Finding Similar Items I

Alexander Schönhuth

UNIVERSITÄT BIELEFELD
Faculty of Technology

Bielefeld University
April 30, 2020

TODAY

Features Today

- Lecture will be *recorded*, edited and posted
- *Arsnova:*
  - Session ID: 50395809 (also pasted into Zoom chat today)
  - Use Case: “Questions / Comments from Audience” (German: “Kummerkasten”)
  - Session will be active throughout semester
  - I will (be happy to) respond whenever I can

Learning Goals

- Turning documents into sets ☞ shingles
- Computing the similarity of sets ☞ minhashing

Finding Similar Items: Introduction

FINDING SIMILAR ITEMS

Fundamental problem in data mining: retrieve pairs of similar elements of a dataset.

Applications

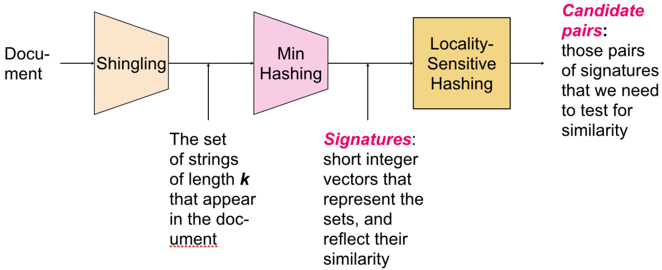
- Detecting plagiarism in a set of documents
- Identifying near-identical mirror pages during web searches
- Identifying documents from the same author
- *Collaborative Filtering*
  - Online Purchases (Amazon: suggestions based on ′similar′ customers)
  - Movie Ratings (Netflix: suggestions based on ′similar′ users)

ISSUES

Consider a dataset of N items, for example: N webpages or N text documents.

- Comparing all items requires $O(N^2)$ runtime.
  - Ok for small N.
  - If $N \approx 10^6$, we have 10^{12} comparisons. Maybe not OK!
- How to efficiently compute similarity if items themselves are large?
- Similarity works well for sets of items. How to turn data into sets of items?

Overview



From `mmds.org`

- *Shingling:* turning text files into sets
- *Minhashing:* computing similarity for large sets
- *Locality Sensitive Hashing:* avoids $O(N^2)$ comparisons by determining candidate pairs

Shingles

–

Turning Documents into Sets

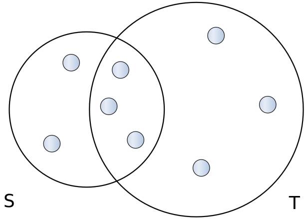
Jaccard Similarity

Definition [Jaccard Similarity]

Consider two sets S and T. The *Jaccard similarity* $\mathrm{SIM}(S,T)$ is defined as

$$\mathrm{SIM}(S,T) = \frac{|S \cap T|}{|S \cup T|} \tag{1}$$

the ratio of elements in the intersection and in the union of S and T.



$\mathrm{SIM}(S,T) = \frac{3}{8}$

SHINGLES: DEFINITION

- Document = large string of characters
- *k-shingle:* a substring of a particular length k
- *Idea: A document is set of k-shingles*
- *Example:* document = "*acadacc*$''$, k-shingles:

$$\{ac, ad, ca, cc, da\}$$

- We can now compute *Jaccard similarity* for two documents by considering them as sets of shingles.
- *Example:* documents $D_1 = "abcd", D_2 = "dbcd"$ using 2-shingles yields $D_1 = \{ab, bc, cd\}, D_2 = \{bc, cd, db\}$, so $\text{SIM}(D_1, D_2) = \frac{|\{bc,cd\}|}{|\{ab,bc,cd,db\}|} = 2/4 = 1/2$

Shingles: Definition

- Issue: Determining right size of k.
  - k large enough such that any particular k-shingle appears in document with low probability ($k = 5$, yielding 256^5 different shingles on 256 different characters, ok for emails)
  - too large k yields too large universe of elements (example: $k = 9$ means $256^9 = (2^8)^9 = 2^{72}$ on the order of number of atoms in the universe)
- Solution if necessary k is too large: hash shingles to buckets, such that buckets are evenly covered, and collisions are rare
- We would like to compute Jaccard similarity for pairs of sets
- *But:* even when hashed, size of the universe of elements (= # buckets when hashed) may be prohibitive to do that fast
- What to do?

