

<http://poloclub.gatech.edu/cse6242>

CSE6242 / CX4242: Data & Visual Analytics

Text Analytics (Text Mining)

Concepts, Algorithms, LSI/SVD

Duen Horng (Polo) Chau

Assistant Professor

Associate Director, MS Analytics

Georgia Tech

Partly based on materials by

Professors Guy Lebanon, Jeffrey Heer, John Stasko, Christos Faloutsos, Parishit Ram (GT PhD alum; SkyTree), Alex Gray

Text is everywhere

We use documents as primary information artifact in our lives

Our access to documents has grown tremendously thanks to the Internet

- *WWW*: webpages, Twitter, Facebook, Wikipedia, Blogs, ...
- *Digital libraries*: Google books, ACM, IEEE, ...
- Lyrics, closed caption... (youtube)
- Police case reports
- Legislation (law)
- Reviews (products, rotten tomatoes)
- Medical reports (EHR - electronic health records)
- Job descriptions

Big (Research) Questions

... in understanding and gathering information from text and document collections

- establish authorship, authenticity; plagiarism detection
- classification of genres for narratives (e.g., books, articles)
- tone classification; sentiment analysis (online reviews, twitter, social media)
- code: syntax analysis (e.g., find common bugs from students' answers)

Popular Natural Language Processing (NLP) libraries

- **Stanford NLP**

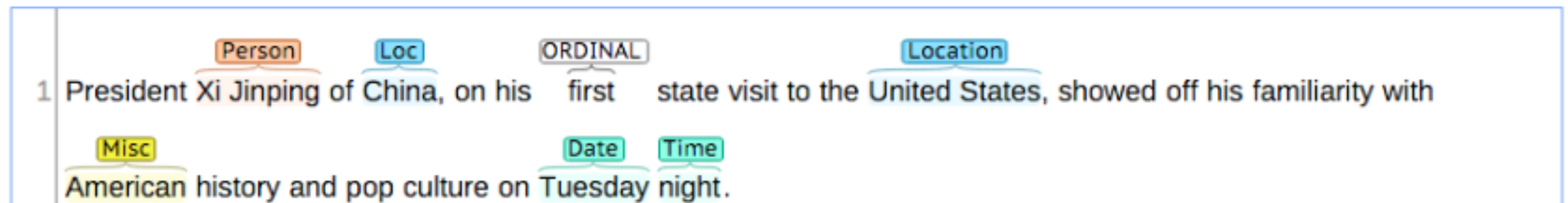
- **OpenNLP**

- **NLTK (python)**

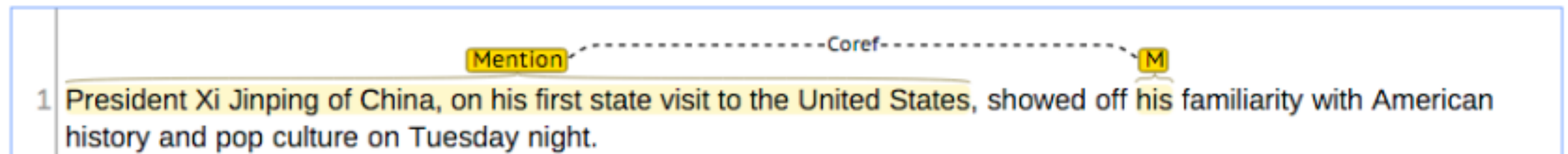
tokenization, sentence segmentation, part-of-speech tagging, named entity extraction, chunking, parsing

Named Entity Recognition:

Image source: <https://stanfordnlp.github.io/CoreNLP/>



Coreference:



Basic Dependencies:

Outline

- **Preprocessing** (e.g., stemming, remove stop words)
- **Document representation** (most common: bag-of-words model)
- **Word importance** (e.g., word count, TF-IDF)
- **Latent Semantic Indexing** (find “concepts” among documents and words), which helps with **retrieval**

To learn more: Prof. Jacob Eisenstein's
CS 4650/7650 Natural Language Processing

Stemming

Reduce words to their **stems** (or base forms)

Words: compute, computing, computer, ...

Stem: comput

Several classes of algorithms to do this:

- Stripping suffixes, lookup-based, etc.

<http://en.wikipedia.org/wiki/Stemming>

Stop words: http://en.wikipedia.org/wiki/Stop_words

Bag-of-words model

Represent each **document** as a **bag of words**, ignoring words' ordering. Why? For **simplicity**.

Unstructured text becomes **a vector of numbers**

e.g., docs: “I like visualization”, “I like data”.

1 : “I”

2 : “like”

3 : “data”

4 : “visualization”

“I like visualization” \Rightarrow [1, 1, 0, 1]

“I like data” \Rightarrow [1, 1, 1, 0]

TF-IDF

A word's importance score in a document, among N documents

When to use it? Everywhere you use “word count”, you can likely use TF-IDF.

TF: term frequency

= #appearance a document

(high, if terms appear many times in this document)

IDF: inverse document frequency

= $\log(N / \text{\#document containing that term})$

(penalize “common” words appearing in almost any documents)

Final score = TF * IDF

(higher score \Rightarrow more “characteristic”)

Vector Space Model

Why?

Each document \Rightarrow vector

Each query \Rightarrow vector

Search for documents \Rightarrow find “similar” vectors

Cluster documents \Rightarrow cluster “similar” vectors

Latent Semantic Indexing (LSI)

Main idea

- map each **document** into some ‘**concepts**’
- map each **term** into some ‘**concepts**’

‘**Concept**’ : ~ a set of terms, with weights.

For example, **DBMS_concept**:

“data” (0.8),

“system” (0.5),

Latent Semantic Indexing (LSI)

~ pictorially (before) ~

document-term matrix

	data	system	retireval	lung	ear
doc1	1	1	1		
doc2	1	1	1		
doc3				1	1
doc4				1	1

Latent Semantic Indexing (LSI)

