



# Algorithms for Data Science

## Finding Similar Items

---

**Silviu Maniu**

September 17th, 2021

M2 Data Science

# Table of contents

Similar Items Problem

Shingling

Min-Hashing

Locality-Sensitive Hashing (LSH)

# Similar Items



# General Problem

Many data mining tasks can be expressed as finding **similar** sets.

- same as **finding near-neighbours in high-dimensional space**

Some applications:

- **similar pages on the Web**: duplicate detection for search engines
- **customer who purchased similar products**
- **images having similar features**

# Similarity and Distance

**Input** – set of high dimensional data points represented as vectors  $(x_1 \ x_2 \ x_3 \ \dots \ x_n)$  and a distance function  $d(p_i, p_j)$

**Problem** – find pairs of data points  $(p_i, p_j)$  that are close, i.e., in a distance threshold  $d(p_i, p_j) \leq \tau$

- comparing all pairs would take  $\mathcal{O}(N^2)$  ( $N$  number of data points)
  - too expensive
- can be done **much faster**, around  $\mathcal{O}(N)$

# Documents and Set Similarity

In this lecture, we will study how to find **similar documents** – **near-duplicate pairs**

- plagiarism, mirror pages, articles having the same source

Documents are represented as **sets** / **bags** – we will discuss *how*

Focus on **Jaccard** distance/similarity:

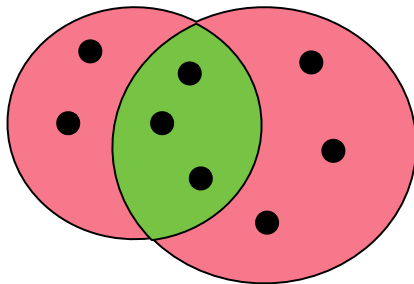
- **Jaccard similarity** of two sets  $S_1, S_2$ :

$$\text{sim}(S_1, S_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$$

- **Jaccard distance** is:

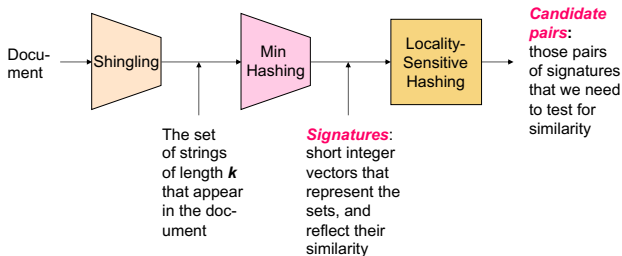
$$d(S_1, S_2) = 1 - \text{sim}(S_1, S_2)$$

# Jaccard similarity/distance



1. **similarity**  $3/8$  – *fraction of the green area*
2. **distance**  $5/8$  – *fraction of the red area*

# Steps for Finding Similar Documents



1. **shingling** – converting documents to sets
2. **min-hashing** – convert each document to a short signature
3. **locality-sensitive hashing** – reduce the number of pairs of signatures to compare



# Table of contents

Similar Items Problem

Shingling

Min-Hashing

Locality-Sensitive Hashing (LSH)

# Shingling

*Naïve way* – represent documents as the set of their words – would find many documents that are similar (common words in the language of the document)

- better way – **shingling**

**Shingling:**  $k$ -shingle = any substring of length  $k$  found in the document

- the document is then **the set of shingles appearing at least once**

*Example*

- take the document  $D$  represented by the string `abcdabd`
- the set of **2**-shingles is then  $\{\text{ab, bc, cd, da, bd}\}$

# Shingling in Practice

**Principle** –  $k$  should be picked large enough that the probability of any given shingle appearing in any given document is as low as possible

- assume a document has the **27** chars in the ASCII character set and  $k = 5$
- the number of shingles is  $27^5 = 14,348,907$  possible shingles – so  $k = 5$  works well for any document that is much smaller than the above size

# Shingling in Practice

- in practice,  $k = 5$  is good for emails,  $k = 10$  is good for large documents
- the size of the sets can be larger than the documents – **hash** the shingles to an integer having a limited number of bits – e.g., for  $k = 2$  we only need **10** bits:

$$\{\text{ab}, \text{bc}, \text{cd}, \text{da}, \text{bd}\} \rightarrow \{342, 825, 312, 54\}$$

- the **similarity/distance** is the Jaccard similarity of sets, applied on the  $k$ -shingle sets of each document

# Representing Sets of Documents as a Matrix

Conceptually, we will represent the sets of documents as a **Boolean matrix**

- **rows** are the indexes of the possible shingles
- **columns** represent the documents as

## Running example

Documents represented as sets  $D_1 = \{a, d\}$ ,  $D_2 = \{c\}$ ,  $D_3 = \{b, d, e\}$ ,  $D_4 = \{a, c, d\}$

Table:

Shingle hash	$D_1$	$D_2$	$D_3$	$D_4$
1	1	0	0	1
2	0	0	1	0
3	0	1	0	1
4	1	0	1	1
5	0	0	1	0