Minhashing
–
Rapidly Computing Similarity of Sets

SETS AS BITVECTORS

- Representing sets as bitvectors
  - Length of bitvectors is size of universal set
  - For example, when hashed, length of bitvector = number of buckets
  - Entries zero if *element not in set*, one if *element in set*
- Does not reflect to really store the sets, but nice visualization

Sets as Bitvectors: the Characteristic Matrix

Definition [Characteristic Matrix]

Given C sets over a universe R, the *characteristic matrix* $M \in \{0,1\}^{|R| \times |C|}$ is defined to have entries

$$M(r,c) = \begin{cases} 0 & \text{if } r \notin c \\ 1 & \text{if } r \in c \end{cases} \tag{2}$$

for $r \in R, c \in C$.

Element	S_1	S_2	S_3	S_4
a	1	0	0	1
b	0	0	1	0
c	0	1	0	1
d	1	0	1	1
e	0	0	1	0

Characteristic matrix of four sets (S_1, S_2, S_3, S_4) over universal set $\{a, b, c, d, e\}$
From `mmds.org`

PERMUTATIONS

DEFINITION [BIJECTION,PERMUTATION]

- A *bijection* is a map $\pi : S \to S$ such that
  - $\pi(x) = \pi(y)$ implies $x = y$ (π is *injective*)
  - For all $y \in S$ there is $x \in S$ such that $\pi(x) = y$ (π is surjective)
- A *permutation* is a bijection

$$\pi : \{1, ..., m\} \to \{1, ..., m\} \tag{3}$$

Example: A permutation on $\{1, 2, 3, 4, 5\}$ may map

$$1 \to 4, 2 \to 3, 3 \to 1, 4 \to 5 \text{ and } 5 \to 2$$

Permuting Rows of Characteristic Matrix

Element	S_1	S_2	S_3	S_4
a	1	0	0	1
b	0	0	1	0
c	0	1	0	1
d	1	0	1	1
e	0	0	1	0

Element	S_1	S_2	S_3	S_4
b	0	0	1	0
e	0	0	1	0
a	1	0	0	1
d	1	0	1	1
c	0	1	0	1

A characteristic matrix of four sets (S_1, S_2, S_3, S_4) over universal set $\{a, b, c, d, e\}$ and a permutation of its rows $1 \to 3, 2 \to 1, 3 \to 5, 4 \to 4, 5 \to 2$

MINHASH - DEFINITION

Given

- a characteristic matrix with m rows and a column S
- a permutation π on the rows, that is $\pi : \{1, ..., m\} \rightarrow \{1, ..., m\}$ is a bijection

DEFINITION [MINHASH]
The *minhash* function h_π on S is defined by

$$h_\pi(S) = \min_{i \in \{1,\dots,m\}} \{\pi(i) \mid S[i] = 1\}$$

MINHASH - DEFINITION

DEFINITION [MINHASH]

The *minhash* function h_π on S is defined by

$$h_\pi(S) = \min_{i \in \{1,\dots,m\}} \{\pi(i) \mid S[i] = 1\}$$

EXPLANATION

The minhash of a column S relative to permutation π is

- after reordering rows according to the permutation π
- the first row in which a one in S appears

Minhash - Example

Example

Let

- 1 corresponds to a, 2 to b, ...
- $\pi : 1 \to 3, 2 \to 1, 3 \to 5, 4 \to 4, 5 \to 2$ and

Element	S_1	S_2	S_3	S_4
b	0	0	1	0
e	0	0	1	0
a	1	0	0	1
d	1	0	1	1
c	0	1	0	1

$$h_\pi(S_1) = 3, h_\pi(S_2) = 5, h_\pi(S_3) = 1, h_\pi(S_4) = 3$$

Minhashing and Jaccard Similarity

Given

- two columns (sets) S_1, S_2 of a characteristic matrix
- a randomly picked permutation π on the rows (on $\{1, ..., m\}$)

Theorem [Minhash and Jaccard Similarity]:
The probability that $h_\pi(S_1) = h_\pi(S_2)$ is $\text{SIM}(S_1, S_2)$.

