CAMPUS
DE EXCELENCIA
INTERNACIONAL

POLITÉCNICA
"Ingeniamos el futuro"

Text Mining 1
Information Retrieval

Madrid Summer School on
Advanced Statistics and Data Mining

Florian Leitner
Data Catalytics, S.L.
leitner@datacatytics.com

Converting tokens to numbers (part 1)

- **Tokenization** is the process of splitting text into words, punctuation, and symbols (aka. tokens).
- **Indexing** can refer to linking tokens to document and counting the frequencies of each token...
  - within a document: (token or) **term frequency TF**
  - number of documents with that token: **document frequency DF**
  - overall count in your document collection: **corpus frequency CF**
- The remaining question then is: how to make computations with these numbers?

The inverted index

factors, normalization (len[text]), probabilities, and n-grams

Text 1: He that not wills to the end neither wills to the means.

Text 2: If the mountain will not go to Moses, then Moses must go to the mountain.

tokens	Text 1	Text 2
end	1	0
go	0	2
he	1	0
if	0	1
means	1	0
Moses	0	2
mountain	0	2
must	0	1
not	1	1
that	1	0
the	2	2
then	0	1
to	2	2
will	2	1

unigrams	T1	T2	p(T1)	p(T2)
end	1	**0**	0.09	0.00
go	**0**	2	0.00	0.13
he	1	**0**	0.09	0.00
if	**0**	1	0.00	0.07
means	1	**0**	0.09	0.00
Moses	**0**	2	0.00	0.13
mountain	**0**	2	0.00	0.13
must	**0**	1	0.00	0.07
not	1	1	0.09	0.07
that	1	**0**	0.09	0.00
the	2	2	0.18	0.13
then	**0**	1	0.00	0.07
to	2	2	0.18	0.13
will	2	1	0.18	0.07
SUM	11	15	1.00	1.00

bigrams	Text 1	Text 2
end, neither	1	**0**
go, to	**0**	2
he, that	1	**0**
if, the	**0**	1
Moses, must	**0**	1
Moses, then	**0**	1
mountain, will	**0**	1
must, go	**0**	1
not, go	**0**	1
not, will	1	**0**
that, not	1	**0**
the, means	1	**0**
the, mountain	**0**	2
then, Moses	**0**	1
to, Moses	**0**	1
to, the	2	1
will, not	**0**	1
will, to	2	**0**

Document similarity

- Similarity measures
  - **Cosine similarity (of document/text vectors)**
  - **Correlation coefficients**
- Document vector normalization
  - **TF-IDF**
- Dimensionality reduction/clustering
  - **Locality sensitive hashing**
  - **Latent semantic indexing**
  - **Latent Dirichlet allocation**

Document vectors

Collections of vectorized texts: **Inverted index**

Text 1: He that not wills to the end neither wills to the means.

