Lecture 2
Finding Similar Items I

Alexander Schönhuth

UNIVERSITÄT BIELEFELD
Faculty of Technology

Bielefeld University
April 12, 2023

TODAY

Announcements

- Lecture will be *recorded*, edited and posted (as usual)
- Starting from "finding similar items" today, topics are relevant for exam
- *Reminder:* Please assign yourself to a group in the LernraumPlus, if desired; individual work possible, of course
- Groups were supposed to be up to 2-3 people; individual work possible, of course

Today: Overview

Contents today

- Useful things II (not relevant for exam)
- Similarity of sets: purpose, basic idea
- Similarity of documents: turning documents into sets ☞ shingles
- Computing the similarity of sets ☞ minhashing

Useful Things to Know II

Useful Things to Know

- The TF.IDF measure of word importance ☞ done!
- Hash functions ☞ done!
- Secondary storage (disk) and running time of algorithms
- The natural logarithm
- Power laws

SECONDARY STORAGE

- Important to keep in mind when dealing with big data: accessing data from disks (hard drives) costs time (and energy).
- Disks are organized into blocks; e.g. blocks of 64K bytes.
- Takes approx. 10 milliseconds to *access* and read a disk block.
- About 10^5 times slower than accessing data in main memory.
- And taking a block to main memory costs more time than executing the computations on the data when being in main memory.

Secondary Storage

- One can alleviate problem by putting related data on a single *cylinder*; accessing all blocks on a cylinder costs considerably less time per block
- Establishes limit of 100MB per second to transfer blocks to main memory
- If data is in the hundreds of gigabytes, let alone terabytes, this is an issue
- *Integrate this knowledge into runtime considerations when dealing with big data!*

The Natural Logarithm I

- Euler constant:

$$e = \lim_{x \to \infty} (1 + \frac{1}{x})^x \approx 2.71828 \tag{1}$$

- Consider computing $(1+a)^b$ where a is small:

$$(1+a)^b = (1+a)^{(1/a)(ab)} \stackrel{a=1/x}{=} (1 + \frac{1}{x})^{x(ab)} = ((1+\frac{1}{x})^x)^{ab} \stackrel{x \text{ large}}{\approx} e^{ab}$$

- Consider computing $(1-a)^b$ where a is small:

$$(1-a)^b = (1-a)^{(1/a)(ab)} \stackrel{-a=1/x}{=} ((1+\frac{1}{x})^x)^{-ab} \stackrel{x \text{ large}}{\approx} e^{-ab}$$

Euler Constant: Taylor Expansion of e^x

- The Taylor expansion of e^x is

$$e^x = \sum_{i=0}^{\infty} \frac{x^i}{i!} = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \dots \tag{2}$$

- Convergence slow on large x, so not helpful.
- Convergence fast on small (positive and negative) x.
- *Example:* $x = 1/2$

$$e^{1/2} = 1 + \frac{1}{2} + \frac{1}{8} + \frac{1}{48} + \frac{1}{384} + \dots \approx 1.64844$$

- *Example:* $x = -1$

$$e^{-1} = 1 - 1 + \frac{1}{2} - \frac{1}{6} + \frac{1}{24} - \frac{1}{120} + \frac{1}{720} - \frac{1}{5040} \dots \approx 0.36786$$

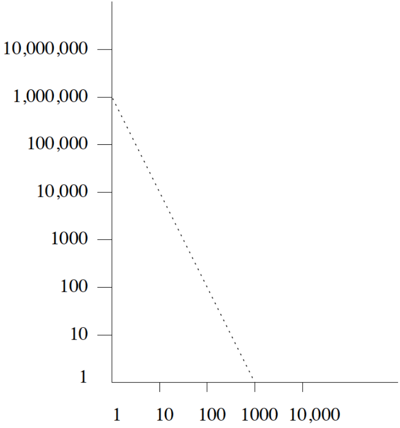
Power Laws

- Consider two variables y and x and their functional relationship.
- General form of a power law is

$$\log y = b + a \log x \tag{3}$$

so a linear relationship between the logarithms of x and y.

Power Law: Example



$$\log_{10} y = 6 - 2\log_{10} x$$

Power Laws

- Power law:

$$\log y = b + a \log x \tag{4}$$

- Transforming yields:

$$y = e^b \cdot e^{a \log x} = e^b \cdot e^{\log x^a} = e^b \cdot x^a \tag{5}$$

so power law expresses polynomial relationship $y = cx^a$

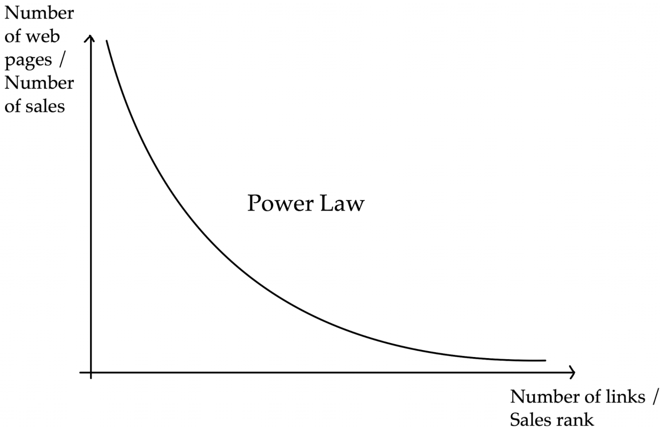
- Example slide before (logarithm base 10):

$$y = 10^6 \cdot x^{-2} \tag{6}$$

Real World Scenarios

- *Node degrees in web graph*
  - Nodes are web pages
  - Nodes are linked when there are links between pages
  - Order pages by numbers of links: number of pages as a function of the order number is power law
- *Sales of products: y* is the number of sales of the x-th most popular item (books at amazon.com, say)
- *Sizes of web sites: y* is number of pages at the x-th largest web site

Power Law: Example II



Power law for links in web pages / sales of books

Real World Scenarios

- *Zipf's Law:* Order words in document by frequency, and let y be the number of times the x-th word appears in the document.
    - Zipf found the relationship to approximately reflect $y = cx^{-1/2}$.
    - Other relationships follow that law, too. For example, y is population of x-th most populous (American) state.
- Summary: *The Matthew Effect* = "The rich get ever richer"

Finding Similar Items: Introduction

Finding Similar Items

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

Applications

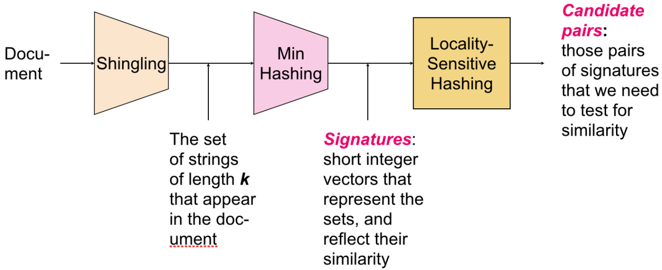
- Detecting plagiarism in a set of documents
- Identifying near-identical mirror pages during web searches
- Identifying documents from the same author
- *Collaborative Filtering*
  - Online Purchases (Amazon: suggestions based on ′similar′ customers)
  - Movie Ratings (Netflix: suggestions based on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

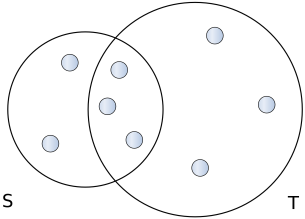
Jaccard Similarity

Definition [Jaccard Similarity]

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

$$\mathrm{SIM}(S,T) = \frac{|S \cap T|}{|S \cup T|} \tag{7}$$

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 for $k = 2$:

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

- 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 $\mathrm{SIM}(D_1, D_2) = \frac{|\{bc,cd\}|}{|\{ab,bc,cd,db\}|} = 2/4 = 1/2$

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

UNIVERSITÄT BIELEFELD

Sets as Bitvectors

- Bitvectors:
  - A bitvector is an array containing zeroes and ones
  - E.g. $[1, 0, 0, 1, 1]$ is a bitvector of length 5
  - Formally: bitvectors of length N are elements of $\{0, 1\}^N$
- *Sets as bitvectors:*
  - Length of bitvectors is size of universal set
  - Entries zero if *element not in set*, one if *element in set*
  - *Example:* universal set $= \{a, b, c, d, e\}$; set $A = \{b, c, e\}$

$$A = [\underset{a}{0}, \underset{b}{1}, \underset{c}{1}, \underset{d}{0}, \underset{e}{1}]$$

  - When hashing shingles to buckets, length of bitvector = number of buckets
- 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

$$M(r,c) = \begin{cases} 0 & \text{if } r \notin c \\ 1 & \text{if } r \in c \end{cases} \tag{8}$$

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

$$\pi : \{1, ..., m\} \to \{1, ..., m\} \tag{9}$$

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

Consider

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

Definition [Minhash]
The *minhash* function h_π on S is defined by

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

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, ...
- $\pi : 1 \rightarrow 3, 2 \rightarrow 1, 3 \rightarrow 5, 4 \rightarrow 4, 5 \rightarrow 2$ 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 $\mathrm{SIM}(S_1, S_2)$.

Minhash and Jaccard Similarity - Proof

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

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

$$\mathrm{SIM}(S_1, S_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|} = \frac{x}{x+y} \tag{10}$$

Minhash and Jaccard Similarity - Proof

Proof. (cont.)

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

$$\frac{x}{x+y} \tag{11}$$

  - 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 (11), which by (10) concludes the proof.

$\square$

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

Example / Illustration