MINHASH AND JACCARD SIMILARITY - PROOF

THEOREM [MINHASH AND JACCARD SIMILARITY]:
The probability that $h_\pi(S_1) = h_\pi(S_2)$ is $\text{SIM}(S_1, S_2)$.

PROOF.
Distinguish three different classes of rows:

- *Type X rows* have a 1 in both S_1, S_2
- *Type Y rows* have a 1 in only one of S_1, S_2
- *Type Z rows* have a 0 in both S_1, S_2

Let x be the number of type X rows and y the number of type Y rows.

- So $x = |S_1 \cap S_2|$ and $x + y = |S_1 \cup S_2|$
- Hence

$$\text{SIM}(S_1, S_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|} = \frac{x}{x+y} \tag{4}$$

MinHash and Jaccard Similarity - Proof

Proof. (cont.)

- Consider the *probability* that $h(S_1) = h(S_2)$
- Imagine rows to be permuted randomly, and proceed from the top
- The probability to encounter type X before type Y is

$$\frac{x}{x+y} \tag{5}$$

  - If first non type Z row is type X, then $h(S_1) = h(S_2)$
  - If first non type Z row is type Y, then $h(S_1) \neq h(S_2)$
- So $h(S_1) = h(S_2)$ happens with probability (5), which by (4) concludes the proof.

□

MinHash - Intermediate Summary / Expansion of Idea

- Computing a minhash means turning a set into one number
- For different sets, numbers agree with probability equal to their Jaccard similarity.
- Can we expand on this idea? Can we compute (ensembles of) numbers that enable us to determine their Jaccard similarity?
- Immediate idea: compute several minhashes. The fraction of times the minhashes of two sets agree equals their Jaccard similarity.
- Several sufficiently well chosen minhashes yield a *minhash signature*.

MINHASH SIGNATURES

Consider

- the m rows of the characteristic matrix
- n permutations $\{1, ..., m\} \to \{1, ..., m\}$
- the corresponding *minhash* functions
  $h_1, ..., h_n : \{0, 1\}^m \to \{1, ..., m\}$
- and a particular column $S \in \{0, 1\}^m$
  ☞ $h_i(S) \in \{1, ..., m\}$ for any $1 \leq i \leq n$

DEFINITION [MINHASH SIGNATURE]
The *minhash signature* SIG_S of S given $h_1, ..., h_n$ is the array

$$[h_1(S), ..., h_n(S)] \in \{1, ..., m\}^n$$

Minhash Signatures

Definition [Minhash Signature]
The *minhash signature* SIG_S of S given $h_1, ..., h_n$ is the array

$$[h_1(S), ..., h_n(S)] \in \{1, ..., m\}^n$$

Meaning: Computing the minhash signature for a column S turns

- the binary-valued array of length m that represents S
  $\leftrightarrow S \in \{0, 1\}^m$
- into an m-valued array of length n
  $\leftrightarrow [h_1(S), ..., h_n(S)] \in \{1, ..., m\}^n$

Because $n < m$ (often $n << m$), the minhash signature is a *reduced representation of a set*.

Signature Matrix

Consider a characteristic matrix, and n permutations $h_1, ..., h_n$.

Definition [Signature Matrix]
The signature matrix SIG is a matrix with n rows and as many columns as the characteristic matrix (i.e. the number of sets), where entries SIG_{ij} are defined by

$$SIG_{ij} = h_i(S_j) \tag{6}$$

where S_j refers to the j-th column in the characteristic matrix.

Signature Matrices: Facts

Let M be a signature matrix.

- Because usually $n << m$, that is n is much smaller than m, a signature matrix is much smaller than the original characteristic matrix.
- The probability that $SIG_{ij_1} = SIG_{ij_2}$ for two sets S_{j_1}, S_{j_2} equals the Jaccard similarity $\text{SIM}(S_{j_1}, S_{j_2})$
- The expected number of rows where columns j_1, j_2 agree, divided by n, is $\text{SIM}(S_{j_1}, S_{j_2})$.

