we imagine the rows permuted randomly, and we proceed from the top, the probability that we shall meet a type $X$ row before we meet a type $Y$ row is, as before, $$P(mh(S) = mh(T)) = \frac{x}{x + y}.$$
Minhashing and Jaccard similarity

- Since the matrix is sparse, most rows are of type $Z$.
- The ratio of the numbers of type $X$ and type $Y$ rows determine both $SIM(S, T)$ and the probability that $mh(S) = mh(T)$.
- Let there be $x$ rows of type $X$ and $y$ rows of type $Y$. 

$$SIM(S, T) = \frac{x}{x + y}$$
Minhashing and Jaccard similarity

- Since the matrix is sparse, most rows are of type $Z$.
- The ratio of the numbers of type $X$ and type $Y$ rows determine both $SIM(S, T)$ and the probability that $mh(S) = mh(T)$.
- Let there be $x$ rows of type $X$ and $y$ rows of type $Y$.
- Then, the Jaccard similarity is:
Minhashing and Jaccard similarity

• Since the matrix is sparse, most rows are of type $Z$.
• The ratio of the numbers of type $X$ and type $Y$ rows determine both $SIM(S, T)$ and the probability that $mh(S) = mh(T)$.
• Let there be $x$ rows of type $X$ and $y$ rows of type $Y$.
• Then, the Jaccard similarity is:

$$SIM(S, T) = \frac{x}{x + y}.$$
Minhashing and Jaccard similarity

• Since the matrix is sparse, most rows are of type \( Z \).
• The ratio of the numbers of type \( X \) and type \( Y \) rows determine both \( \text{SIM}(S, T) \) and the probability that \( mh(S) = mh(T) \).
• Let there be \( x \) rows of type \( X \) and \( y \) rows of type \( Y \).
• Then, the Jaccard similarity is:

\[
\text{SIM}(S, T) = \frac{x}{x + y}.
\]

• If we imagine the rows permuted randomly, and we proceed from the top, the probability that we shall meet a type \( X \) row before we meet a type \( Y \) row is
Minhashing and Jaccard similarity

- Since the matrix is sparse, most rows are of type $Z$.
- The ratio of the numbers of type $X$ and type $Y$ rows determine both $SIM(S, T)$ and the probability that $mh(S) = mh(T)$.
- Let there be $x$ rows of type $X$ and $y$ rows of type $Y$.
- Then, the Jaccard similarity is:

$$SIM(S, T) = \frac{x}{x + y}.$$  

- If we imagine the rows permuted randomly, and we proceed from the top, the probability that we shall meet a type $X$ row before we meet a type $Y$ row is, as before,

$$P(mh(S) = mh(T)) = \frac{x}{x + y}.$$
Minhash signatures

• For a given collection of sets represented by their characteristic matrix $M$, the signatures are produced in the following way:
Minhash signatures

• For a given collection of sets represented by their characteristic matrix $M$, the signatures are produced in the following way:
  ▶ Pick at random some number $n$ of permutations of the rows of $M$ (let say, around 100 or 1000).
Minhash signatures

- For a given collection of sets represented by their characteristic matrix $M$, the signatures are produced in the following way:
  - Pick at random some number $n$ of permutations of the rows of $M$ (let say, around 100 or 1000).
  - Call the minhash functions determined by these permutations $mh_1$, $mh_2$, $\ldots$, $mh_n$. 

  - From the column representing set $S$, construct the minhash signature for $S$, the vector $(mh_1(S), mh_2(S), \ldots, mh_n(S))$ – represented as a column.

- Thus, we can form from matrix $M$ a signature matrix, in which the $i$-th column of $M$ is replaced by the minhash signature for (the set of) the $i$-th column.

- The signature matrix has the same number of columns as $M$, but only $n$ rows!

- Even if $M$ is not represented explicitly (but as a sparse matrix by the location of its ones), it is normal for the signature matrix to be much smaller than $M$. 

