High-level structure in texts as sets of words - III

Anton Alekseev,
Steklov Mathematical Institute in St Petersburg

NRU ITMO, St Petersburg, 2019

anton.m.alexeyev+itmo@gmail.com

Plan

1. ~~Clustering~~
2. Finding similar items
   a. Task and motivation
   b. Document as a set of shingles
   c. MinHash: compressed document representation
   d. A look at LSH
3. Topic modeling (~~in a fast pace~~)
   a. Task and motivation
   b. Matrix factorization as a topic model
   c. Probabilistic topic modeling
      i. pLSA
      ii. LDA
      iii. ARTM
   d. Topic modeling quality evaluation

Similar objects detection (duplicate detection)

Task examples

1. Webpages with almost the same content on a **search result page**
   (e.g. mirrors, same articles published on different websites);
   why show user essentially the same objects?

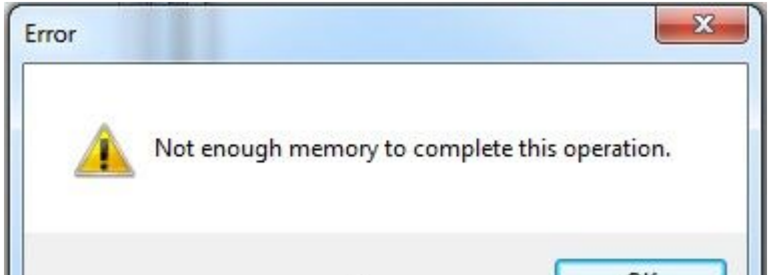
2. **Reviews on objects** (goods, organizations, places,...), collected from all over the Internet. Reasons why there are many duplicate reviews: spam, users’ revenge, platforms stealing reviews from each other, etc.

That is, we need methods to detect ‘**copy and paste**’ case, where extra symbols, paragraphs or words may be inserted. This time we don’t care about the *meaning* of the text.

Similar texts in a big collection problem

Yeah, Big Data, we’ll consider the case when

- **the document collection is large, texts may be long** — RAM will never be enough;
- fragments of documents can be just parts of other documents, **the match is inexact**;
- **we can’t build** a complete **matrix of distances between all pairs of texts** due to the dataset size.



Not enough memory to complete this operation.

Approach

Suggest

1) a way to represent **documents as sets**
   (we have suitable similarity measures for sets)

2) a way to build **signatures** (representations of lower dimensionality)
   of these sets so that **the similar sets have similar signatures**

3) a way to hash signatures so that similar signatures have **‘hash function values close to each other’**

Plan

1. ~~Clustering~~
2. ~~Finding similar items~~
    a. ~~Task and motivation~~
    b. Document as a set of shingles
    c. MinHash: compressed document representation
    d. A look at LSH
3. Topic modeling (~~in a fast pace~~)
    a. Task and motivation
    b. Matrix factorization as a topic model
    c. Probabilistic topic modeling
        i. pLSA
        ii. LDA
        iii. ARTM
    d. Topic modeling quality evaluation

Shingles algorithm

***k*-shingle** — a sequence of k consecutive tokens in the test (character of word n-gram). Let’s use all document’s shingles as its set representation.

Then we can take **Jaccard distance** as a measure of similarity

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}.$$

D = “azart azara” **Sh(D) = { az, za, ar, rt, t_, _a, ra }**

Sometimes multisets are used:
Sh’(D) = { az **x 2**, za **x 2**, ar **x 2**, rt, t_, _a, ra }

From now on, we will be talking about shingles = character n-grams

U. Manber, “Finding similar files in a large file system,” Proc. USENIX Conference, pp. 1–10, 1994.

How to choose k?

Very important but the answer is (as always) — ‘it depends on the task’

- With character shingles of k = 1 all texts
  will have exactly the same signatures :)

- As we are looking for inexact duplicates (where only long non-breaking substrings of the text can be switched with each other), which are not just documents with similar lexics or topics, one should take longer shingles

  for short documents (e.g., emails) — **k = 5**
  for long documents (articles) — **k = 9..10**

Shingles are too many

Suppose there are 27 possible characters in out texts, then
the number of all possible 9-shingles is 27^9
Evident enough, we'll never see most of them in the texts

Trick: let's hash every shingle => map into set **0..2^{32}-1 (int, 4 bytes)**

Every document will be represented as a set of hashes:

D = "azart azara"
Sh(D) = { az, za, ar, rt, t_, _a, ra }
hSh(D) = { 2, 8, 12, 4985, 11, 9800, 0 }

Provided that we have a **reasonable** hash function, collisions (two different objects having the same hash value) don't have much impact

So “signature = a set of integers”...