SIGNATURE MATRICES: ISSUES

Issue:

- For large m, it is time-consuming / storage-intense to determine permutations

$$\pi : \{1, ..., m\} \to \{1, ..., m\}$$

- Re-sorting rows relative to a permutation is even more expensive

Solution:

- Instead of permutations, use hash functions (watch the index shift!)

$$h : \{0, ..., m-1\} \to \{0, ..., m-1\}$$

- Likely, a hash function is not a bijection, so at times
  - places two rows in the same bucket
  - leaves other buckets empty
- Effects are negligible for our purposes, however

Computing Signature Matrices in Practice

- Consider n hash functions $h_i : \{0, ..., m-1\} \to \{0, ..., m-1\}, i = 1, ..., n$
- Let r and c index rows and columns in the characteristic matrix $M \in \{0,1\}^{m \times |C|}$
- So c also index columns, while i indexes rows in the signature matrix $SIG \in \{1, ..., m\}^{n \times |C|}$

```
for each c do
    for 0 ≤ i ≤ n do
        SIG(i, c) = ∞
    end for
end for
for each row r do
    for each column c do
        if M(r, c) = 1 then
            for i=1 to n do
                SIG(i, c) =
                min(SIG(i, c), h_i(r))
            end for
        end if
    end for
end for
```

Computing Signature Matrices: Example

Row	S_1	S_2	S_3	S_4	$x + 1 \mod 5$	$3x + 1 \mod 5$
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

Hash functions computed for a characteristic matrix, with rows indexed from 0 to 4

Computing Signature Matrices in Practice

- Consider n hash functions $h_i : \{0, ..., m-1\} \rightarrow \{0, ..., m-1\}, i = 1, ..., n$
- Let r and c index rows and columns in the characteristic matrix $M \in \{0, 1\}^{m \times |C|}$
- So c also index columns, while i indexes rows in the signature matrix $SIG \in \{1, ..., m\}^{n \times |C|}$

```
for each c do
    for 0 ≤ i ≤ n do
        SIG(i, c) = ∞
    end for
end for
for each row r do
    for each column c do
        if M(r, c) = 1 then
            for i=1 to n do
                SIG(i, c) =
                min(SIG(i, c), h_i(r))
            end for
        end if
    end for
end for
```

Computing Signature Matrices: Example

Row	S_1	S_2	S_3	S_4	$x+1 \mod 5$	$3x+1 \mod 5$
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

Hash functions computed for a characteristic matrix, with rows indexed from 0 to 4

	S_1	S_2	S_3	S_4
h_1	∞	∞	∞	∞
h_2	∞	∞	∞	∞

Signature matrix *SIG*: after initialization

Computing Signature Matrices in Practice

- Consider n hash functions $h_i : \{0, ..., m-1\} \rightarrow \{0, ..., m-1\}, i = 1, ..., n$
- Let r and c index rows and columns in the characteristic matrix $M \in \{0,1\}^{m \times |C|}$
- So c also index columns, while i indexes rows in the signature matrix $SIG \in \{1, ..., m\}^{n \times |C|}$

```
for each c do
    for 0 ≤ i ≤ n do
        SIG(i, c) = ∞
    end for
end for
for each row r do
    for each column c do
        if M(r, c) = 1 then
            for i=1 to n do
                SIG(i, c) =
                min(SIG(i, c), h_i(r))
            end for
        end if
    end for
end for
```

Computing Signature Matrices in Practice

- Consider n hash functions $h_i : \{0, ..., m-1\} \rightarrow \{0, ..., m-1\}, i = 1, ..., n$
- Let r and c index rows and columns in the characteristic matrix $M \in \{0,1\}^{m \times |C|}$
- So c also index columns, while i indexes rows in the signature matrix $SIG \in \{1, ..., m\}^{n \times |C|}$

```
for each c do
    for 0 ≤ i ≤ n do
        SIG(i, c) = ∞
    end for
end for
for each row r do
    // Iteration 1: first row
    for each column c do
        if M(r, c) = 1 then
            for i=1 to n do
                SIG(i, c) =
                min(SIG(i, c), h_i(r))
            end for
        end if
    end for
    // End first row
end for
```

Computing Signature Matrices: Example

Row	S_1	S_2	S_3	S_4	$x+1 \mod 5$	$3x+1 \mod 5$
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

Hash functions computed for a characteristic matrix, with rows indexed from 0 to 4

First iteration: row # 0 has 1's in S_1 and S_4, so put $SIG_{11} = SIG_{14} = 0+1 \mod 5 = 1$ and $SIG_{21} = SIG_{24} = 3 \cdot 0 + 1 \mod 5 = 1$