Minhash signatures

• For a given collection of sets represented by their characteristic matrix $M$, the signatures are produced in the following way:
  ▶ Pick at random some number $n$ of permutations of the rows of $M$ (let say, around 100 or 1000).
  ▶ Call the minhash functions determined by these permutations $mh_1$, $mh_2$, $\ldots$, $mh_n$.
  ▶ From the column representing set $S$, construct the minhash signature for $S$, the vector $(mh_1(S), mh_2(S), \ldots, mh_n(S))$ – represented as a column.
Minhash signatures

• For a given collection of sets represented by their characteristic matrix $M$, the signatures are produced in the following way:
  ▶ Pick at random some number $n$ of permutations of the rows of $M$ (let say, around 100 or 1000).
  ▶ Call the minhash functions determined by these permutations $mh_1$, $mh_2$, $\ldots$, $mh_n$.
  ▶ From the column representing set $S$, construct the minhash signature for $S$, the vector $(mh_1(S), mh_2(S), \ldots, mh_n(S))$ – represented as a column.
  ▶ Thus, we can form from matrix $M$ a signature matrix, in which the $i$-th column of $M$ is replaced by the minhash signature for (the set of) the $i$-th column.
Minhash signatures

- For a given collection of sets represented by their characteristic matrix $M$, the signatures are produced in the following way:
  - Pick at random some number $n$ of permutations of the rows of $M$ (let say, around 100 or 1000).
  - Call the minhash functions determined by these permutations $mh_1$, $mh_2$, $\ldots$, $mh_n$.
  - From the column representing set $S$, construct the minhash signature for $S$, the vector $(mh_1(S), mh_2(S), \ldots, mh_n(S))$ – represented as a column.
  - Thus, we can form from matrix $M$ a **signature matrix**, in which the $i$-th column of $M$ is replaced by the minhash signature for (the set of) the $i$-th column.

- The signature matrix has the same number of columns as $M$, but only $n$ rows!
Minhash signatures

- For a given collection of sets represented by their characteristic matrix $M$, the signatures are produced in the following way:
  - Pick at random some number $n$ of permutations of the rows of $M$ (let say, around 100 or 1000).
  - Call the minhash functions determined by these permutations $mh_1$, $mh_2$, $\ldots$, $mh_n$.
  - From the column representing set $S$, construct the minhash signature for $S$, the vector $(mh_1(S), mh_2(S), \ldots, mh_n(S))$ – represented as a column.
  - Thus, we can form from matrix $M$ a signature matrix, in which the $i$-th column of $M$ is replaced by the minhash signature for (the set of) the $i$-th column.

- The signature matrix has the same number of columns as $M$, but only $n$ rows!
- Even if $M$ is not represented explicitly (but as a sparse matrix by the location of its ones), it is normal for the signature matrix to be much smaller than $M$. 
• Unfortunately, it is **not** feasible to permute a large characteristic matrix explicitly.
Computing minhash signatures

- Unfortunately, it is **not** feasible to permute a large characteristic matrix explicitly.
- Even picking a random permutation of millions or billions of rows is time-consuming.
Computing minhash signatures

- Unfortunately, it is **not** feasible to permute a large characteristic matrix explicitly.
- Even picking a random permutation of millions or billions of rows is time-consuming.
- Fortunately, it is possible to simulate the effect of a random permutation by a **random hash function** that maps row numbers to as many buckets as there are rows.
Computing minhash signatures

- A hash function that maps integers 0, 1, \ldots, \kappa - 1 to bucket numbers 0 through \kappa - 1 typically will map some pairs of integers to the same bucket and leave other buckets unfilled.
Computing minhash signatures

- A hash function that maps integers 0, 1, \ldots, k − 1 to bucket numbers 0 through \( k − 1 \) typically will map some pairs of integers to the same bucket and leave other buckets unfilled.
- This difference is unimportant as long as \( k \) is large and there are not too many collisions.
Computing minhash signatures

- A hash function that maps integers 0, 1, \ldots, k − 1 to bucket numbers 0 through k − 1 typically will map some pairs of integers to the same bucket and leave other buckets unfilled.
- This difference is unimportant as long as k is large and there are not too many collisions.
- We can maintain the fiction that our hash function \( h \) **permutes** row \( r \) to position \( h(r) \) in the permuted order.
Computing minhash signatures

• Instead of picking $n$ random permutations of rows, we pick $n$ randomly chosen hash functions $h_1, h_2, \ldots, h_n$ on the rows.
Computing minhash signatures