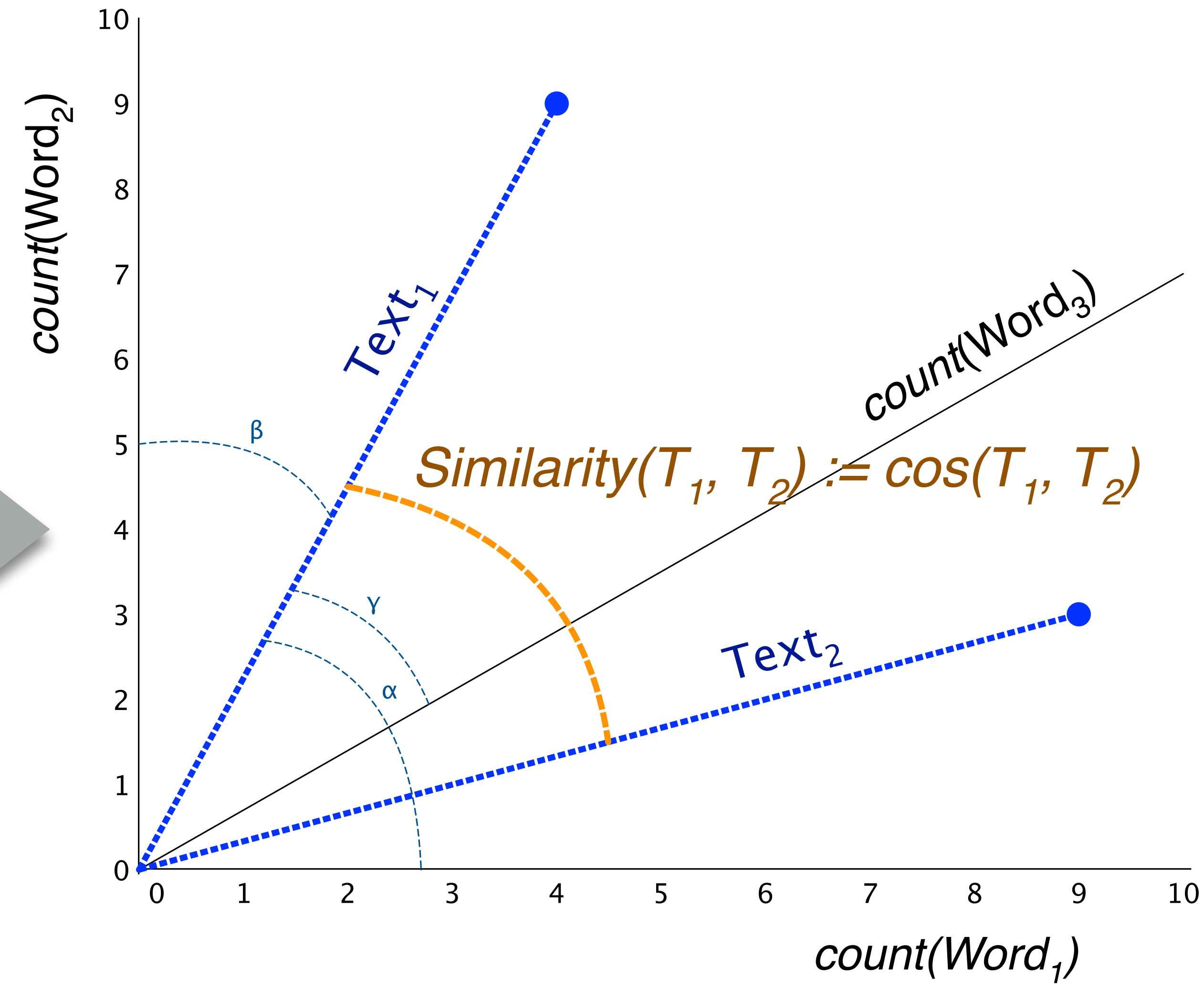
Text 2: If the mountain will not go to Moses, then Moses must go to the mountain.

each token/word is a dimension!

tokens	Text 1	Text 2
end	1	0
go	0	2
he	1	0
if	0	1
means	1	0
Moses	0	2
mountain	0	2
must	0	1
not	1	1
that	1	0
the	2	2
then	0	1
to	2	2
will	2	1

↑ word vector ↑ (Text 1) ↑ word vector ↑ (Text 2)

Comparing word vectors: **Cosine similarity**



Terrier Sphinx Xapian INDRI Lucene

Cosine similarity

$$sim(\vec{x},\vec{y}) = cos(\vec{x},\vec{y}) = \frac{\vec{x}\cdot\vec{y}}{|\vec{x}||\vec{y}|} = \frac{\sum x_i y_i}{\sqrt{\sum x_i^2}\sqrt{\sum y_i^2}}$$

can be dropped if using unit vectors ("length-normalized" a.k.a. "cosine normalization") only dot-product is now left: extremely efficient ways to compute on modern CPUs