	S_1	S_2	S_3	S_4
h_1	1	∞	∞	1
h_2	1	∞	∞	1

Signature matrix after considering first row

Computing Signature Matrices in Practice

- Consider n hash functions $h_i : \{0, ..., m-1\} \to \{0, ..., m-1\}, i = 1, ..., n$
- Let r and c index rows and columns in the characteristic matrix $M \in \{0,1\}^{m \times |C|}$
- So c also index columns, while i indexes rows in the signature matrix $SIG \in \{1, ..., m\}^{n \times |C|}$

```
for each c do
    for 0 ≤ i ≤ n do
        SIG(i, c) = ∞
    end for
end for
for each row r do
    // Iteration 2: second row
    for each column c do
        if M(r, c) = 1 then
            for i=1 to n do
                SIG(i, c) =
                min(SIG(i, c), h_i(r))
            end for
        end if
    end for
    // End second row
end for
```

Computing Signature Matrices: Example

Row	S_1	S_2	S_3	S_4	$x+1 \mod 5$	$3x+1 \mod 5$
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

Hash functions computed for a characteristic matrix, with rows indexed from 0 to 4

Second iteration: row #1 has 1 in S_3, so put $SIG_{31} = 1+1 \mod 5 = 2$ and $SIG_{32} = 3+1 \mod 5 = 4$.

	S_1	S_2	S_3	S_4
h_1	1	∞	2	1
h_2	1	∞	4	1

Signature matrix M after considering second row

Computing Signature Matrices in Practice

- Consider n hash functions $h_i : \{0, ..., m-1\} \rightarrow \{0, ..., m-1\}, i = 1, ..., n$
- Let r and c index rows and columns in the characteristic matrix $M \in \{0,1\}^{m \times |C|}$
- So c also index columns, while i indexes rows in the signature matrix $SIG \in \{1, ..., m\}^{n \times |C|}$

```
for each c do
    for 0 ≤ i ≤ n do
        SIG(i, c) = ∞
    end for
end for
for each row r do
    // Iteration 3: third row
    for each column c do
        if M(r, c) = 1 then
            for i=1 to n do
                SIG(i, c) =
                min(SIG(i, c), h_i(r))
            end for
        end if
    end for
    // End third row
end for
```

Computing Signature Matrices: Example

Row	S_1	S_2	S_3	S_4	$x+1 \mod 5$	$3x+1 \mod 5$
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

Hash functions computed for a characteristic matrix, with rows indexed from 0 to 4

Third iteration: row # 2 has 1's in S_2 and S_4, so put $SIG_{21} = 2+1 \mod 5 = 3$, $SIG_{41} = \min(1, 2+1 \mod 5 = 3) = 1$ and $SIG_{22} = 6+1 \mod 5 = 2$ and $SIG_{42} = \min(1, 6+1 \mod 5 = 2) = 1$

	S_1	S_2	S_3	S_4
h_1	1	3	2	1
h_2	1	2	4	1

Signature matrix after considering third row

Computing Signature Matrices in Practice

- Consider n hash functions $h_i : \{0, ..., m-1\} \rightarrow \{0, ..., m-1\}, i = 1, ..., n$
- Let r and c index rows and columns in the characteristic matrix $M \in \{0,1\}^{m \times |C|}$
- So c also index columns, while i indexes rows in the signature matrix $SIG \in \{1, ..., m\}^{n \times |C|}$

```
for each c do
    for 0 ≤ i ≤ n do
        SIG(i, c) = ∞
    end for
end for
for each row r do
    // Iteration 4: fourth row
    for each column c do
        if M(r, c) = 1 then
            for i=1 to n do
                SIG(i, c) =
                min(SIG(i, c), h_i(r))
            end for
        end if
    end for
    // End fourth row
end for
```

UNIVERSITÄT BIELEFELD

e

Computing Signature Matrices: Example

Row	S_1	S_2	S_3	S_4	$x+1 \mod 5$	$3x+1 \mod 5$
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

Hash functions computed for a characteristic matrix, with rows indexed from 0 to 4