• Instead of picking $n$ random permutations of rows, we pick $n$ randomly chosen hash functions $h_1, h_2, \ldots, h_n$ on the rows.
• We construct the signature matrix by considering each row in their given order.
Computing minhash signatures

• Instead of picking $n$ random permutations of rows, we pick $n$ randomly chosen hash functions $h_1, h_2, \ldots, h_n$ on the rows.
• We construct the signature matrix by considering each row in their given order.
• Let $SIG(i, c)$ be the element of the signature matrix for the $i$-th hash function and column $c$ defined by

$$
SIG(i, c) = \min\{h_i(r) : \text{for such } r \text{ that } c \text{ has 1 in row } r\}
$$
Computing minhash signatures

- **Example:**
  - Let us consider two hash functions $h_1$ and $h_2$:

$$h_1(r) = r + 1 \mod 5 \quad h_2(r) = 3r + 1 \mod 5$$

<table>
<thead>
<tr>
<th>Row</th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_4$</th>
<th>$h_1(r)$</th>
<th>$h_2(r)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Computing minhash signatures

• **Example:**
  - Let us consider two hash functions $h_1$ and $h_2$:

$$h_1(r) = r + 1 \mod 5 \quad h_2(r) = 3r + 1 \mod 5$$

<table>
<thead>
<tr>
<th>Row</th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_4$</th>
<th>$h_1(r)$</th>
<th>$h_2(r)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>
Computing minhash signatures

- Example:
  - The signature matrix is:

<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>$SIG(1, c)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$SIG(2, c)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We can estimate the Jaccard similarities of the underlying sets from this signature matrix:

- $\text{SIM}(S_1, S_2) = 0$
- $\text{SIM}(S_1, S_3) = \frac{1}{2}$
- $\text{SIM}(S_1, S_4) = 1$

while the true similarities are:

- $\text{SIM}(S_1, S_2) = 0$
- $\text{SIM}(S_1, S_3) = \frac{1}{4}$
- $\text{SIM}(S_1, S_4) = \frac{2}{3}$
Computing minhash signatures

- **Example:**
  - The signature matrix is:
    
    \[
    \begin{array}{cccc}
    S_1 & S_2 & S_3 & S_4 \\
    SIG(1, c) & 1 & 3 & 0 & 1 \\
    SIG(2, c) & 0 & 2 & 0 & 0 \\
    \end{array}
    \]

- We can estimate the Jaccard similarities of the underlying sets from this signature matrix:
Computing minhash signatures

- **Example:**
  - The signature matrix is:

<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>$SIG(1, c)$</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$SIG(2, c)$</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- We can estimate the Jaccard similarities of the underlying sets from this signature matrix:

\[
SIM(S_1, S_2) = 0 \quad SIM(S_1, S_3) = 1/2 \quad SIM(S_1, S_4) = 1
\]
Computing minhash signatures

- **Example:**
  
  The signature matrix is:

  \[
  \begin{array}{cccc}
  SIG(1, c) & S_1 & S_2 & S_3 & S_4 \\
  1 & 3 & 0 & 1 \\
  SIG(2, c) & 0 & 2 & 0 & 0 \\
  \end{array}
  \]

- We can estimate the Jaccard similarities of the underlying sets from this signature matrix:

  \[
  \text{SIM}(S_1, S_2) = 0 \quad \text{SIM}(S_1, S_3) = 1/2 \quad \text{SIM}(S_1, S_4) = 1
  \]

  while the true similarities are:

  \[
  \text{SIM}(S_1, S_2) = 0 \quad \text{SIM}(S_1, S_3) = 1/4 \quad \text{SIM}(S_1, S_4) = 2/3
  \]
Outline

1. Motivation
2. Shingling of Documents
3. Similarity-Preserving Summaries of Sets
4. Locality-Sensitive Hashing for Documents
5. Summary
We can use minhashing to compress large documents into small signatures and preserve the expected similarity of any pair of documents.
Locality-sensitive hashing for documents

- We can use minhashing to compress large documents into small signatures and preserve the expected similarity of any pair of documents.
- But still, it may be impossible to find the pairs with greatest similarity efficiently!!!