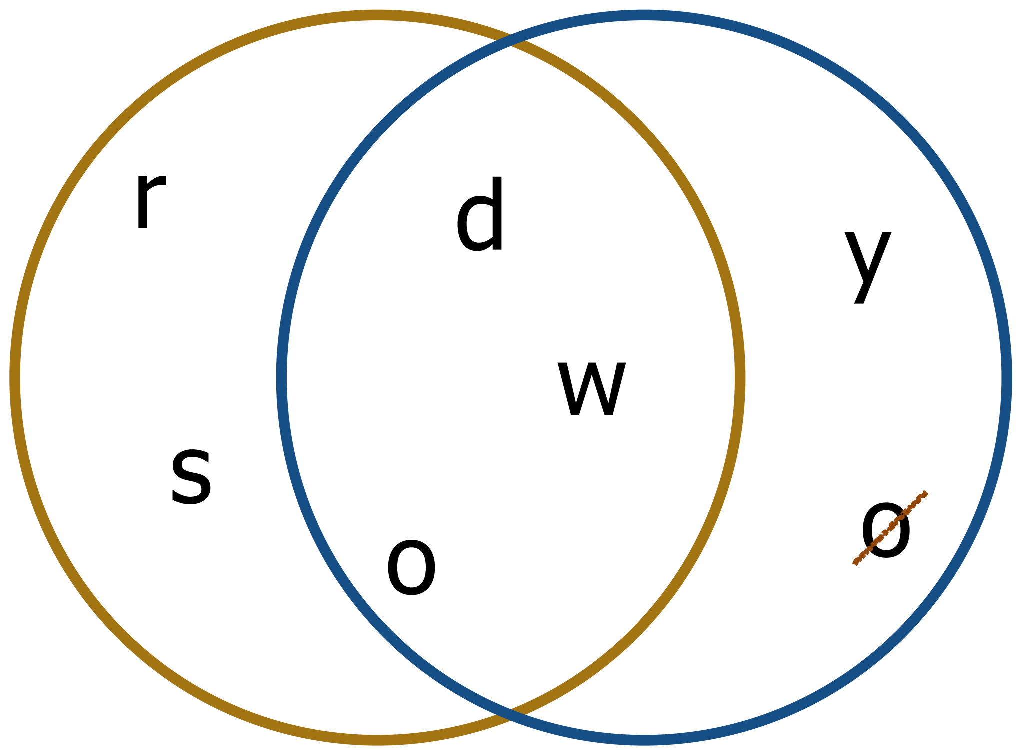
- Define a similarity score between two document vectors (or the query vector in Information Retrieval)
- **Euclidian** vector **distance** is **length dependent**
- The **cosine** [angle] between two vectors **is not**

tokens	Text 1	Text 2
end	1	0
go	0	2
he	1	0
if	0	1
means	1	0
Moses	0	2
mountain	0	2
must	0	1
not	***1***	***1***
that	1	0
the	***2***	***2***
then	0	1
to	***2***	***2***
will	***2***	***1***

$$sim(T_1,T_2) = \frac{1*1+2*2+2*2+2*1}{\sqrt{(17)} * \sqrt{(25)}} = 0.5336$$

Jaccard’s [set] similarity

“words” vs “wo~~o~~dy”



ABC

vs.

ABCABCABC

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{3}{\cancel{7}\;6} = \cancel{0.43}\quad 0.5$$

Alternative similarity **coefficients**

- **Spearman's rank correlation coefficient** ρ (r[ho])
  - Ranking is done by term frequency (**TF**; count)
  - Critique: sensitive to ranking differences that are likely to occur with high-frequency words (e.g., "the", "a", …) ➡ use the **log** of the term count, rounded to two significant digits
    - NB that this is not relevant when only short documents (e.g. titles) with low TF counts are compared
- **Pearson's chi-square test** χ^2
  - Directly on the TFs (counts) - intuition:
    Are the TFs "random samples" from the same base distribution?
  - Usually, χ^2 should be preferred over ρ (Kilgarriff & Rose, 1998)
    - NB that both measures have no inherent normalization of document size
      - preprocessing might be necessary!

