Text Vectorization

- Simple example: a list of course titles
- Each title as one document (data instance)

Text data on course titles
“INFO 111 Information and Systems”
“INFO 222 Data and Information”
“INFO 333 System Development and System Programming”

Text Vectorization: process

Text data on course titles
“INFO 111 Information and Systems”
“INFO 222 Data and Information”
“INFO 333 Systems and System Programming”

Tokenize (and perhaps normalize)

info, 111, information, and, system, info,
222, data, and, information,
Info, 333, system, and, system, programming

Sort and merge

111, 222, 333, and, data, info, information, program, system

Text Vectorization: transformation of data

@data
“INFO 111 Information and Systems”
“INFO 222 Data and Information”
“INFO 333 Systems and System Programming”

111	222	333	and	data	info	information	program	system
Vectorized data (numeric attributes)								
1	0	0	1	0	1	1	0	1
0	1	0	1	1	1	1	0	0
0	0	1	1	0	1	0	1	1

For now, it is **binary representation**:

- **1** if the token appears in the document
- **0** if it does not appear in the document

Text Vectorization: with Term Frequency (raw TF)

@data
“INFO 111 Information and Systems”
“INFO 222 Data and Information”
“INFO 333 Systems and System Programming”

111	222	333	and	data	info	information	program	system
@data								
1	0	0	1	0	1	1	0	1
0	1	0	1	1	1	1	0	0
0	0	1	1	0	1	0	1	**2**

“System” appears twice in the third course title.

idf weight

- df_t is the document frequency of *t*: the number of documents that contain *t*
  - df_t is an inverse measure of the informativeness of *t*
  - $\mathrm{df}_t \leq N$ *(the total number of documents in a collection)*
- We define the idf (inverse document frequency) of *t* by

$$\mathrm{idf}_t = \log_{10}(N/\mathrm{df}_t)$$

  - We use log (N/df_t) instead of N/df_t to “dampen” the effect of idf.

the base of the log doesn’t matter

In the example, *N* = 3

term	df_t	idf_t
data	1	0.48
program	1	0.48
system	2	0.18
info	3	0
and	3	0

$$\text{idf}_t = \log_{10} (N/\text{df}_t)$$

There is one idf value for each term *t* in a collection.

tf-idf weighting

- The tf-idf weight of a term is the product of its TF weight and its IDF weight.

$$w_{t,d} = (1 + \log \mathrm{tf}_{t,d}) \times \log_{10}(N / \mathrm{df}_t)$$

or $w_{t,d} = \mathrm{tf}_{t,d} \times \log_{10}(N/\mathrm{df}_t)$

  - Best known weighting scheme in information retrieval
  - Alternative names: tf.idf, tf x idf, TF*IDF, etc.
  - Increases with the number of occurrences within a document
  - Increases with the rarity of the term in the collection