Suppose we have N = 1 million documents, now we want to find all similar ones, using the Jaccard distance

N (N - 1) / 2 = 5 • 10^{11}
One day is 3600 • 24 = 86400 seconds
Suppose we can do, 1’000’000 set comparisons per second

> 5,5 days

We have to do something about it!

Plan

1. ~~Clustering~~
2. ~~Finding similar items~~
    a. ~~Task and motivation~~
    b. ~~Document as a set of shingles~~
    c. MinHash: compressed document representation
    d. A look at LSH
3. Topic modeling (~~in a fast pace~~)
    a. Task and motivation
    b. Matrix factorization as a topic model
    c. Probabilistic topic modeling
        i. pLSA
        ii. LDA
        iii. ARTM
    d. Topic modeling quality evaluation

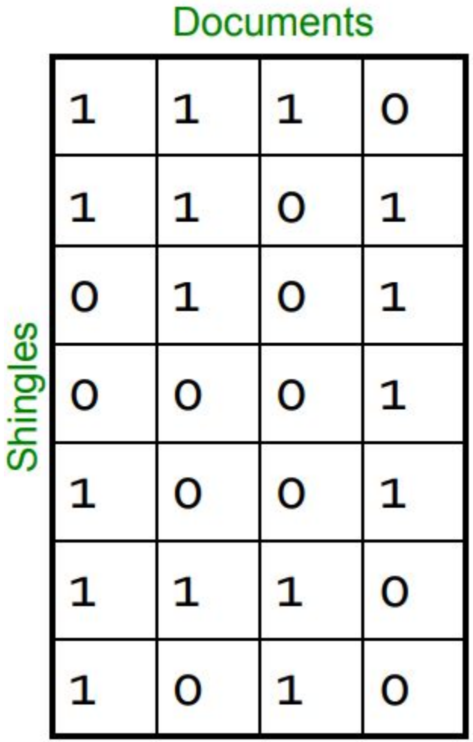
MinHashing: bit vectors

Imagine that we have a sparse ‘shingle-document’ matrix; a bit per every shingle in every column

column1 **AND** column2 = intersection
column1 **OR** column2 = union

We want to hash COLUMNS so that
with high probability

sim(c1, c2) is high ⇔ h(c1) = h(c2)
sim(c1, c2) is low ⇔ h(c1) <> h(c2)



Documents

Shingles

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

http://www.mmds.org/mmds/v2.1/ch03-lsh.pdf

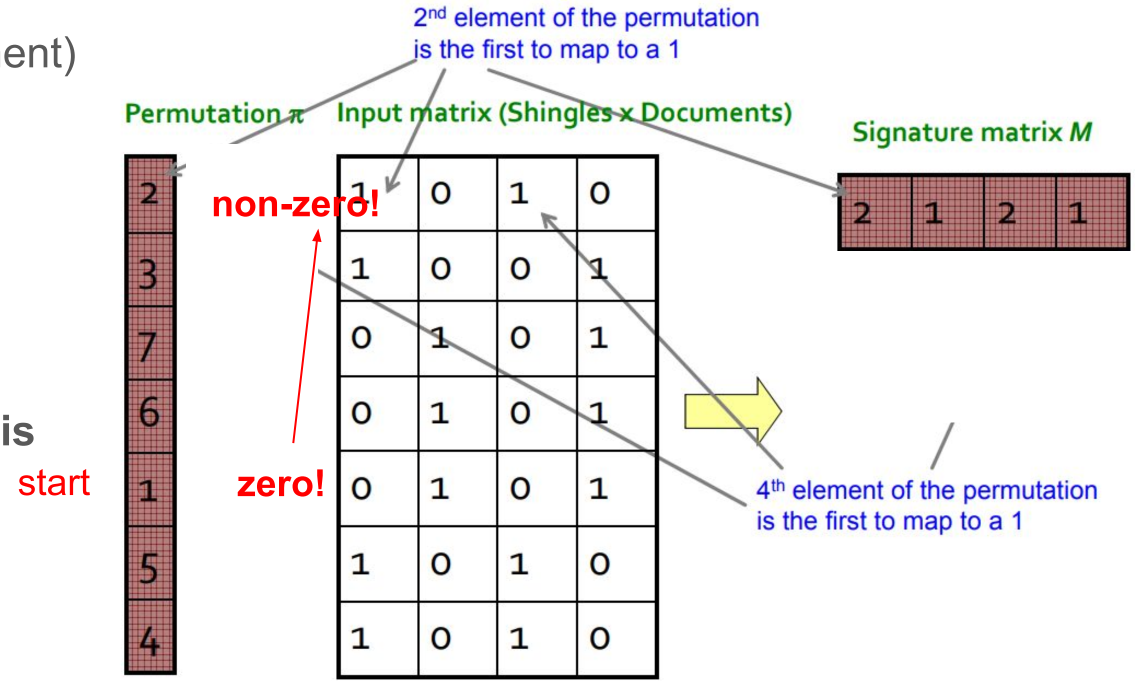
A.Z. Broder, M. Charikar, A.M. Frieze, and M. Mitzenmacher, “Min-wise independent permutations,” ACM Symposium on Theory of Computing, pp. 327–336, 1998.