Example: We have a million documents and use signatures of length 250:

- Then we use 1000 bytes per document for the signatures.
- The entire data fits in a gigabyte – less than a typical main memory of a laptop.
- However, there are \( \binom{1000000}{2} \) or half a trillion pairs of documents.
- If it takes a microsecond to compute the similarity of two signatures, then it takes almost six days to compute all the similarities on that laptop.
We can use minhashing to compress large documents into small signatures and preserve the expected similarity of any pair of documents.

But still, it may be impossible to find the pairs with greatest similarity efficiently!!!

The reason is that the number of pairs of documents may be too large.
Locality-sensitive hashing for documents

• We can use minhashing to compress large documents into small signatures and preserve the expected similarity of any pair of documents.

• But still, it may be impossible to find the pairs with greatest similarity efficiently!!!

• The reason is that the number of pairs of documents may be too large.

• Example: We have a million documents and use signatures of length 250:

  ▶ Then we use 1000 bytes per document for the signatures.
  ▶ The entire data fits in a gigabyte – less than a typical main memory of a laptop.
  ▶ However, there are \((1000000^2)\) or half a trillion pairs of documents.
  ▶ If it takes a microsecond to compute the similarity of two signatures, then it takes almost six days to compute all the similarities on that laptop.
• We can use minhashing to compress large documents into small signatures and preserve the expected similarity of any pair of documents.

• But still, it may be impossible to find the pairs with greatest similarity efficiently!!!

• The reason is that the number of pairs of documents may be too large.

• **Example:** We have a million documents and use signatures of length 250:
  ▶ Then we use 1000 bytes per document for the signatures.
Locality-sensitive hashing for documents

- We can use minhashing to compress large documents into small signatures and preserve the expected similarity of any pair of documents.
- But still, it may be impossible to find the pairs with greatest similarity efficiently!!!
- The reason is that the number of pairs of documents may be too large.
- **Example**: We have a million documents and use signatures of length 250:
  - Then we use 1000 bytes per document for the signatures.
  - The entire data fits in a gigabyte – less than a typical main memory of a laptop.
Locality-sensitive hashing for documents

- We can use minhashing to compress large documents into small signatures and preserve the expected similarity of any pair of documents.
- But still, it may be impossible to find the pairs with greatest similarity efficiently!!!
- The reason is that the number of pairs of documents may be too large.
- **Example:** We have a million documents and use signatures of length 250:
  - Then we use 1000 bytes per document for the signatures.
  - The entire data fits in a gigabyte – less than a typical main memory of a laptop.
  - However, there are $\binom{1,000,000}{2}$ or half a trillion pairs of documents.
Locality-sensitive hashing for documents

• We can use minhashing to compress large documents into small signatures and preserve the expected similarity of any pair of documents.

• But still, it may be impossible to find the pairs with greatest similarity efficiently!!!

• The reason is that the number of pairs of documents may be too large.

• **Example:** We have a million documents and use signatures of length 250:
  
  ▶ Then we use 1000 bytes per document for the signatures.
  ▶ The entire data fits in a gigabyte – less than a typical main memory of a laptop.
  ▶ However, there are \(\binom{1000000}{2}\) or half a trillion pairs of documents.
  ▶ If it takes a microsecond to compute the similarity of two signatures, then it takes almost six days to compute all the similarities on that laptop.
• However, often we want only the most similar pairs or all pairs that are above some lower bound in similarity.

Locality-sensitive hashing for documents
Locality-sensitive hashing for documents

- However, often we want only the most similar pairs or all pairs that are above some lower bound in similarity.
- If so, then we need to focus our attention only on pairs that are likely to be similar, without investigating every pair.
• However, often we want only the most similar pairs or all pairs that are above some lower bound in similarity.
• If so, then we need to focus our attention only on pairs that are likely to be similar, without investigating every pair.
• A technique called **locality-sensitive hashing (LSH)** is a solution for this problem.
• General idea of LSH:

- Hash items several times, in such a way that similar items are more likely to be hashed to the same bucket than dissimilar items are.
- Any pair that hashed to the same bucket for any of the hashings is a candidate pair.
- We check only the candidate pairs for similarity.
- The hope is that most of the dissimilar pairs will never hash to the same bucket, and therefore will never be checked.
- Those dissimilar pairs that do hash to the same bucket are false positives.
- The truly similar pairs that will not hash to the same bucket under at least one of the hash functions are false negatives.
- We hope to have a small fraction of false positives and false negatives.
• General idea of LSH:
  ▶ Hash items several times, in such a way that similar items are more likely to be hashed to the same bucket than dissimilar items are.
General idea of LSH:

- Hash items several times, in such a way that similar items are more likely to be hashed to the same bucket than dissimilar items are.
- Any pair that hashed to the same bucket for any of the hashings is a candidate pair.