Term Frequency times Inverse Document Frequency (TF-IDF)

- **Motivation and background**
- The problem
  - **Frequent terms** contribute most to a document vector's direction, but **not all** terms are **relevant** ("the", "a", …).
- The goal
  - **Separate** important terms from frequent, but **irrelevant terms** in the collection.
- The idea
  - Frequent **terms** appearing **in all documents** tend to be less important **versus** frequent terms in just a **few documents**. → Zipf's Law!
    - Also **dampens** the effect of **topic-specific noun phrases** or an **author's bias** for a specific set of adjectives

Term Frequency times Inverse Document Frequency (TF-IDF)

- $\textbf{tf.idf}(w) := \text{tf}(w) \times \text{idf}(w)$
  - tf: (document-specific) term frequency
  - idf: inverse (global) document frequency
- $\textbf{tf}_{\textbf{natural}}(w) := \text{count}(w)$
  - $\text{tf}_{\text{natural}}$: n. of times term w occurs in a document
- $\textbf{tf}_{\textbf{log}}(w) := \log(\text{count}(w) + 1)$
  - tf_{log}: the TF is smoothed by taking its log
- $\textbf{idf}_{\textbf{natural}}(w) := N / \sum^N \{w_i > 0\}$
  - $\text{idf}_{\text{natural}}$: n. documents divided by n. documents in which term w occurs
- $\textbf{idf}_{\textbf{log}}(w) := \log(N / \sum^N \{w_i > 0\})$
  - idf_{log}: the IDF is smoothed by taking its log
  - where N is the **number of documents**, w_i the **count of word** w in document i, and $\{w_i>0\}$ is 1 if document i has word w or 0 otherwise

TF-IDF in information retrieval

- Document vectors = tf_{log}

→ i.e. the doc. vectors do not use any IDF weighting (because its more efficient: the QV uses IDF, and that gets multiplied with the DV values)

- Query vector = $tf_{log}\ idf_{log}$
  - Terms are counted on each individual document & the query
- Cosine vector length normalization for TF-IDF scores:
  - Document W normalization

$$\sqrt{\sum_{w \in W} tf_{log}(w)^2}$$

  - Query Q normalization

$$\sqrt{\sum_{q \in Q} (tf_{log}(q) \times idf_{log}(q))^2}$$

- IDF is calculated over the indexed collection of all documents

TF-IDF query score: An example

	Collection		Query Q				Document D			Similarity
Term	df	idf_{log}	tf	tf_{log}	tf.idf	norm	tf	tf_{log}	tf.1	cos(Q,D)
best	3.5E+05	1.46	1	0.30	0.44	0.21	**0**	0.00	0.00	0.00
text	2.4E+03	3.62	1	0.30	1.09	0.53	10	1.04	0.06	0.03
mining	2.8E+02	4.55	1	0.30	1.37	0.67	8	0.95	0.06	0.04
tutorial	5.5E+03	3.26	1	0.30	0.98	0.48	3	0.60	0.04	0.02
data	9.2E+05	1.04	**0**	0.00	0.00	0.00	10	1.04	0.06	0.00
…	…	…	…	…	0.00	0.00	…	16.00	…	0.00
Sums	1.0E+07		4		2.05		~355	16.11		**0.09**