~ pictorially (after) ~

term-concept
matrix

	database concept	medical concept
data	1	
system	1	
retrieval	1	
lung		1
ear		1

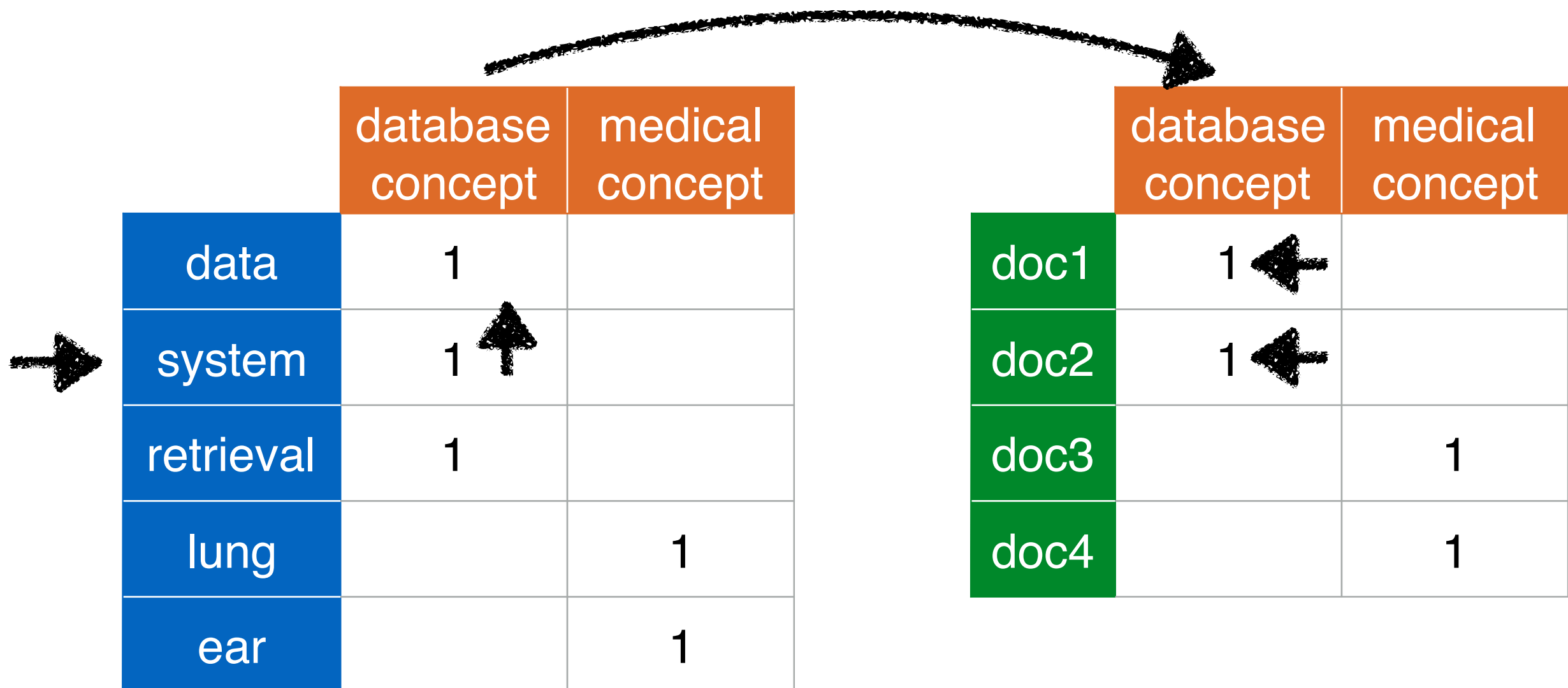
... and
document-concept
matrix

	database concept	medical concept
doc1	1	
doc2	1	
doc3		1
doc4		1

Latent Semantic Indexing (LSI)

Q: How to search, e.g., for “system”?

A: find the corresponding concept(s); and the corresponding documents



Latent Semantic Indexing (LSI)

Works like an **automatically constructed thesaurus**

We may retrieve documents that **DON'T** have the term “system”, but they contain almost everything else (“data”, “retrieval”)

LSI - Discussion

Great idea,

- to derive ‘**concepts**’ from documents
- to build a ‘**thesaurus**’ automatically
- to reduce dimensionality (down to few “concepts”)

How does LSI work?

Uses **Singular Value Decomposition** (SVD)

Singular Value Decomposition (SVD)

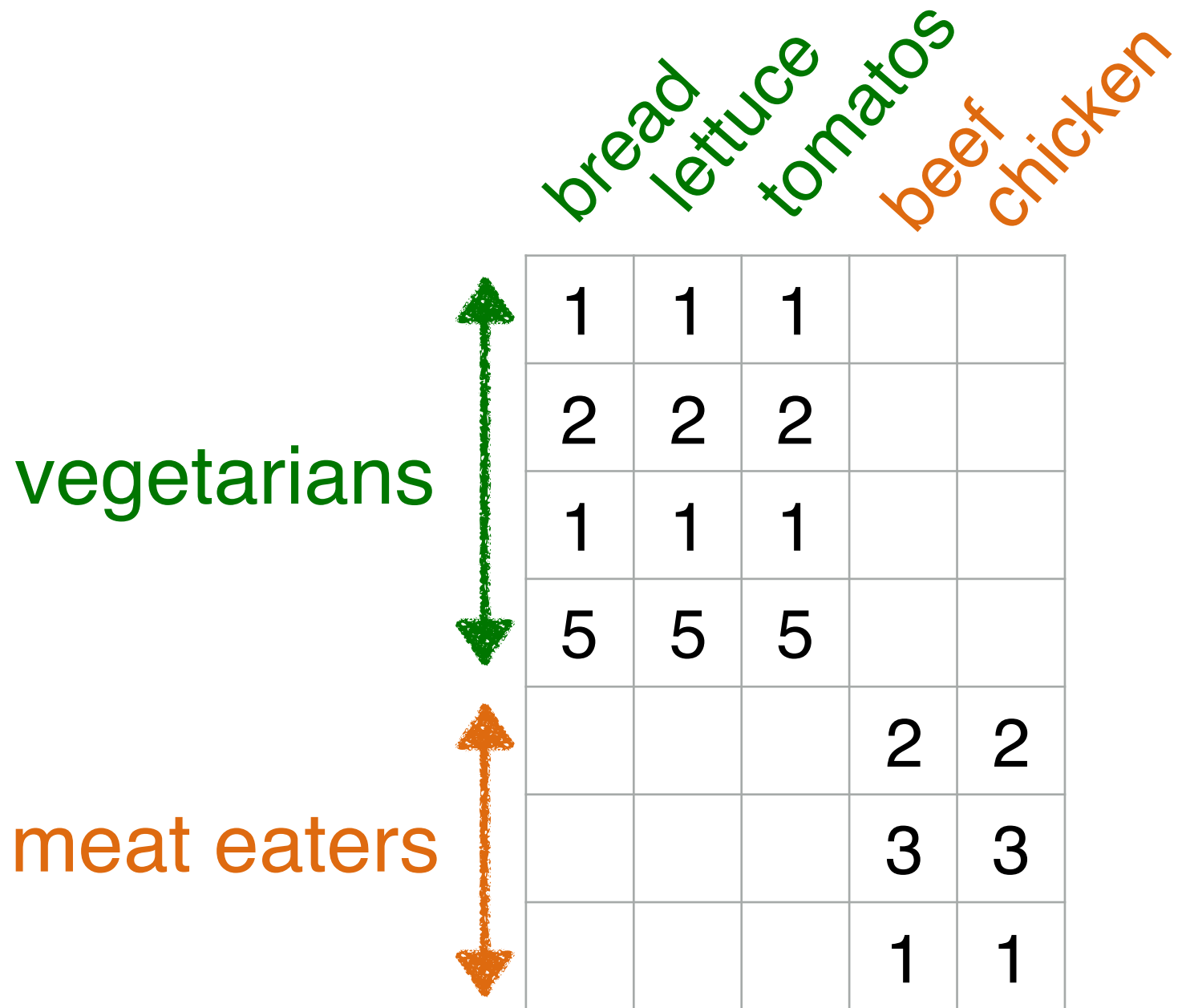
Motivation

Problem #1

Find “concepts”
in matrices

Problem #2

Compression /
dimensionality
reduction



	bread	lettuce	tomatos	beef	chicken
vegetarians	1	1	1		
	2	2	2		
	1	1	1		
	5	5	5		
meat eaters				2	2
				3	3
				1	1

SVD is a **powerful,** **generalizable** technique.