The hope is that most of the dissimilar pairs will never hash to the same bucket, and therefore will never be checked.

Those dissimilar pairs that do hash to the same bucket are false positives.

The truly similar pairs that will not hash to the same bucket under at least one of the hash functions are false negatives.

We hope to have a small fraction of false positives and false negatives.
• General idea of LSH:
  ▶ Hash items several times, in such a way that similar items are more likely to be hashed to the same bucket than dissimilar items are.
  ▶ Any pair that hashed to the same bucket for any of the hashings is a candidate pair.
  ▶ We check only the candidate pairs for similarity.
LSH

• General idea of LSH:
  ▶ Hash items several times, in such a way that similar items are more likely to be hashed to the same bucket than dissimilar items are.
  ▶ Any pair that hashed to the same bucket for any of the hashings is a candidate pair.
  ▶ We check only the candidate pairs for similarity.

• The hope is that most of the dissimilar pairs will never hash to the same bucket, and therefore will never be checked.
**LSH**

- **General idea of LSH:**
  - Hash items several times, in such a way that similar items are more likely to be hashed to the same bucket than dissimilar items are.
  - Any pair that hashed to the same bucket for any of the hashings is a candidate pair.
  - We check only the candidate pairs for similarity.

- **The hope is that most of the dissimilar pairs will never hash to the same bucket, and therefore will never be checked.**

- **Those dissimilar pairs that do hash to the same bucket are false positives.**
LSH

• General idea of LSH:
  ▶ Hash items several times, in such a way that similar items are more likely to be hashed to the same bucket than dissimilar items are.
  ▶ Any pair that hashed to the same bucket for any of the hashings is a candidate pair.
  ▶ We check only the candidate pairs for similarity.

• The hope is that most of the dissimilar pairs will never hash to the same bucket, and therefore will never be checked.

• Those dissimilar pairs that do hash to the same bucket are false positives.

• The truly similar pairs that will not hash to the same bucket under at least one of the hash functions are false negatives.
• General idea of LSH:
  ▶ Hash items several times, in such a way that similar items are more likely to be hashed to the same bucket than dissimilar items are.
  ▶ Any pair that hashed to the same bucket for any of the hashings is a candidate pair.
  ▶ We check only the candidate pairs for similarity.
• The hope is that most of the dissimilar pairs will never hash to the same bucket, and therefore will never be checked.
• Those dissimilar pairs that do hash to the same bucket are false positives.
• The truly similar pairs that will not hash to the same bucket under at least one of the hash functions are false negatives.
• We hope to have a small fraction of false positives and false negatives.
LSH for minhash signatures

- For minhash signatures divide the signature matrix into $b$ bands consisting of $r$ rows each.
LSH for minhash signatures

- For minhash signatures divide the signature matrix into \( b \) bands consisting of \( r \) rows each.
- For each band use a hash function that takes vectors of \( r \) integers (the portion of one column within that band) and hashes them to some large number of buckets.

\[
\begin{array}{c|cccccc}
\text{band 1} & \ldots & 1 & 0 & 0 & 0 & 2 & \ldots \\
\ldots & 3 & 2 & 1 & 2 & 2 & \ldots \\
\ldots & 0 & 1 & 3 & 1 & 1 & \ldots \\
\end{array}
\begin{array}{c|cccccc}
\text{band 2} & \ldots & 5 & 3 & 5 & 1 & 3 & \ldots \\
\ldots & 1 & 4 & 1 & 2 & 4 & \ldots \\
\ldots & 6 & 1 & 6 & 1 & 1 & \ldots \\
\end{array}
\begin{array}{c|cccccc}
\text{band 3} & \ldots & 3 & 1 & 4 & 6 & 6 & \ldots \\
\ldots & 3 & 1 & 1 & 6 & 6 & \ldots \\
\ldots & 2 & 5 & 3 & 4 & 4 & \ldots \\
\end{array}
\]

We assume that the chances of an accidental collision to be very small.
LSH for minhash signatures

• For minhash signatures divide the signature matrix into $b$ bands consisting of $r$ rows each.