× ×

√ of ∑ of ² ÷ √ of ∑ of ² ÷

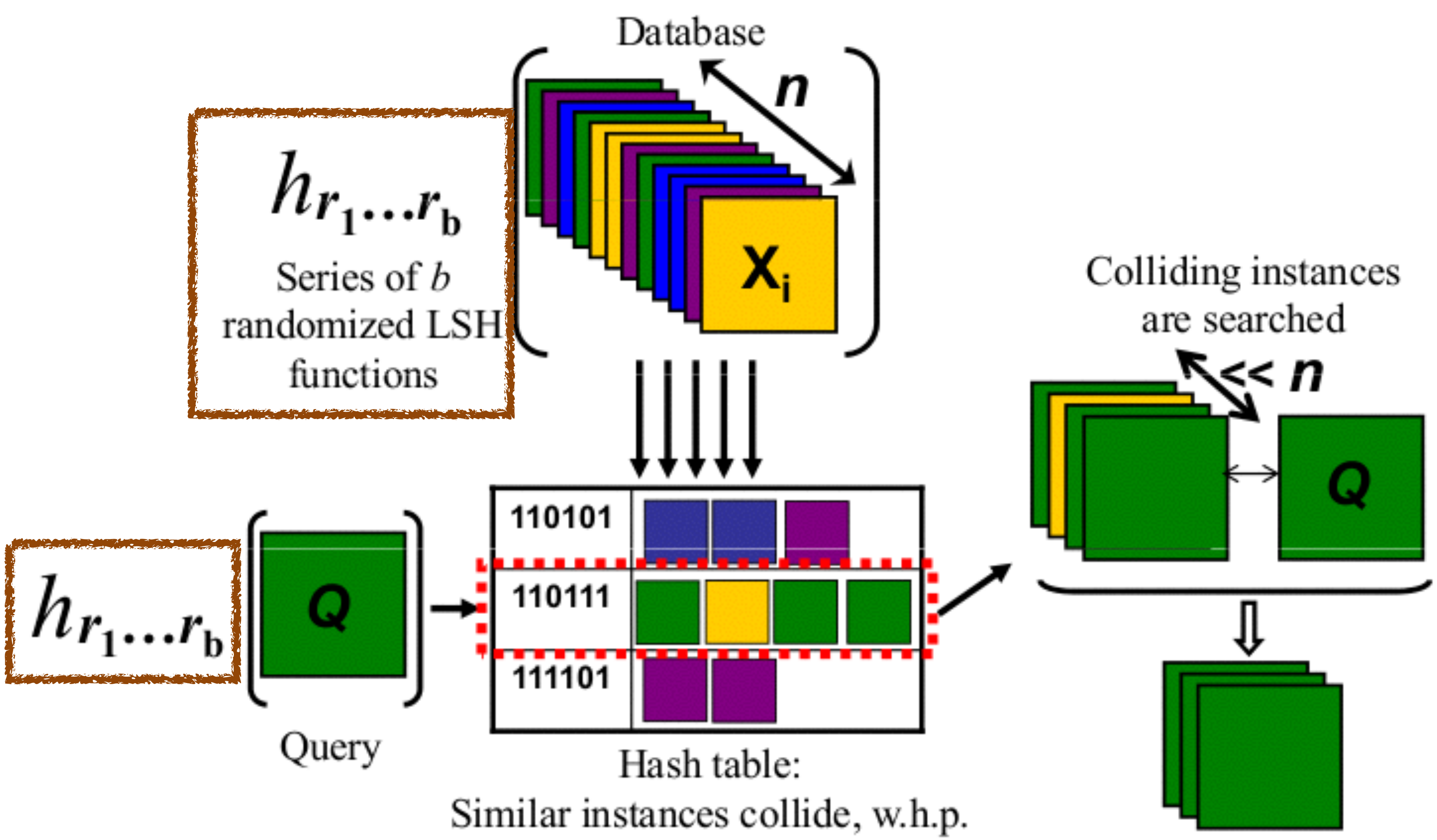
3 out of hundreds of unique words match (Jaccard < 0.03)

Example idea from: Manning et al. Introduction to Information Retrieval. 2009 *free PDF!*

Locality Sensitive Hashing (LSH) 1/2

- A **hashing** approach to group **near neighbors**.
- Map similar items (e.g., documents or words) into the same [hash] buckets.
- LSH "**maximizes**" (instead of minimizing) hash **collisions**.
- It therefore is a **dimensionality reduction** technique.
- For documents or words, **minhashing** can be used.
  - Approach from Rajaraman & Ullman, Mining of Massive Datasets, 2010
    - http://infolab.stanford.edu/~ullman/mmds/ch3a.pdf

Locality Sensitive Hashing (LSH) 2/2



M Vogiatzis. micvog.com 2013.