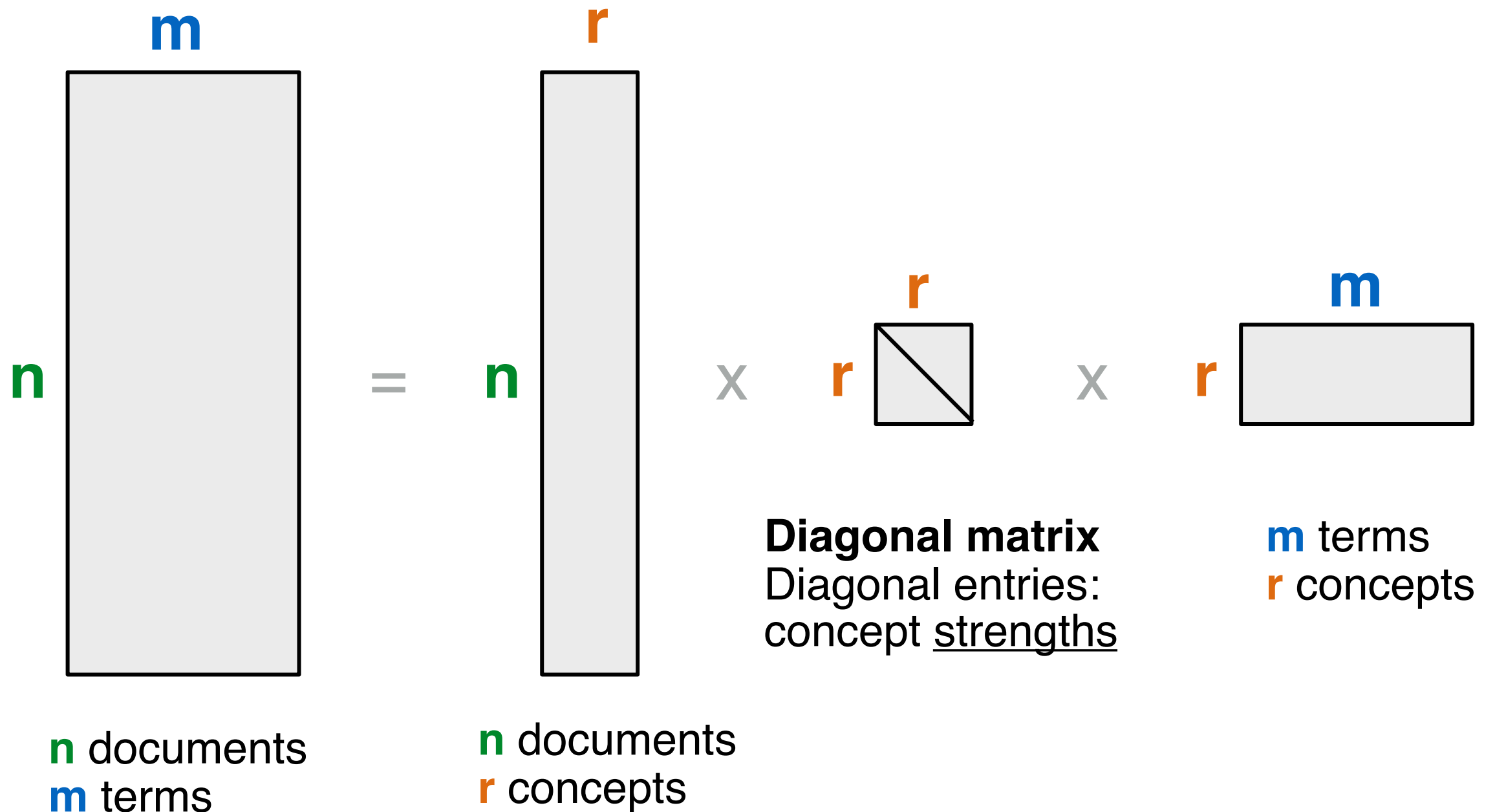
Songs / Movies / Products

Customers

1	1	1		
2	2	2		
1	1	1		
5	5	5		
			2	2
			3	3
			1	1

SVD Definition (pictorially)

$$\mathbf{A}_{[n \times m]} = \mathbf{U}_{[n \times r]} \mathbf{\Lambda}_{[r \times r]} (\mathbf{V}_{[m \times r]})^T$$



SVD Definition (in words)

$$\mathbf{A}_{[n \times m]} = \mathbf{U}_{[n \times r]} \mathbf{\Lambda}_{[r \times r]} (\mathbf{V}_{[m \times r]})^T$$