• For each band use a hash function that takes vectors of $r$ integers (the portion of one column within that band) and hashes them to some large number of buckets.

<table>
<thead>
<tr>
<th></th>
<th>band 1</th>
<th></th>
<th>band 2</th>
<th></th>
<th>band 3</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>...</td>
<td>1 0 0 0 0 2</td>
<td>...</td>
<td>5 3 5 1 3</td>
<td>...</td>
<td>3 1 4 6 6</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>3 2 1 2 2</td>
<td>...</td>
<td>1 4 1 2 4</td>
<td>...</td>
<td>3 1 1 6 6</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>0 1 3 1 1</td>
<td>...</td>
<td>6 1 6 1 1</td>
<td>...</td>
<td>2 5 3 4 4</td>
</tr>
</tbody>
</table>

• We assume that the chances of an accidental collision to be very small.
Analysis of the banding technique

• Suppose we use $b$ bands of $r$ rows each and that a particular pair of documents have Jaccard similarity $s$. 

$\text{Conclusion}$

The probability that two documents become a candidate pair is:

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

because of the following reasoning:

▶ The probability that the signatures agree in all rows of one particular band is $s^r$.
▶ The probability that the signatures do not agree in at least one row of a particular band is $1 - s^r$.
▶ The probability that the signatures do not agree in all rows of any of the bands is $(1 - s^r)^b$.
▶ The probability that the signatures agree in all the rows of at least one band, and therefore become a candidate pair, is $1 - (1 - s^r)^b$. 

36 / 43
Analysis of the banding technique

• Suppose we use $b$ bands of $r$ rows each and that a particular pair of documents have Jaccard similarity $s$.

• Recall that the probability the minhash signatures for these documents agree in any one particular row of the signature matrix is $s$.
Analysis of the banding technique

- Suppose we use $b$ bands of $r$ rows each and that a particular pair of documents have Jaccard similarity $s$.
- Recall that the probability the minhash signatures for these documents agree in any one particular row of the signature matrix is $s$.
- The probability that two documents become a candidate pair is:
Analysis of the banding technique

• Suppose we use $b$ bands of $r$ rows each and that a particular pair of documents have Jaccard similarity $s$.

• Recall that the probability the minhash signatures for these documents agree in any one particular row of the signature matrix is $s$.

• The probability that two documents become a candidate pair is:

$$1 - (1 - s^r)^b,$$
Analysis of the banding technique

- Suppose we use $b$ bands of $r$ rows each and that a particular pair of documents have Jaccard similarity $s$.
- Recall that the probability the minhash signatures for these documents agree in any one particular row of the signature matrix is $s$.
- The probability that two documents become a candidate pair is:

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

because of the following reasoning:
Analysis of the banding technique

• Suppose we use $b$ bands of $r$ rows each and that a particular pair of documents have Jaccard similarity $s$.

• Recall that the probability the minhash signatures for these documents agree in any one particular row of the signature matrix is $s$.

• The probability that two documents become a candidate pair is:

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

because of the following reasoning:

▶ The probability that the signatures agree in all rows of one particular band is
Analysis of the banding technique

- Suppose we use $b$ bands of $r$ rows each and that a particular pair of documents have Jaccard similarity $s$.
- Recall that the probability the minhash signatures for these documents agree in any one particular row of the signature matrix is $s$.
- The probability that two documents become a candidate pair is:

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

because of the following reasoning:

- The probability that the signatures agree in all rows of one particular band is $s^r$.
Analysis of the banding technique

- Suppose we use $b$ bands of $r$ rows each and that a particular pair of documents have Jaccard similarity $s$.
- Recall that the probability the minhash signatures for these documents agree in any one particular row of the signature matrix is $s$.
- The probability that two documents become a candidate pair is:

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

because of the following reasoning:

- The probability that the signatures agree in all rows of one particular band is $s^r$.
- The probability that the signatures do not agree in at least one row of a particular band is
Analysis of the banding technique

• Suppose we use $b$ bands of $r$ rows each and that a particular pair of documents have Jaccard similarity $s$.