Min-hash signatures (1/2)

Create a
n-gram/k-shingle × document
matrix (likely **very** sparse!):

- **T**: shingle/n-gram in document
- **F**: shingle/n-gram not in document

Step 1/2 - permuting the row order:

n-gram/shingle ID

x	D_1	D_2	D_3	D_4	h_1	h_2
0	**T**	F	F	**T**	1	1
1	F	F	**T**	F	2	4
2	F	**T**	F	**T**	3	2
3	**T**	F	**T**	**T**	4	0
4	F	F	**T**	F	0	3

$h_1(x) = (x+1)\%n$

$h_2(x) = (3x+1)\%n$

with n=5 (rows)

hash-"permuted" row IDs

Lemma: Two docs will have the same first "true" shingle/n-gram when looking from top to bottom with a probability equal to their Jaccard (Set) Similarity.

a family of hash functions h_i

Idea: Create ~~sufficient **permutations**~~ of the row (shingle/n-gram) ordering so that the Jaccard Similarity can be approximated by comparing the number of coinciding vs. differing rows.

Min-hash signatures (2/2)

shingle or n-gram ID

S	D_1	D_2	D_3	D_4	h_1	h_2
0	T	F	F	T	1	1
1	F	F	T	F	2	4
2	F	T	F	T	3	2
3	T	F	T	T	4	0
4	F	F	T	F	0	3

$h_1(x) = (x+1)\%n$

$h_2(x) = (3x+1)\%n$

$n=5$

from step one

Step 2/2 - generating the hash signature:

(vertical, per-document)

minhash signatures:

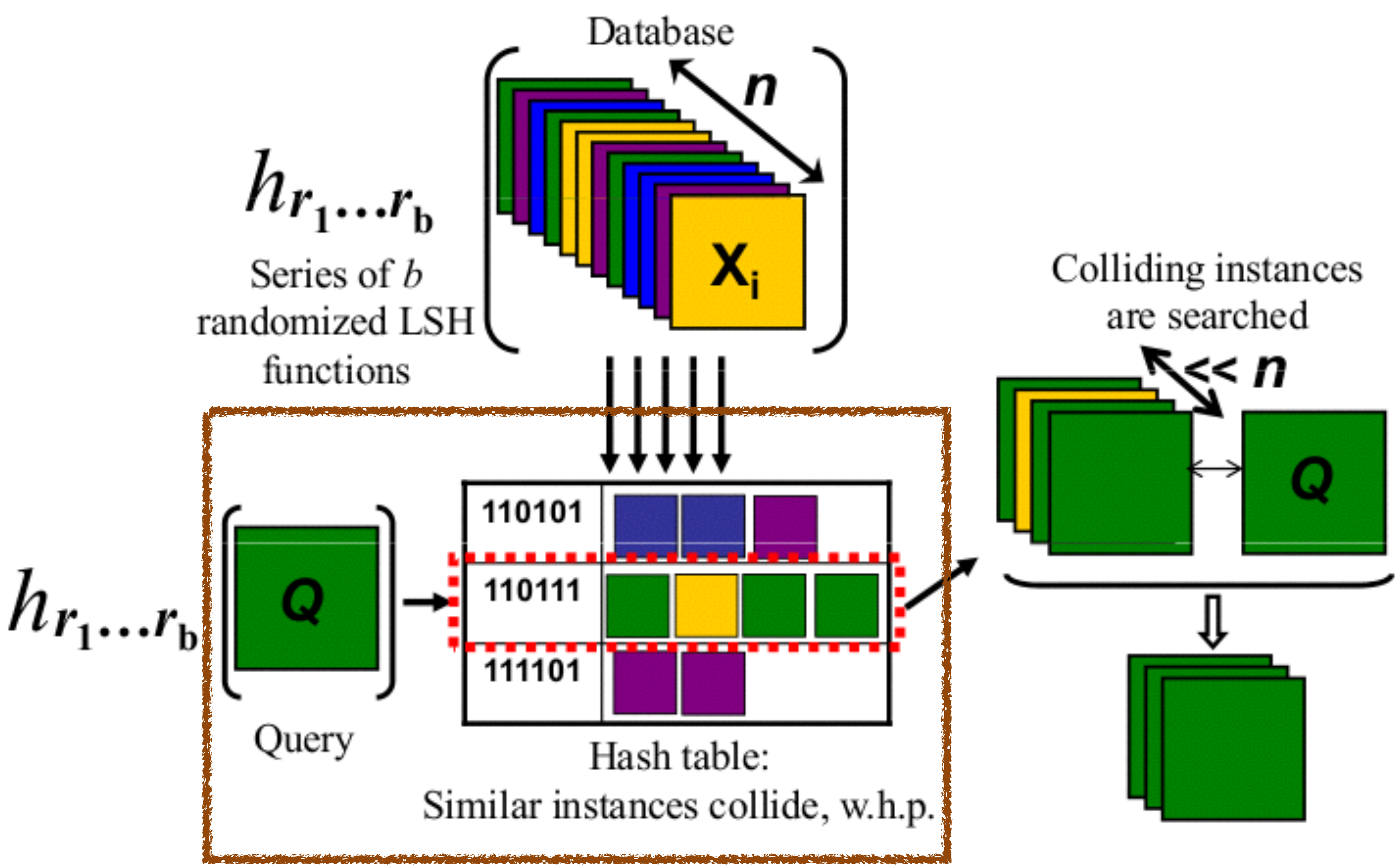
M	D_1	D_2	D_3	D_4
h_1	1	3	**0**	1
h_2	0	2	0	0