A: n x m matrix

e.g., n documents, m terms

U: n x r matrix

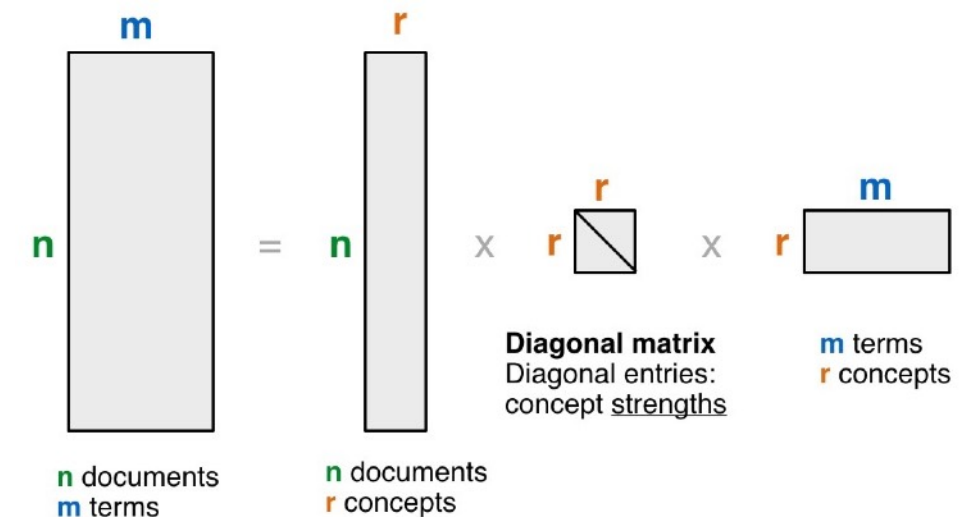
e.g., n documents, r concepts

$\mathbf{\Lambda}$: r x r diagonal matrix

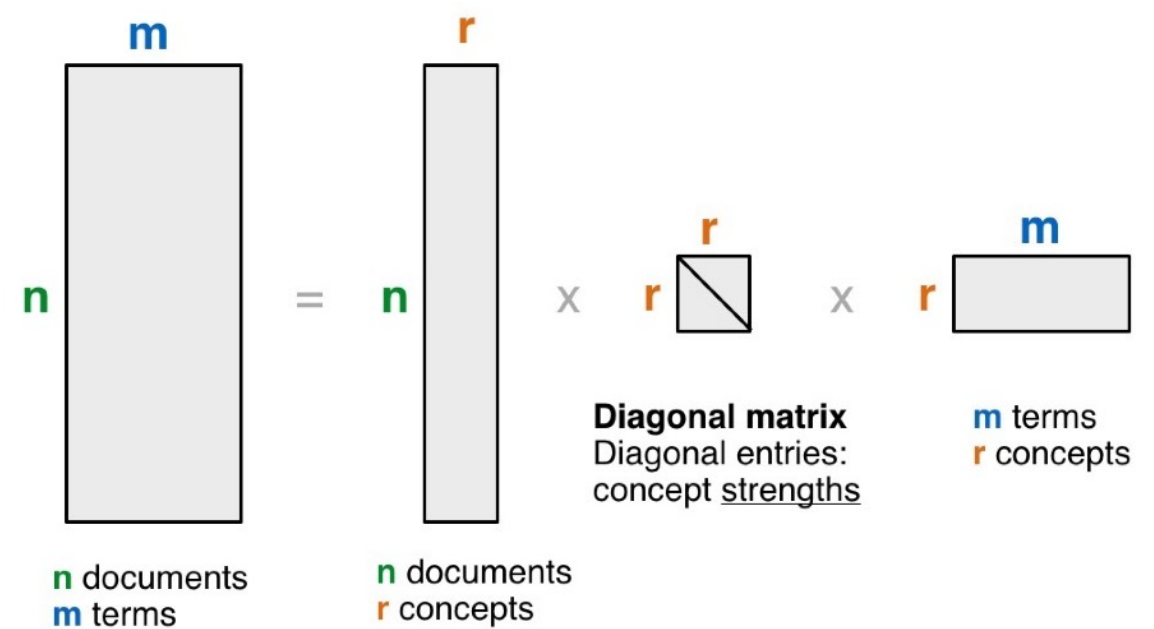
r : rank of the matrix; strength of each 'concept'

V: m x r matrix

e.g., m terms, r concepts



SVD - Properties



THEOREM [Press+92]:

always possible to decompose matrix **A** into

$$A = U \Lambda V^T$$

U, Λ , V: **unique**, most of the time

U, V: column **orthonormal**

i.e., columns are **unit vectors**, and **orthogonal** to each other

$$U^T U = I$$

$$V^T V = I \quad (I: \text{identity matrix})$$

Λ : **diagonal** matrix with non-negative diagonal entries, sorted in **decreasing order**

SVD - Example

	data	info	retrieval	brain	lung
CS docs	1	1	1	0	0
	2	2	2	0	0
	1	1	1	0	0
	5	5	5	0	0
MD docs	0	0	0	2	2
	0	0	0	3	3
	0	0	0	1	1

=

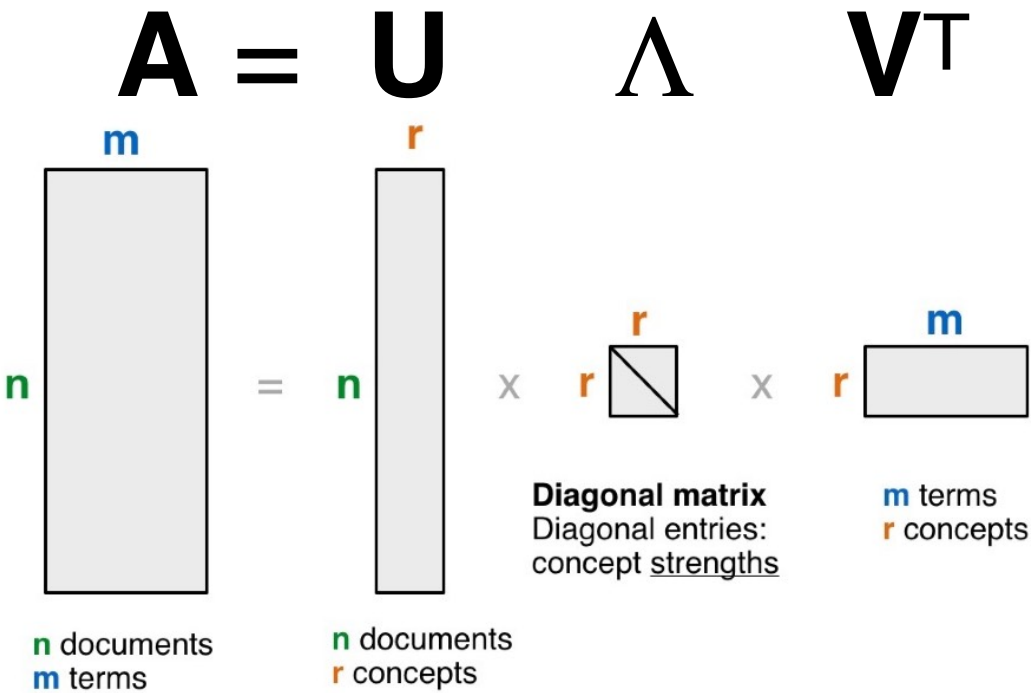
0.18	0
0.36	0
0.18	0
0.90	0
0	0.53
0	0.80
0	0.27