What is MinHashing?

For every column (that is, a document) we write down **which index hits non-zero first** (this **is** minhash)

Then we do so for several permutations of the rows

The fraction of matching values is used as documents similarity


2nd element of the permutation is the first to map to a 1
Permutation π
Input matrix (Shingles x Documents)
Signature matrix M
non-zero!
start
zero!
4th element of the permutation is the first to map to a 1

Permutation π	Input matrix (Shingles x Documents)			
2	1	0	1	0
3	1	0	0	1
7	0	1	0	1
6	0	1	0	1
1	0	1	0	1
5	1	0	1	0
4	1	0	1	0

Signature matrix M			
2	1	2	1

http://www.mmds.org/mmds/v2.1/ch03-lsh.pdf

MinHashing: why this approach works

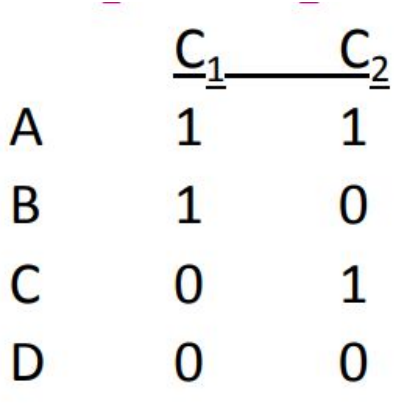
Statement: the probability of minhashes matching for a random permutation of two sets’ elements is equal to Jaccard distance between those sets

1. Jaccard coefficient — **A / (A + B + C)**

2. Let’s consider a random permutation of two sets

   We go down one column, then the probability of hitting the situation “1-1” (A) **before** “1-0” (B) or “0-1” (C) equals **A / (A+B+C)**

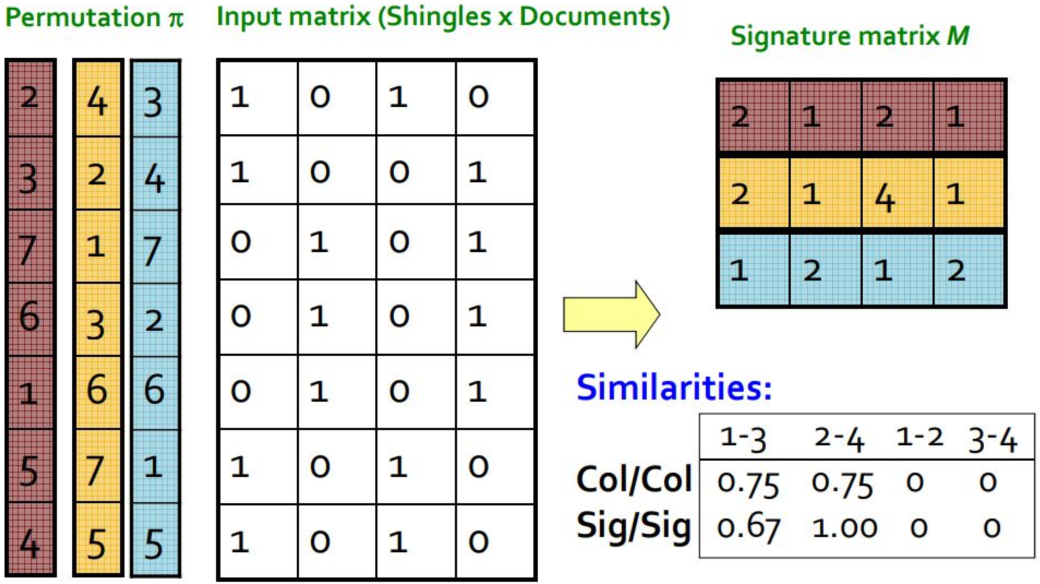
   And in case “1-0” (“0-1”) we know for sure that the minhash value of the second set is not equal to the one of the first set