perfectly map-reduce-able and embarrassingly parallel!

```
init matrix M = ∞

for each shingle s:
 for each hash h:
  for each doc d:
   if S[d,s] and M[d,h] > h(s):
    M[d,h] = h(s)
```

Locality Sensitive Hashing (LSH) 2/2



M Vogiatzis. micvog.com 2013.

Banded Locality Sensitive Min-hashing

Bands		T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	…
1	h_1	**1**	0	2	**1**	7	1	4	5	
	h_2	**1**	2	4	**1**	6	5	5	6	
	h_3	**0**	5	6	**0**	6	4	7	9	
2	h_5	4	0	8	8	**7**	6	5	**7**	
	h_1	7	7	0	8	**3**	8	7	**3**	
	h_4	8	9	0	7	**2**	4	8	**2**	
3	h_2	8	5	**4**	0	9	8	**4**	7	
	h_6	9	4	**3**	9	0	8	**3**	9	
	h_7	8	5	**8**	0	0	6	**8**	0	
…	…									

typical:
$r=10$
$b=30$

(in at least one band)

Bands $b \propto p_{agreement}$

$p_{agreement} = \mathbf{1 - (1 - s^r)^b}$

Hashes/Band $r \propto 1/p_{agreement}$

$s = \text{Jaccard}(A,B)$

UnionFind to connect buckets across bands

	T_0	T_1	T_2	T_3	T_4	T_5	T_6	T_7	…
h_1	0	**1**	2	7	**1**	**1**	4	5	
h_2	2	**5**	4	6	**5**	**5**	5	6	
h_3	5	**0**	6	6	**0**	**0**	7	9	
h_5	4	0	**2**	8	8	**2**	5	5	
h_1	7	7	**1**	8	0	**1**	7	3	
h_6	8	9	**6**	7	0	**6**	8	8	
h_2	8	5	**4**	**4**	9	8	**4**	**4**	
h_5	9	4	**3**	**3**	0	1	**3**	**3**	
h_4	8	5	**8**	**8**	0	6	**8**	**8**	
…									

new "connection":
{ 2 5 }

Document "clusters"

{0}, {1 4 5}, {2 3 6 7} $\xrightarrow{\text{union (2,5)}}$ {0}, {1 2 3 4 5 6 7}