x

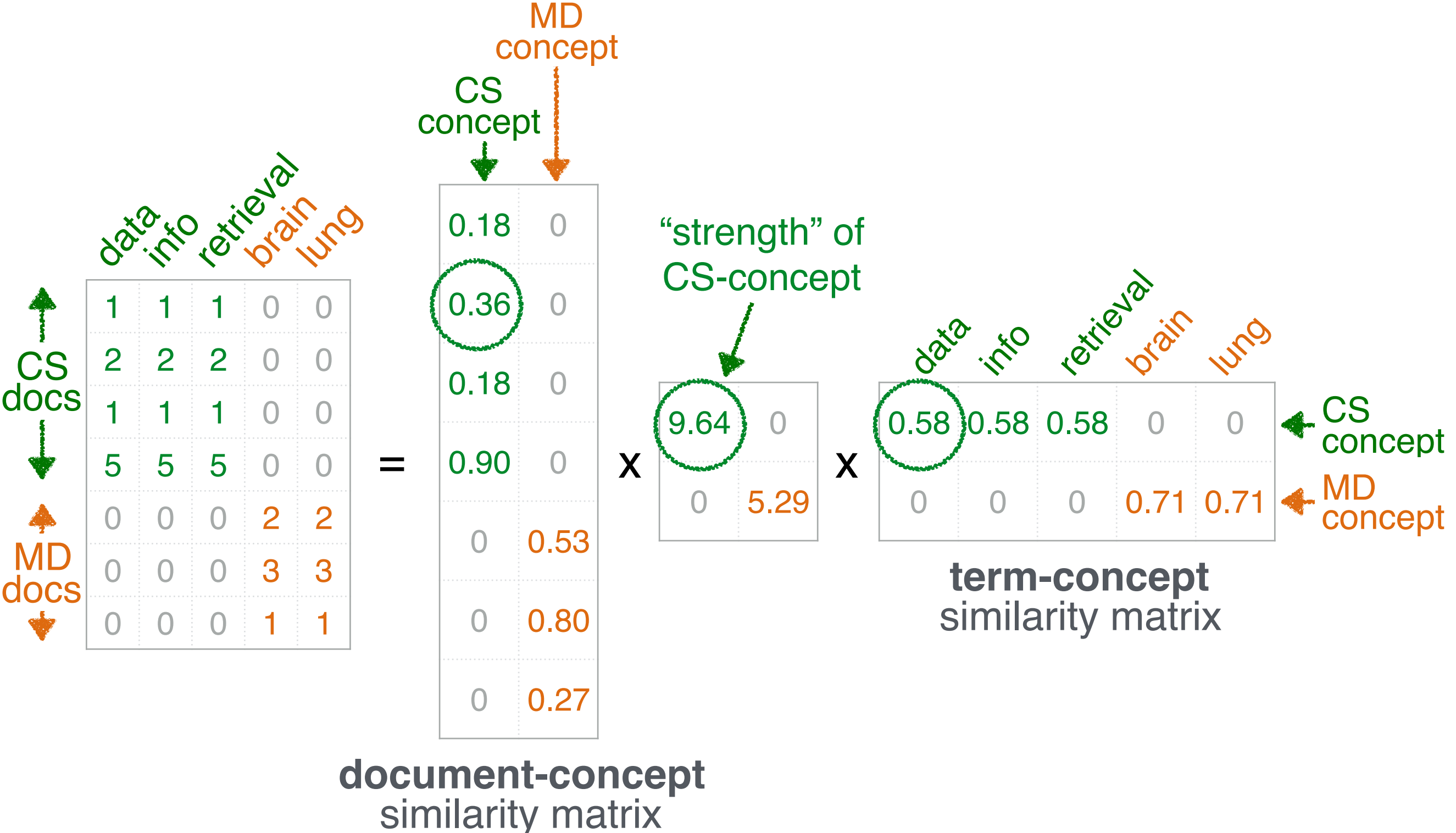
9.64	0
0	5.29

x

0.58	0.58	0.58	0	0
0	0	0	0.71	0.71



SVD - Example



SVD - Interpretation #1

‘documents’, ‘terms’ and ‘concepts’:

U: document-concept similarity matrix

V: term-concept similarity matrix

Λ : diagonal elements: concept “strengths”

SVD - Interpretation #1

‘documents’, ‘terms’ and ‘concepts’:

Q: if \mathbf{A} is the document-to-term matrix,
what is the similarity matrix $\mathbf{A}^T \mathbf{A}$?

A:

Q: $\mathbf{A} \mathbf{A}^T$?

A:

SVD - Interpretation #1

‘documents’, ‘terms’ and ‘concepts’:

Q: if **A** is the document-to-term matrix,
what is the similarity matrix **A^T A** ?

A: term-to-term ($[m \times m]$) similarity matrix

Q: **A A^T** ?

A: document-to-document ($[n \times n]$) similarity matrix

SVD properties

V are the eigenvectors of the *covariance matrix* $\mathbf{A}^T \mathbf{A}$

$$\mathbf{A}^T \mathbf{A} = (\mathbf{U} \mathbf{\Sigma} \mathbf{V}^T)^T (\mathbf{U} \mathbf{\Sigma} \mathbf{V}^T) = \mathbf{V} \mathbf{\Sigma}^2 \mathbf{V}^T$$

U are the eigenvectors of the *Gram (inner-product) matrix* $\mathbf{A} \mathbf{A}^T$

$$\mathbf{A} \mathbf{A}^T = (\mathbf{U} \mathbf{\Sigma} \mathbf{V}^T) (\mathbf{U} \mathbf{\Sigma} \mathbf{V}^T)^T = \mathbf{U} \mathbf{\Sigma}^2 \mathbf{U}^T$$

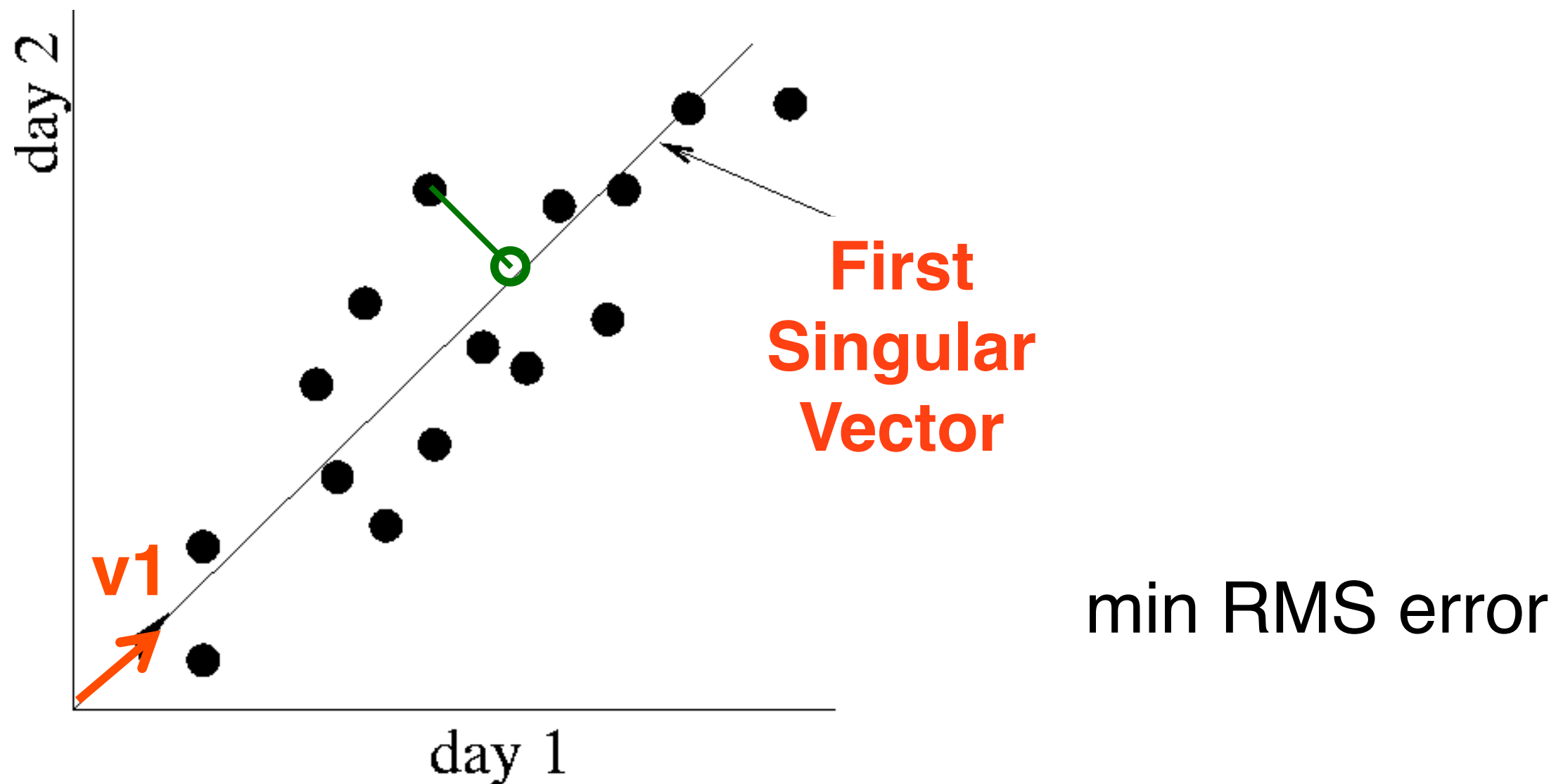
SVD is closely related to PCA, and can be numerically more stable.
For more info, see:

<http://math.stackexchange.com/questions/3869/what-is-the-intuitive-relationship-between-svd-and-pca> Ian T. Jolliffe, Principal Component Analysis (2nd ed), Springer, 2002. Gilbert Strang, Linear Algebra and Its Applications (4th ed), Brooks Cole, 2005.