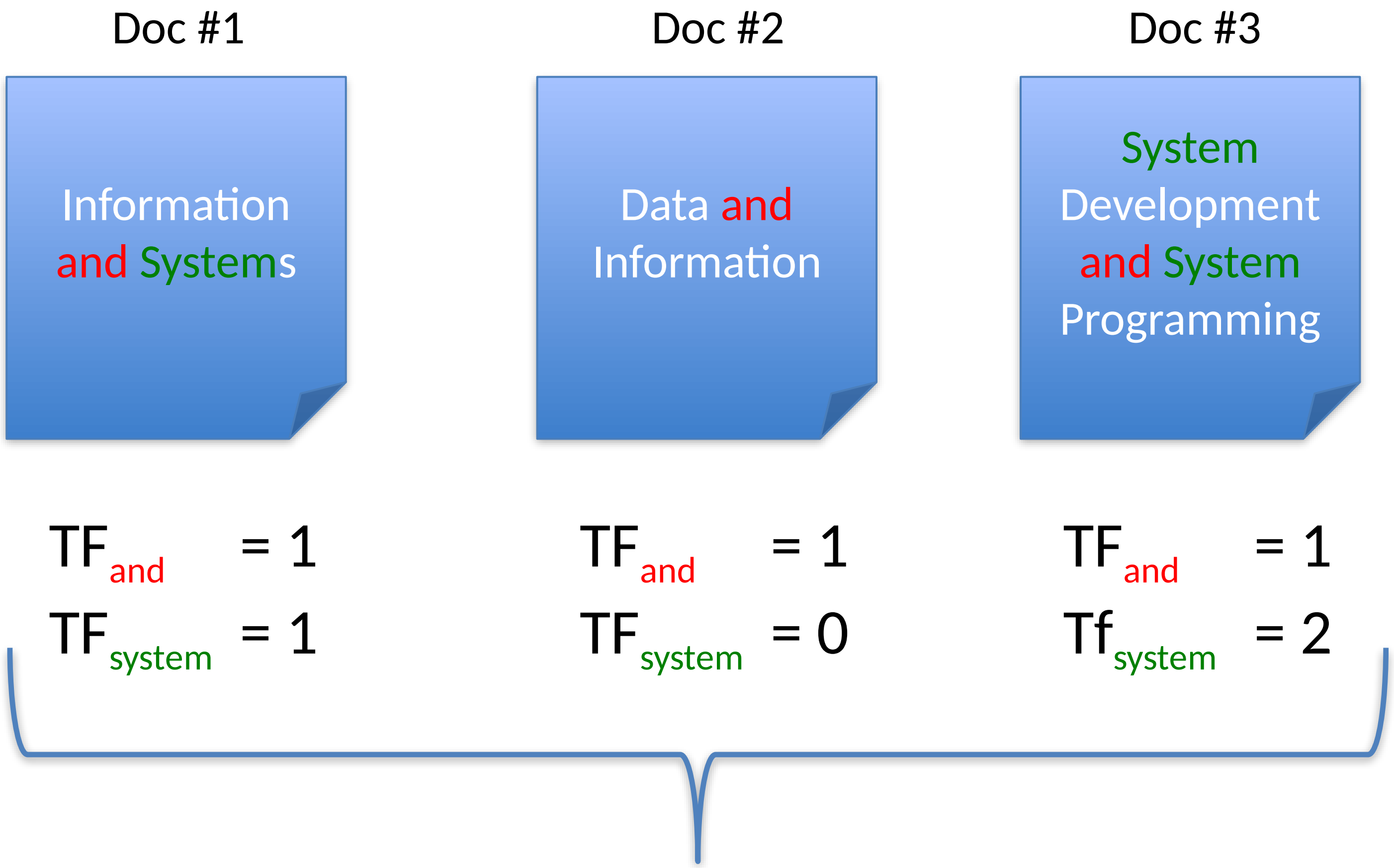
Text Vectorization: with TF*IDF

Data
“INFO 111 Information and Systems”
“INFO 222 Data and Information”
“INFO 333 Systems and System Programming”

111	222	333	and	data	info	information	program	system

Vectorized Data with TF*IDF								
0.48	0	0	0	0	0	0.18	0	0.18
0	0.48	0	0	**0.48**	0	0.18	0	0
0	0	0.48	0	0	0	0	0.18	**0.36**

Visualize TF and DF



Doc #1

Information and Systems

Doc #2

Data and Information

Doc #3

System Development and System Programming

Term Freq:

$TF_{and} = 1$

$TF_{system} = 1$

$TF_{and} = 1$

$TF_{system} = 0$

$TF_{and} = 1$

$Tf_{system} = 2$

Doc Freq:

$DF_{and} = 3$

$DF_{system} = 2$

out of 3 documents in the collection

Cosine Similarity

$$\cos(\vec{q},\vec{d}) = \frac{\sum_{i=1}^{n} q_i d_i}{\sqrt{\sum_{i=1}^{n} q_i^2}\sqrt{\sum_{i=1}^{n} d_i^2}}$$

Query **q**
data
information
system

	111	222	333	and	data	info	information	program	system
d1	0.48	0	0	0	0	0	0.18	0	0.18
d2	0	0.48	0	0	**0.48**	0	0.18	0	0
d3	0	0	0.48	0	0	0	0	0.18	**0.36**