	C_1	C_2
A	1	1
B	1	0
C	0	1
D	0	0

MinHashing: how it is actually done

Problem 1: that probability estimate has large variance, we need a more precise one ...so we can take a bunch of permutations (the greater the number, the better) and compute the fraction of matches



Permutation π

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

Input matrix (Shingles x Documents)

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

Signature matrix *M*

2	1	2	1
2	1	4	1
1	2	1	2

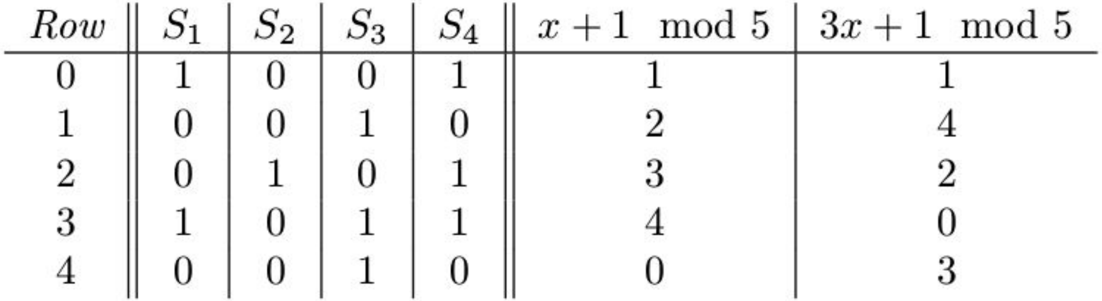
Similarities:

	1-3	2-4	1-2	3-4
Col/Col	0.75	0.75	0	0
Sig/Sig	0.67	1.00	0	0

MinHashing: how it is actually done

Problem 2: permutations take a while to generate!

...to deal with that one may take hash functions, that **generate** permutations

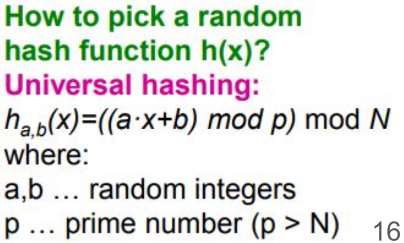


Row	S_1	S_2	S_3	S_4	$x+1 \mod 5$	$3x+1 \mod 5$
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

...then traverse all rows and

1) compute hash-permutations,

2) for each column, having met 1, update minimal values of indices for the signature



How to pick a random hash function h(x)?

Universal hashing:

$h_{a,b}(x)=((a \cdot x+b) \bmod p) \bmod N$

where:

a,b … random integers

p … prime number (p > N)

Interim results

1) documents as shingles sets
2) representing sets as signatures of small size allowing to estimate their similarity (with some probability)

One can try to invent something hacky for searching the best-matching ones (in terms of matching vector values) from scratch, but the signatures are not that short, so this may be hard

However, there is a well-known and an effective method for that!

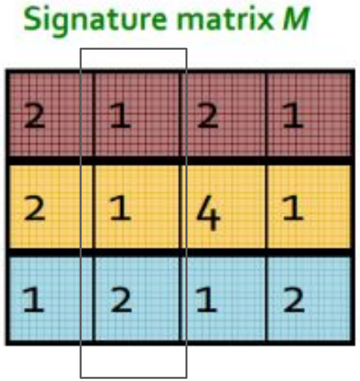
Plan

1. ~~Clustering~~
2. ~~Finding similar items~~
    a. ~~Task and motivation~~
    b. ~~Document as a set of shingles~~
    c. ~~MinHash: compressed document representation~~
    d. A look at LSH
3. Topic modeling (~~in a fast pace~~)
    a. Task and motivation
    b. Matrix factorization as a topic model
    c. Probabilistic topic modeling
        i. pLSA
        ii. LDA
        iii. ARTM
    d. Topic modeling quality evaluation