• Recall that the probability the minhash signatures for these documents agree in any one particular row of the signature matrix is $s$.

• The probability that two documents become a candidate pair is:

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

because of the following reasoning:

▶ The probability that the signatures agree in all rows of one particular band is $s^r$.

▶ The probability that the signatures do not agree in at least one row of a particular band is $1 - s^r$. 
Analysis of the banding technique

- Suppose we use $b$ bands of $r$ rows each and that a particular pair of documents have Jaccard similarity $s$.
- Recall that the probability the minhash signatures for these documents agree in any one particular row of the signature matrix is $s$.
- The probability that two documents become a candidate pair is:

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

because of the following reasoning:

- The probability that the signatures agree in all rows of one particular band is $s^r$.
- The probability that the signatures do not agree in at least one row of a particular band is $1 - s^r$.
- The probability that the signatures do not agree in all rows of any of the bands is
Analysis of the banding technique

• Suppose we use $b$ bands of $r$ rows each and that a particular pair of documents have Jaccard similarity $s$.

• Recall that the probability the minhash signatures for these documents agree in any one particular row of the signature matrix is $s$.

• The probability that two documents become a candidate pair is:

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

because of the following reasoning:

▶ The probability that the signatures agree in all rows of one particular band is $s^r$.

▶ The probability that the signatures do not agree in at least one row of a particular band is $1 - s^r$.

▶ The probability that the signatures do not agree in all rows of any of the bands is $(1 - s^r)^b$. 
Analysis of the banding technique

- Suppose we use \( b \) bands of \( r \) rows each and that a particular pair of documents have Jaccard similarity \( s \).
- Recall that the probability the minhash signatures for these documents agree in any one particular row of the signature matrix is \( s \).
- The probability that two documents become a candidate pair is:

\[
1 - (1 - s^r)^b ,
\]

because of the following reasoning:
- The probability that the signatures agree in all rows of one particular band is \( s^r \).
- The probability that the signatures do not agree in at least one row of a particular band is \( 1 - s^r \).
- The probability that the signatures do not agree in all rows of any of the bands is \((1 - s^r)^b\).
- The probability that the signatures agree in all the rows of at least one band, and therefore become a candidate pair, is
Analysis of the banding technique

• Suppose we use $b$ bands of $r$ rows each and that a particular pair of documents have Jaccard similarity $s$.

• Recall that the probability the minhash signatures for these documents agree in any one particular row of the signature matrix is $s$.

• The probability that two documents become a candidate pair is:

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

because of the following reasoning:

► The probability that the signatures agree in all rows of one particular band is $s^r$.

► The probability that the signatures do not agree in at least one row of a particular band is $1 - s^r$.

► The probability that the signatures do not agree in all rows of any of the bands is $(1 - s^r)^b$.

► The probability that the signatures agree in all the rows of at least one band, and therefore become a candidate pair, is $1 - (1 - s^r)^b$. 
Analysis of the banding technique

- The probability that two documents become a candidate pair has a form of an S-curve.

```python
>>> import numpy as np
>>> import matplotlib.pyplot as plt
>>> s = np.arange(0., 1., 0.05)
>>> plt.plot(s, 1-(1-s**4)**16, 'r--')
>>> plt.show()
```
Analysis of the banding technique

- The threshold, the value of similarity $s$ at which the rise becomes steepest, is a function of $b$ and $r$.
- Use `sympy` to compute the threshold:

```
>>> from sympy import *
>>> s, r, b=Symbol( 's' ), Symbol( 'r' ), Symbol( 'b' )
>>> d = diff(1-(1-s**r)**b, s, 2)
>>> solve(d, s)
[(((r - 1)/(b*r - 1)))**(1/r)]
```

- An approximation to the threshold is $(1/b)^{1/r}$.
- **Example**: for $b = 16$ and $r = 4$, the threshold is approximately $1/2$. 