	S_1	S_2	S_3	S_4
h_1	1	3	2	1
h_2	0	2	0	0

Signature matrix after considering fourth row

Computing Signature Matrices in Practice

- Consider n hash functions $h_i : \{0, ..., m-1\} \to \{0, ..., m-1\}, i = 1, ..., n$
- Let r and c index rows and columns in the characteristic matrix $M \in \{0,1\}^{m \times |C|}$
- So c also index columns, while i indexes rows in the signature matrix $SIG \in \{1, ..., m\}^{n \times |C|}$

```
for each c do
    for 0 ≤ i ≤ n do
        SIG(i, c) = ∞
    end for
end for
for each row r do
    // Iteration 5: fifth (final) row
    for each column c do
        if M(r, c) = 1 then
            for i=1 to n do
                SIG(i, c) =
                min(SIG(i, c), h_i(r))
            end for
        end if
    end for
    // End fifth (final) row
end for
```

Computing Signature Matrices: Example

Row	S_1	S_2	S_3	S_4	$x+1 \mod 5$	$3x+1 \mod 5$
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

Hash functions computed for a characteristic matrix, with rows indexed from 0 to 4

	S_1	S_2	S_3	S_4
h_1	1	3	0	1
h_2	0	2	0	0

Signature matrix after considering fifth row: final signature matrix

Computing Signature Matrices: Example

	S_1	S_2	S_3	S_4
h_1	1	3	0	1
h_2	0	2	0	0

Signature matrix after considering fifth row: final signature matrix

- *Estimates* for Jaccard similarity: $\text{SIM}(S_1, S_3) = \frac{1}{2}, \text{SIM}(S_1, S_4) = 1$
- *True* Jaccard similarities: $\text{SIM}(S_1, S_3) = \frac{1}{3}, \text{SIM}(S_1, S_4) = \frac{2}{3}$
- Estimates will be better when raising number of hash functions that is increasing number of rows of the signature matrix

Minhashing - Issues Remaining

- Minhashing is time-consuming, because iterating through all m rows of M necessary, and m is large (huge!)
- *Thought experiment:*
  - Imagine using real permutations
  - Recall: minhash is first row in permuted order with a 1
  - Consider permutations $\pi : \{1, ..., \bar{m}\} \rightarrow \{1, ..., \bar{m}\}$ for $\bar{m} < m$
  - Consider only examining the first $\bar{m}$ of the permuted rows
  - Speed up of a factor of $\frac{m}{\bar{m}}$
  - However, all first $\bar{m}$ rows may have 0 in some columns
- How to deal with that? Can we nevertheless work with only $\bar{m} < m$ rows and compute sufficiently accurate estimates for the Jaccard similarity of two columns?

SPEEDING UP MINHASHING: MOTIVATION

- Continue *thought experiment*
- Consider computing signature matrices by only examining $\bar{m} < m$ rows in the characteristic matrix, and using permutations $\pi : \{1, ..., \bar{m}\} \to \{1, ..., \bar{m}\}$ where
- the chosen $\bar{m}$ rows need not be the first $\bar{m}$ rows
- For each permutation where no 1 shows, keep ∞ as symbol in the signature matrix *SIG*
- *Situation:* Much faster to compute *SIG*, but $SIG(i, c) = \infty$ in some places (how many? is this bad?)
- Consider computing Jaccard similarities for pairs of columns

SPEEDING UP MINHASHING: MOTIVATION

Situation:

- Compute Jaccard similarities for pairs of columns, while possibly
- $SIG(i,c) = \infty$ for some (i,c)
- *Algorithm for estimating Jaccard similarity:*
  - Row by row, by iterative updates,
  - maintain count of rows a where columns agree
  - maintain count of rows d where columns disagree
  - Estimate SIM as $\frac{a}{a+d}$

Three cases:

1. *Both columns do not contain ∞ in row:* update counts as usual (either $a \to a+1$ or $d \to d+1$
2. *Only one column has ∞ in row:*
   - Let two rows be c_1, c_2, and $SIG(i,c_1) = \infty$, but $SIG(i,c_2) \neq \infty$:
   - It follows that $SIG(i,c_1) > SIG(i,c_2)$
   - So increase count of disagreeing rows by one ($d \to d+1$)
3. *Both columns have ∞ in a row:* unclear situation, skip updating counts

Speeding up Minhashing: Motivation

Summary: One determines $\frac{a}{a+d}$ as estimate for $SIM(c_1, c_2)$

- Counts rely on less rows than before. How reliable are they?
- However, since each permutation only refers to $\bar{m} < m$ rows, we can afford more permutations
- The one makes counts less reliable, while the other compensates for it
- Can we control the corresponding trade-off to our favour?
- What are the consequences in practice when using real hash functions and not permutations?