# Table of contents

Similar Items Problem

Shingling

Min-Hashing

Locality-Sensitive Hashing (LSH)

# Why Min-Hashing?

The encoding of sets as boolean values can still be **too costly** – cost = the number of different possible shingles, or the **size of the universal set**

We want to minimize the size of this set, and transform the set into a **signature set**

- in other words, compress the size of the **columns** (=documents) in the matrix

**Principle** – similarity of signature sets = similarity of shingle sets = similarity of documents

# Min-Hashing for Jaccard similarity

**Objective** – find a hash function  $h$  (on the shingle set of the documents), such that:

- if  $\text{sim}(D_1, D_2)$  is high, then with high probability,  $h(D_1) = h(D_2)$
- if  $\text{sim}(D_1, D_2)$  is low, then with high probability,  $h(D_1) \neq h(D_2)$

Not all similarity metrics / distances have such a hash function!

- Jaccard has one – **Min-Hashing**



# Min-Hashing for Jaccard similarity

Hash each column  $C$  of the table to a small signature  $h(C)$ :

1.  $h(C)$  is **small enough to fit in main memory**
2.  $\text{sim}(C_i, C_j)$  is **the same** as  $\text{sim}(h(C_i), h(C_j))$

# Min-Hashing

Shuffle the rows of the matrix using a **random permutation**  $\pi$

Define the hash function  $h_\pi(\mathbf{C})$  as the **first row** (in permutation order of  $\pi$ ) where we find a value of **1**

## Example

Permutation:

$\pi$	$D_1$	$D_2$	$D_3$	$D_4$
2	0	0	<b>1</b>	0
5	0	0	1	0
1	<b>1</b>	0	0	<b>1</b>
4	1	0	1	1
3	0	<b>1</b>	0	1

Min-Hash:	$D_1$	$D_2$	$D_3$	$D_4$
	1	3	2	1

# Min-Hash Property

**Property**  $\Pr[h_\pi(D_1) = h_\pi(D_2)] = \text{sim}(D_1, D_4)$ , for any random permutation  $\pi$

Proof sketch:

$D_1$	$D_4$
1	1
0	0
0	1
1	1
0	0

$$\text{sim}(D_1, D_4) = 2/3$$

- let  $s \in D$  a shingle
- equally likely that  $s \in D$  is mapped to the min element;  $\Pr[\pi(s) = \min(\pi(D))] = 1/|D|$
- let  $s$  be such that  $\pi(s) = \min(\pi(D_1 \cup D_4))$
- either  $\pi(s) = \min(\pi(D_1))$  if  $s \in D_1$ , or  $\pi(s) = \min(\pi(D_4))$  if  $s \in D_4$
- probability that **both** are true is  $\Pr[s \in D_1 \cap D_4]$
- $\Pr[h_\pi(D_1) = h_\pi(D_2)] = \frac{|D_1 \cap D_4|}{|D_1 \cup D_4|} = \text{sim}(D_1, D_4)$ .

# Min-Hash in Practice

In practice, we need **multiple hash functions**, and thus the similarity of two documents is the fraction of the hash functions in which they agree

- this works because of **the min-hash property**, the similarity of columns is the same as the **expected** similarity of their signatures

## Implementation

- permuting rows is **too costly**!
- we can use well-chosen hash functions that achieve a permutation
- the more hash functions we choose, the more exact the computation is – but **more costly**

Signature of a document:  $\mathcal{O}(K)$  (number of hash functions)

# Min-Hash in Practice

	$D_1$	$D_2$	$D_3$	$D_4$	$x + 1 \bmod 5$	$3x + 1 \bmod 5$
1	1	0	0	1	1	1
2	0	0	1	0	2	4
3	0	1	0	1	3	2
4	1	0	1	1	4	0
5	0	0	1	0	0	3

**Algorithm:** Choose  $K$  permutation functions, initialize  $\text{sig}(i, \mathbf{c}) = \infty$ , and then for each row(=shingle)  $r$ :

1. compute  $h_1(r), \dots, h_K(r)$
2. for each column(=document)  $\mathbf{c}$ : if  $\mathbf{c}$  has 1, then set  $\text{sig}(i, \mathbf{c}) = \min(h_i(r), \text{sig}(i, \mathbf{c}))$  for  $i \in 1, \dots, K$

# Table of contents

Similar Items Problem

Shingling

Min-Hashing

Locality-Sensitive Hashing (LSH)

# Objectives

We achieved smaller documents, but we still need to find a way to compare as few pairs as possible

**Idea** find a way to only compare pairs that have a similarity above a threshold  $t$

- LSH: use a function  $f(x, y)$  that tell whether the pair  $x, y$  is a **candidate pair** for comparison

# LSH for Min-Hashing

Assume we have a similarity threshold  $s$

Columns  $x$  and  $y$  of the signature matrix  $M$  are **candidate pairs** if the signature agrees on at least a fraction  $s$  of their rows