Analysis of the banding technique

- Example:

  Consider the case for $b = 20$ and $r = 5$ (we have signatures of length 100)

  For $s = 0.8$, $s^r = 0.672$, and $(1 - s^r)^b = 0.00035$.

  Interpretation:
  - If we consider two documents with similarity 0.8, then in any one band, they have only about 33% chance of becoming a candidate pair.
  - However, there are 20 bands and thus 20 chances to become a candidate.
  - That is why the final probability is 0.99965 (since the probability of a false negative is 0.00035).
• **Example:**
  - Consider the case for $b = 20$ and $r = 5$ (we have signatures of length 100)
Analysis of the banding technique

• Example:
  ▶ Consider the case for $b = 20$ and $r = 5$ (we have signatures of length 100)
  ▶ For $s = 0.8$, $1 - s^r = 0.672$, and $(1 - s^r)^b = 0.00035$. 
Analysis of the banding technique

• Example:
  ▶ Consider the case for $b = 20$ and $r = 5$ (we have signatures of length 100)
  ▶ For $s = 0.8$, $1 - s^r = 0.672$, and $(1 - s^r)^b = 0.00035$.
  ▶ Interpretation:
• **Example:**
  - Consider the case for $b = 20$ and $r = 5$ (we have signatures of length 100)
  - For $s = 0.8$, $1 - s^r = 0.672$, and $(1 - s^r)^b = 0.00035$.
  - Interpretation:
    - If we consider two documents with similarity 0.8, then in any one band, they have only about 33% chance of becoming a candidate pair.
Analysis of the banding technique

- **Example:**
  - Consider the case for $b = 20$ and $r = 5$ (we have signatures of length 100)
  - For $s = 0.8$, $1 - s^r = 0.672$, and $(1 - s^r)^b = 0.00035$.
  - Interpretation:
    - If we consider two documents with similarity 0.8, then in any one band, they have only about 33% chance of becoming a candidate pair.
    - However, there are 20 bands and thus 20 chances to become a candidate.
Example:

- Consider the case for $b = 20$ and $r = 5$ (we have signatures of length 100)
- For $s = 0.8$, $1 - s^r = 0.672$, and $(1 - s^r)^b = 0.00035$.
- Interpretation:
  - If we consider two documents with similarity 0.8, then in any one band, they have only about 33% chance of becoming a candidate pair.
  - However, there are 20 bands and thus 20 chances to become a candidate.
  - That is why the final probability is 0.99965 (since the probability of a false negative is 0.00035).
Procedure for finding similar documents

• Choose a threshold $t$ that defines how similar documents have to be in order for them to be regarded as a desired “similar pair.”
Procedure for finding similar documents

• Choose a threshold $t$ that defines how similar documents have to be in order for them to be regarded as a desired “similar pair.”

• Pick a number of bands $b$ and a number of rows $r$ such that $br = n$, and the threshold $t$ is approximately $(1/b)^{1/r}$.
Procedure for finding similar documents

- Choose a threshold $t$ that defines how similar documents have to be in order for them to be regarded as a desired “similar pair.”
- Pick a number of bands $b$ and a number of rows $r$ such that $br = n$, and the threshold $t$ is approximately $(1/b)^{1/r}$.
- If avoidance of false negatives is important, you may wish to select $b$ and $r$ to produce a threshold lower than $t$. 
Procedure for finding similar documents

- Choose a threshold $t$ that defines how similar documents have to be in order for them to be regarded as a desired “similar pair.”
- Pick a number of bands $b$ and a number of rows $r$ such that $br = n$, and the threshold $t$ is approximately $(1/b)^{1/r}$.
- If avoidance of false negatives is important, you may wish to select $b$ and $r$ to produce a threshold lower than $t$.
- If speed is important and you wish to limit false positives, select $b$ and $r$ to produce a higher threshold.
1 Motivation

2 Shingling of Documents

3 Similarity-Preserving Summaries of Sets

4 Locality-Sensitive Hashing for Documents

5 Summary
Summary

- Similarity of documents.
- Jaccard similarity.
- Minhash technique.
- Locality-Sensitive Hashing for Documents.

• P. Indyk. Algorithms for nearest neighbor search