SVD - Interpretation #2

Find the best axis to project on.

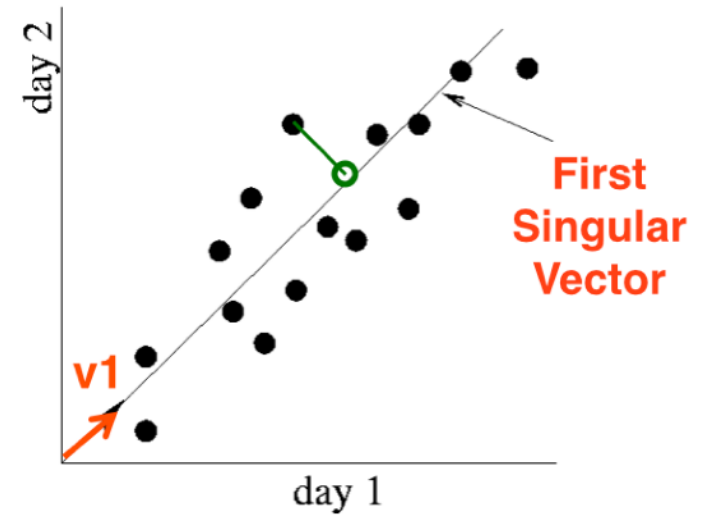
(‘best’ = min sum of squares of projection errors)



Beautiful visualization explaining PCA:
<http://setosa.io/ev/principal-component-analysis/>

SVD - Interpretation #2

$$\mathbf{A} = \mathbf{U} \mathbf{\Lambda} \mathbf{V}^T$$



1	1	1	0	0
2	2	2	0	0
1	1	1	0	0
5	5	5	0	0
0	0	0	2	2
0	0	0	3	3
0	0	0	1	1

=

0.18	0
0.36	0
0.18	0
0.90	0
0	0.53
0	0.80
0	0.27

x

9.64	0
0	5.29

x

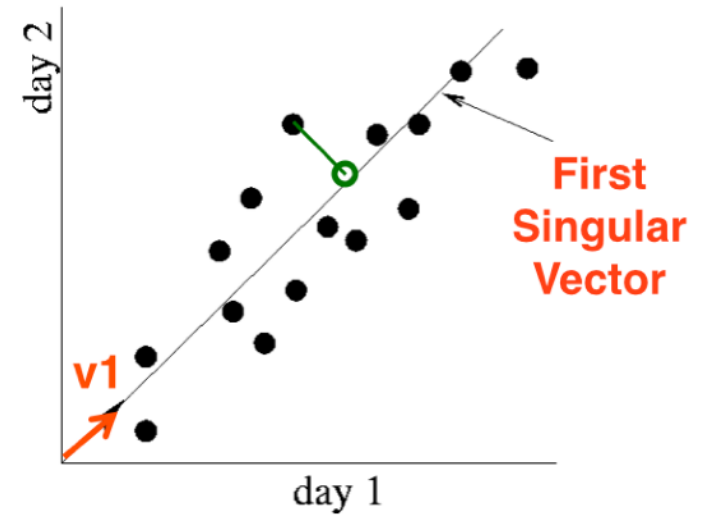
0.58	0.58	0.58	0	0
0	0	0	0.71	0.71

variance ('spread')
on the v1 axis

v1

SVD - Interpretation #2

$U \Lambda$ gives the **coordinates** of the points in the projection axis



$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 2 & 2 & 2 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 3 & 3 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0.18 & 0 \\ 0.36 & 0 \\ 0.18 & 0 \\ 0.90 & 0 \\ 0 & 0.53 \\ 0 & 0.80 \\ 0 & 0.27 \end{bmatrix} \times \begin{bmatrix} 9.64 & 0 \\ 0 & 5.29 \end{bmatrix} \times \begin{bmatrix} 0.58 & 0.58 & 0.58 & 0 & 0 \\ 0 & 0 & 0 & 0.71 & 0.71 \end{bmatrix}$$

$v1$

SVD - Interpretation #2

More details

Q: how exactly is dim. reduction done?

1	1	1	0	0
2	2	2	0	0
1	1	1	0	0
5	5	5	0	0
0	0	0	2	2
0	0	0	3	3
0	0	0	1	1

=

0.18	0
0.36	0
0.18	0
0.90	0
0	0.53
0	0.80
0	0.27

x

9.64	0
0	5.29

x

0.58	0.58	0.58	0	0
0	0	0	0.71	0.71

SVD - Interpretation #2

More details

Q: how exactly is dim. reduction done?

A: set the smallest singular values to zero:

1	1	1	0	0
2	2	2	0	0
1	1	1	0	0
5	5	5	0	0
0	0	0	2	2
0	0	0	3	3
0	0	0	1	1

=

0.18	0
0.36	0
0.18	0
0.90	0
0	0.53
0	0.80
0	0.27

x

9.64	0
0	5.19

x

0.58	0.58	0.58	0	0
0	0	0	0.71	0.71

SVD - Interpretation #2

More details

Q: how exactly is dim. reduction done?

A: set the smallest singular values to zero:

The diagram illustrates the process of dimensionality reduction in SVD. It shows the decomposition of a matrix into three components: U , Σ , and V^T . The matrix U is a 7x5 matrix with columns of varying magnitudes. The matrix Σ is a 7x5 diagonal matrix with singular values. The matrix V^T is a 5x5 matrix. The singular values are highlighted in green, and the smallest ones are highlighted in orange. Red boxes and 'X' marks indicate which singular values are being set to zero for dimensionality reduction.

1	1	1	0	0
2	2	2	0	0
1	1	1	0	0
5	5	5	0	0
0	0	0	2	2
0	0	0	3	3
0	0	0	1	1

=

0.18	0
0.36	0
0.18	0
0.90	0
0	0.53
0	0.80
0	0.27

x

9.64	0
0	5.19

x

0.58	0.58	0.58	0	0
0	0	0	0.71	0.71

SVD - Interpretation #2

More details

Q: how exactly is dim. reduction done?