LSH: Locality-Sensitive Hashing

We want to find all pairs of sets, Jaccard distance between which is no larger than **s**

Core idea
hash all columns of a signature matrix into many buckets; documents falling into one buckets are candidates for checking whether the Jaccard distance is small between all of them



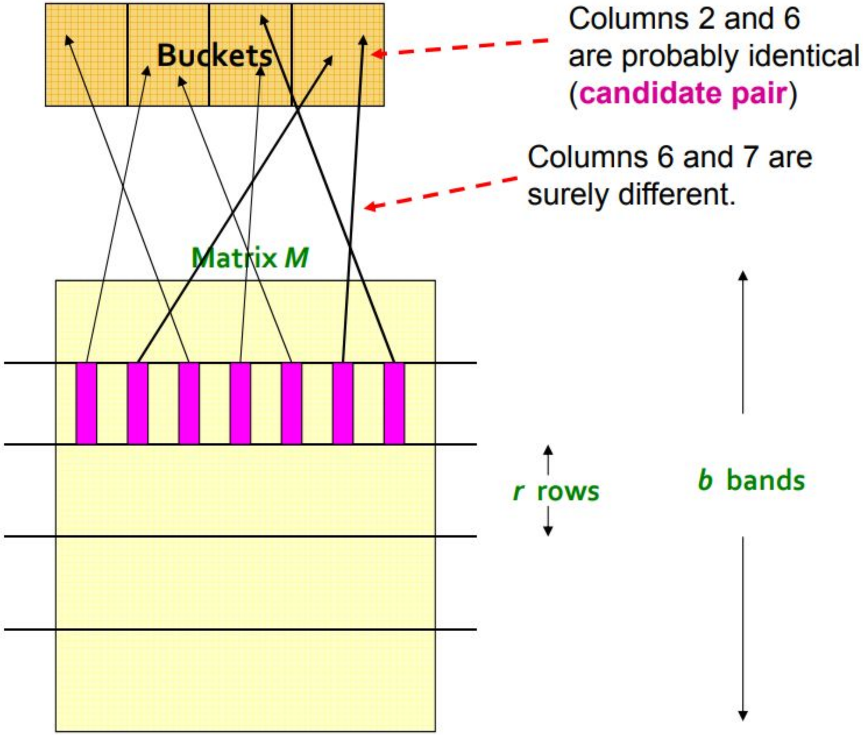
2	1	2	1
2	1	4	1
1	2	1	2

LSH: splitting signatures into *b* parts

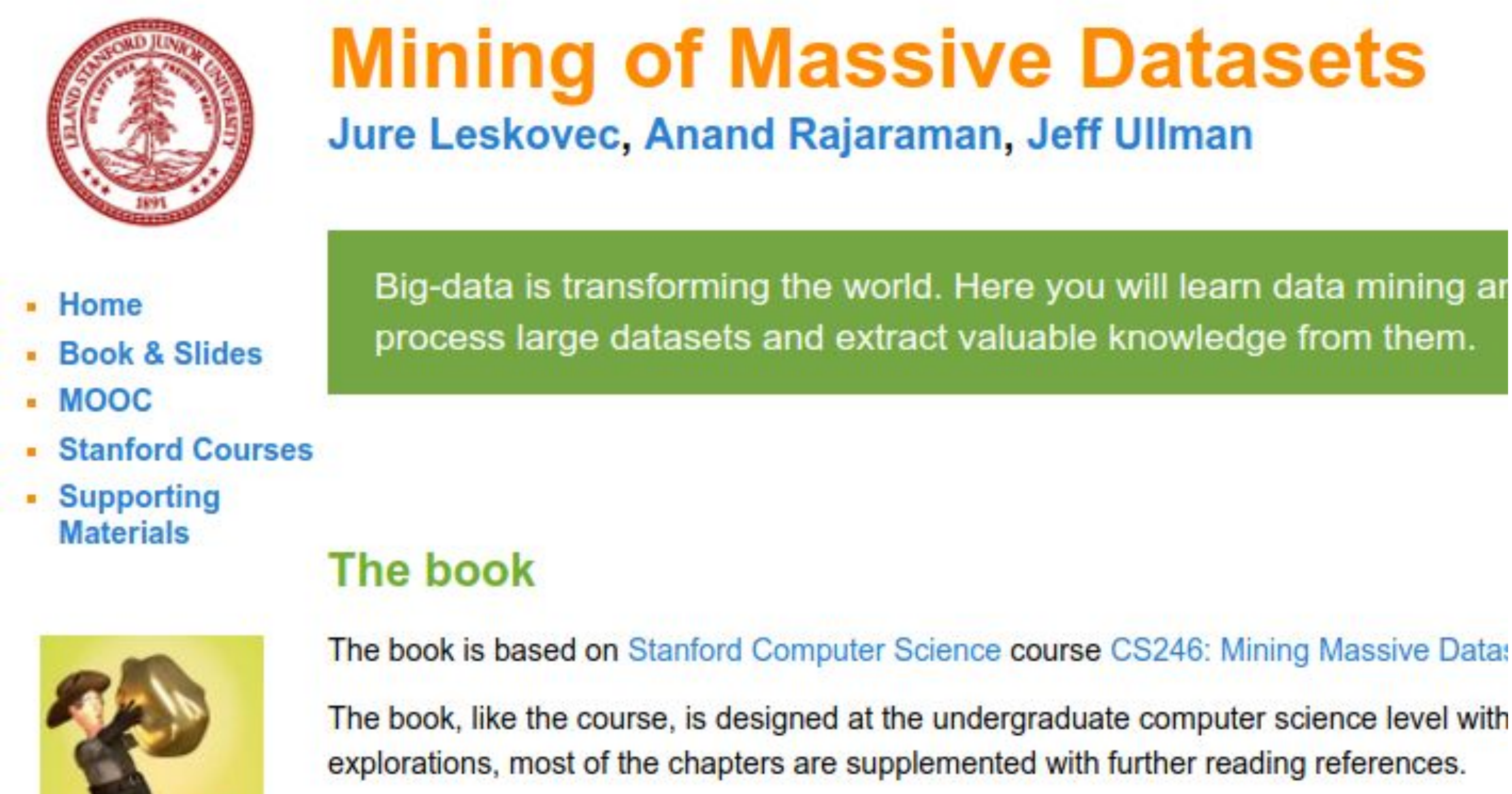
...then hashing each part, throwing into k buckets (the larger k, the better)

Signatures, hashes of which fell into the same bucket more than once, are good **duplicates candidates**

b (r) -- tuneable paremeters


Buckets
Columns 2 and 6 are probably identical (candidate pair)
Columns 6 and 7 are surely different.
Matrix M
r rows
b bands

For more on LSH please see (free btw)



Mining of Massive Datasets

Jure Leskovec, Anand Rajaraman, Jeff Ullman

- Home
- Book & Slides
- MOOC
- Stanford Courses
- Supporting Materials

Big-data is transforming the world. Here you will learn data mining ar process large datasets and extract valuable knowledge from them.

The book

The book is based on Stanford Computer Science course CS246: Mining Massive Datas

The book, like the course, is designed at the undergraduate computer science level with explorations, most of the chapters are supplemented with further reading references.

Also see

SimHash