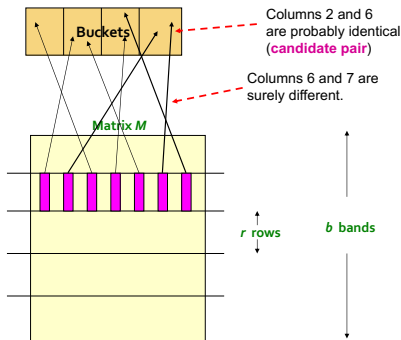
- **reminder** we assume that the min-hashed signature output the same expected similarity as the real one

## Idea behind LSH for Min-Hashing:

- hash columns of the signature matrix several times, so that only **similar columns are likely to hash to the same bucket** – candidate pairs are those that hash to the same bucket
- we can divide  $M$  into  $b$  bands of  $r$  rows each



# LSH for Min-Hashing



- for each band, hash the portion of the column into  $k$  buckets
- **candidates** are column pairs (=document pairs) hashing to the same bucket **at least once**
- have to tune  $b$  and  $r$  to catch **most** similar pairs, but fewer non-similar pairs

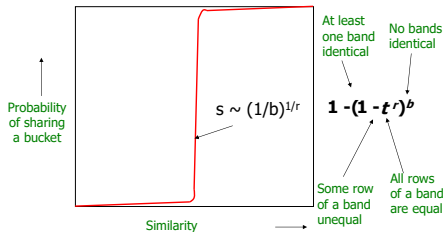
## Tuning $b$ and $r$

**Tradeoff** number of min-hashes, number of bands  $b$ , number of rows per band  $r$

How to compute this?

1. prob. that signatures agree in all rows of one band is  $s^r$
2. prob. that signatures disagree in at least one row is  $1 - s^r$
3. the prob. that signatures disagree in at least one row of each of the bands is  $(1 - s^r)^b$
4. **candidate pair** if agrees in all the rows of at least one band, prob. is  $1 - (1 - s^r)^b$

# S-curve for LSH



have to choose the threshold roughly where the probability is  $1/2$  – where the curve is steepest

**Approximate threshold  $t = (1/b)^{1/r}$**

## Example

Say  $D_1$  and  $D_2$  are 80% similar,  $b = 20$ ,  $r = 5$

- prob.  $D_1, D_2$  identical in a given band  $0.8^5 = 0.328$
- prob. are not similar in any of the bands:  $(1 - 0.328)^{20} = 0.00035$

Only 0.035% of the documents are **false negatives** (similar but they do not hash in the same bucket anywhere), 99.965% of **true positives** are found

## Example

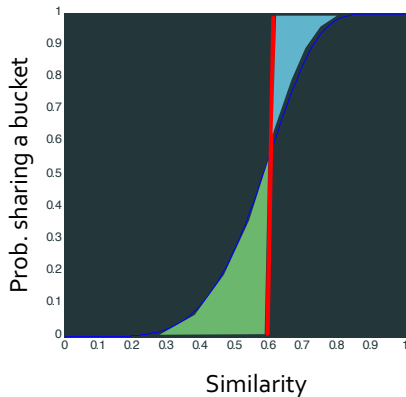
Say  $D_1$  and  $D_2$  are only 30% similar,  $b = 20$ ,  $r = 5$

- prob.  $D_1, D_2$  identical in a given band  $0.3^5 = 0.00243$
- prob. are similar in at least one of the bands:  
 $1 - (1 - 0.00243)^{20} = 0.0474$

Around 4.74% of documents having similarity of 30% end as candidate pairs – **false positives** (since they are not similar, but we still have to check them)

## Using the S-curve

Have to select  $r$  and  $b$  to get the best curve – one which minimizes the **false negatives** (blue) and **false positives** (green)



# Putting it All Together

## Outline of the steps for similar items:

1. pick a value of  $k$ , and construct  $k$ -shingles for each documents
2. pick a length  $n$  for the min-hash signatures (number of permutations)
3. choose a threshold  $t$ , along with  $b$  and  $r$  such that  $br = n$  and  $t = (1/b)^{1/r}$
4. construct candidate pairs by applying LSH
5. check each candidate pairs **in main memory** for similarity

## To Go Further

Other **similarity/distance functions** with various application (Sec. 3.5 of [Leskovec et al., 2020])

The mathematical theory behind LSH functions and applying LSH to other similarities (Sec. 3.6 and 3.7 of [Leskovec et al., 2020], [Indyk et al., 1997])



# Acknowledgments

The contents and some figures taken from Chapter 3 of  
[Leskovec et al., 2020]. <https://www.mmds.org/>

## References i



Broder, A. (1997).

**On the resemblance and containment of documents.**

In *Proceedings of the Compression and Complexity of Sequences (SEQUENCES)*.



Indyk, P., Motwani, R., Raghavan, P., and Vempala, S. (1997).

**Locality-preserving hashing in multidimensional spaces.**

In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing (STOC)*, page 618–625.



Leskovec, J., Rajaraman, A., and Ullman, J. (2020).

***Mining of Massive Datasets.***

Cambridge University Press.