Speeding up Minhashing: Issues to Resolve

- Let T be the set of elements of the universal set that correspond to the initial $\bar{m}$ rows in the characteristic matrix.
- When executing the above algorithm on only these $\bar{m}$ rows, we determine
$$\frac{|S_1 \cap S_2 \cap T|}{|(S_1 \cup S_2) \cap T|} \tag{7}$$
as an estimate for the true Jaccard similarity $\frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$.
- If T is chosen randomly, the expected value of (7) is the Jaccard similarity $\mathrm{SIM}(S_1, S_2)$
- But: there may be some disturbing variation to this estimate

Speeding up Minhashing: Strategy

Idea in practice using hash functions

- Divide m rows into $\frac{m}{\bar{m}}$ blocks of $\bar{m}$ rows each
- For each hash function $h : \{0, ..., \bar{m} - 1\} \to \{0, ..., \bar{m} - 1\}$, compute minhash values for each block of $\bar{m}$ rows
- Yields $\frac{m}{\bar{m}}$ minhash values for a single hash function, instead of just one
- *Extreme:* If $\frac{m}{\bar{m}}$ is large enough, only one hash function may be necessary
- *Possible advantage:* By using all m rows, one balances out errors in the particular estimates on only $\bar{m}$ of the m rows:
  - The overall x of the type X rows are distributed across blocks of $\bar{m}$ rows
  - Likewise, the overall y type Y rows are distributed across the blocks

Speeding up Minhashing: Example

S_1	S_2	S_3
0	0	0
0	0	0
0	0	1
0	1	1
1	1	1
1	1	0
1	0	0
0	0	0

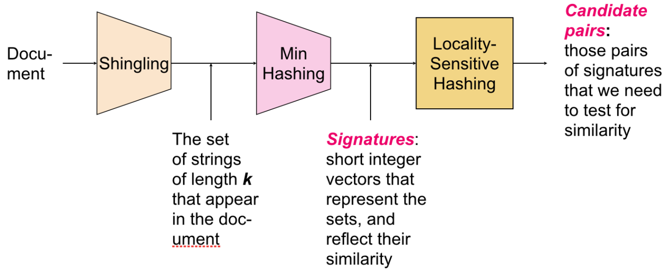
Characteristic matrix for three sets S_1, S_2, S_3. $m = 8, \bar{m} = 4$.

- *Truth:* $\text{SIM}(S_1, S_2) = \frac{1}{2}, \text{SIM}(S_1, S_3) = \frac{1}{5}, \text{SIM}(S_2, S_3) = \frac{1}{2}$
- *Estimate for first four rows:* $\text{SIM}(S_1, S_2) = 0$
- *Estimate for last four rows:* $\text{SIM}(S_1, S_2) = \frac{2}{3}$ on average across randomly picked hash functions
- *Overall estimate (expected across randomly picked hash functions):* $\text{SIM}(S_1, S_2) = \frac{1}{3}$, Ok estimate for two hash functions

Current Status: Issues Still Remaining

- Estimating similarity for each pair of sets may be infeasible even when using minhash signatures just because number of pairs is too large
- Apart from parallelism nothing can help
- *Question/Idea:* Can we determine *candidate pairs*, and only compute similarity for them, knowing similarity will be small for all others?
- *Solution: Locality Sensitive Hashing (a.k.a. Near Neighbor Search)*

Summary of Current Status



From mmds.org

- *Shingling:* turning text files into sets ☞ Done!
- *Minhashing:* computing similarity for large sets ☞ Done!
- *Locality Sensitive Hashing:* avoids $O(N^2)$ comparisons by determining candidate pairs ☞ next lecture!

Materials / Outlook

- See *Mining of Massive Datasets*, chapter 3.1–3.3
- As usual, see `http://www.mmds.org/` in general for further resources
- Next lecture: "Finding Similar Items II"
  - See *Mining of Massive Datasets* 3.4–3.6