Gurmeet Singh Manku, Arvind Jain, and Anish Das Sarma. 2007. Detecting near-duplicates for web crawling. In Proceedings of the 16th international conference on World Wide Web (WWW '07). ACM, New York, NY, USA, 141-150.

(algorithm is described well here: M. Charikar. Similarity estimation techniques from rounding algorithms. In Proc. 34th Annual Symposium on Theory of Computing (STOC2002), pages 380–388, 2002.)

Rumours say that some time ago Google used Simhash for web pages, and MinHash+LSH — for Google News

Tools & data

instruments

They created Scrapy

1. https://github.com/**scrapinghub**/python-simhash
2. Looks promising: https://github.com/ekzhu/datasketch
3. Just google it: custom implementations of MinHash, LSH, etc.

datasets

1. Wikipedia revisions
   (see New Issues in Near-duplicate Detection Martin Potthast and Benno Stein)
2. Webdata+xml + linuxdocs
   (see A Scalable System for Identifying Co-Derivative Documents Yaniv Bernstein Justin Zobel)
3. 8B documents
   (but you have to be Google:))
4. Just google, there are more

Used/recommended materials

1. Mining Massive Datasets, Chapter 3.
2. References on slides above
3. Wikipedia

High-level structure in texts as sets of words - III

Anton Alekseev,
Steklov Mathematical Institute in St Petersburg

NRU ITMO, St Petersburg, 2019
anton.m.alexeyev+itmo@gmail.com

Thanks for help in slides preparation go to Denis Kiryanov and Andrey Filchenkov