A: set the smallest singular values to zero:

1	1	1	0	0
2	2	2	0	0
1	1	1	0	0
5	5	5	0	0
0	0	0	2	2
0	0	0	3	3
0	0	0	1	1

=

0.18	
0.36	
0.18	
0.90	
0	
0	
0	

x

9.64	0
0	

x

0.58	0.58	0.58	0	0

SVD - Interpretation #2

More details

Q: how exactly is dim. reduction done?

A: set the smallest singular values to zero:

1	1	1	0	0
2	2	2	0	0
1	1	1	0	0
5	5	5	0	0
0	0	0	2	2
0	0	0	3	3
0	0	0	1	1

~

1	1	1	0	0
2	2	2	0	0
1	1	1	0	0
5	5	5	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

SVD - Interpretation #3

finds non-zero 'blobs' in a data matrix

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 2 & 2 & 2 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 3 & 3 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0.18 & 0 \\ 0.36 & 0 \\ 0.18 & 0 \\ 0.90 & 0 \\ 0 & 0.53 \\ 0 & 0.80 \\ 0 & 0.27 \end{bmatrix} \times \begin{bmatrix} 9.64 & 0 \\ 0 & 5.29 \end{bmatrix} \times \begin{bmatrix} 0.58 & 0.58 & 0.58 & 0 & 0 \\ 0 & 0 & 0 & 0.71 & 0.71 \end{bmatrix}$$

SVD - Interpretation #3

finds non-zero 'blobs' in a data matrix

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 2 & 2 & 2 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ \hline 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 3 & 3 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0.18 & 0 \\ 0.36 & 0 \\ 0.18 & 0 \\ 0.90 & 0 \\ 0 & 0.53 \\ 0 & 0.80 \\ 0 & 0.27 \end{bmatrix} \times \begin{bmatrix} 9.64 & 0 \\ 0 & 5.29 \end{bmatrix} \times \begin{bmatrix} 0.58 & 0.58 & 0.58 & 0 & 0 \\ 0 & 0 & 0 & 0.71 & 0.71 \end{bmatrix}$$

SVD - Interpretation #3

- finds non-zero 'blobs' in a data matrix =
- 'communities' (bi-partite cores, here)

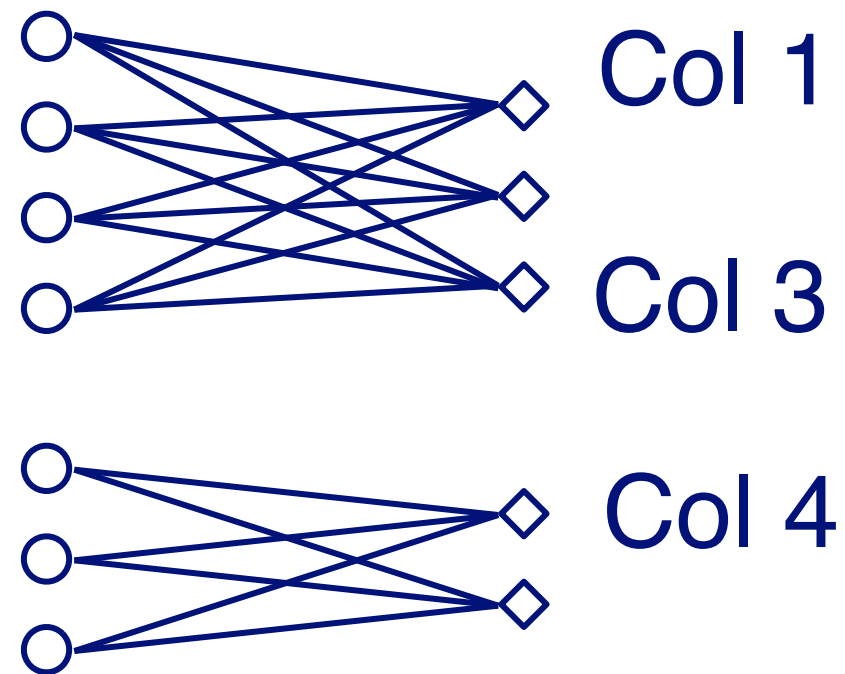
1	1	1	0	0
2	2	2	0	0
1	1	1	0	0
5	5	5	0	0
			2	2
0	0	0	3	3
0	0	0	1	1

Row 1

Row 4

Row 5

Row 7



SVD - Complexity

$O(n*m*m)$ or $O(n*n*m)$ (whichever is less)

Faster version, if just want singular values
or if we want first k singular vectors
or if the matrix is sparse [Berry]

No need to write your own!

Available in most linear algebra packages
(LINPACK, matlab, Splus/R,
mathematica ...)

Case Study

How to do queries with LSI?

Case Study

How to do queries with LSI?

For example, how to find documents with 'data'?

Diagram illustrating the LSI query process for finding documents related to 'data'.

Input Matrices:

- CS docs (Green):** A 6x5 matrix representing document counts for categories: data, info, retrieval, brain, lung.
- MD docs (Orange):** A 6x5 matrix representing document counts for categories: data, info, retrieval, brain, lung.

Matrix Dimensions: The CS docs matrix is 6x5, and the MD docs matrix is 6x5.

LSI Calculation:

$$\text{CS docs} \times \text{MD docs} = \text{Result Matrix}$$

Result Matrix (6x2):

0.18	0
0.36	0
0.18	0
0.90	0
0	0.53
0	0.80
0	0.27

Intermediate Matrix (2x2):

9.64	0
0	5.29

Final Matrix (2x5):

0.58	0.58	0.58	0	0
0	0	0	0.71	0.71

Case Study

How to do queries with LSI?

For example, how to find documents with 'data'?

A: map query vectors into 'concept space' – how?

Diagram illustrating the LSI query process:

CS docs (Concept Space Documents) and **MD docs** (Meaning Dimensions Documents) are mapped into the concept space.

CS docs matrix:

	data	info	retrieval	brain	lung
1	1	1	1	0	0
2	2	2	2	0	0
1	1	1	1	0	0
5	5	5	5	0	0
0	0	0	0	2	2
0	0	0	0	3	3
0	0	0	0	1	1

MD docs matrix:

0.18	0
0.36	0
0.18	0
0.90	0
0	0.53
0	0.80
0	0.27

Intermediate matrices:

Matrix 1:

9.64	0
0	5.29

Matrix 2:

0.58	0.58	0.58	0	0
0	0	0	0.71	0.71

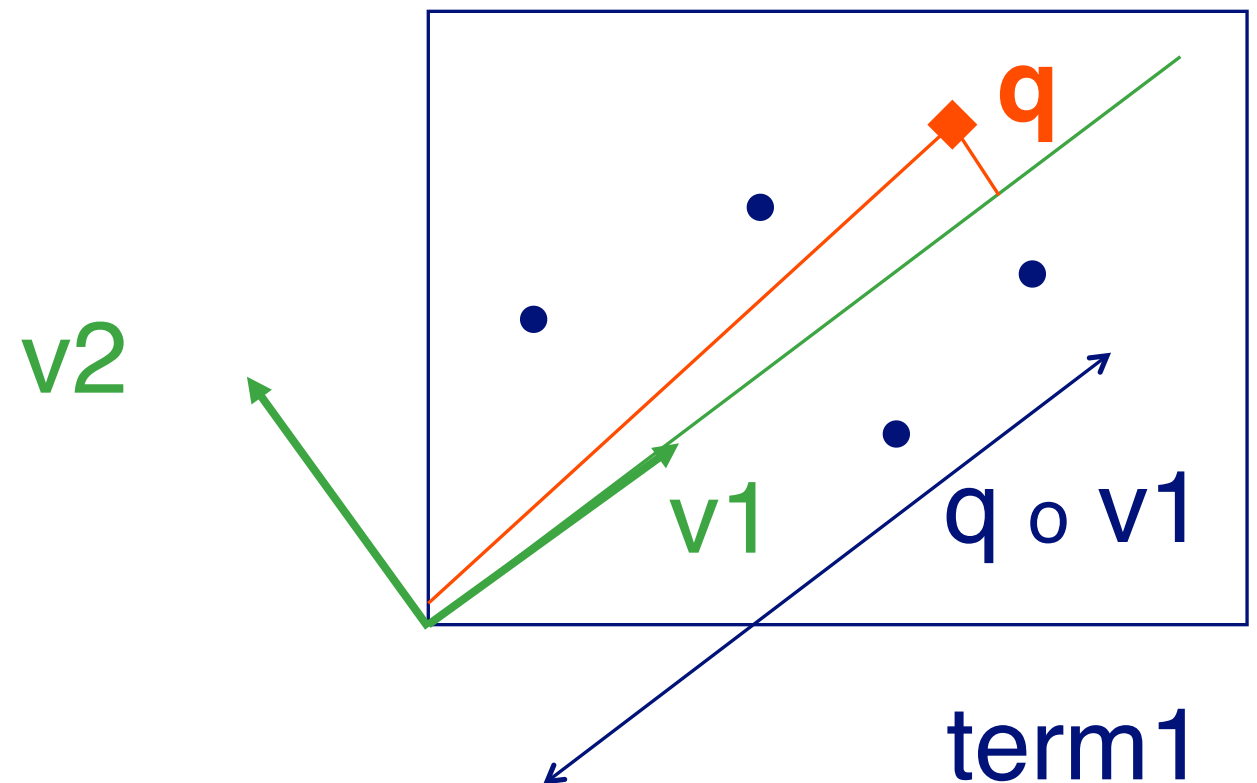
The final result is the product of the CS docs matrix, the MD docs matrix, and the intermediate matrices.

Case Study

How to do queries with LSI?

For example, how to find documents with ‘data’?
A: map query vectors into ‘concept space’, using **inner product** (cosine similarity) with each ‘concept’ vector v_i

$$\mathbf{q} = \begin{array}{c} \text{data} \\ \text{info} \\ \text{retrieval} \\ \text{brain} \\ \text{lung} \end{array} \begin{array}{|c|c|c|c|c|} \hline 1 & 0 & 0 & 0 & 0 \\ \hline \end{array}$$

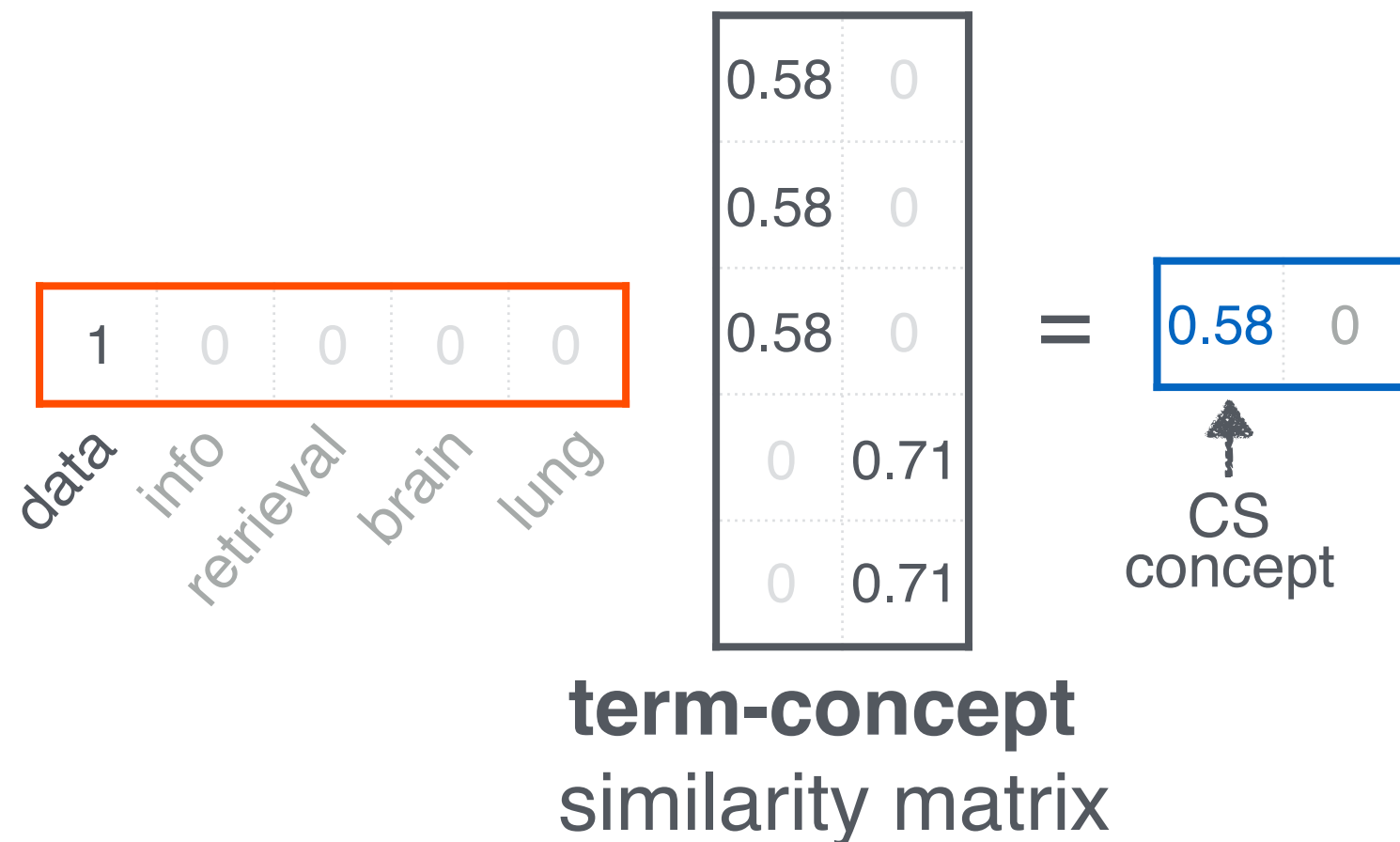


Case Study

How to do queries with LSI?

Compactly, we have:

$$\mathbf{q} \mathbf{V} = \mathbf{q}_{\text{concept}}$$



Case Study

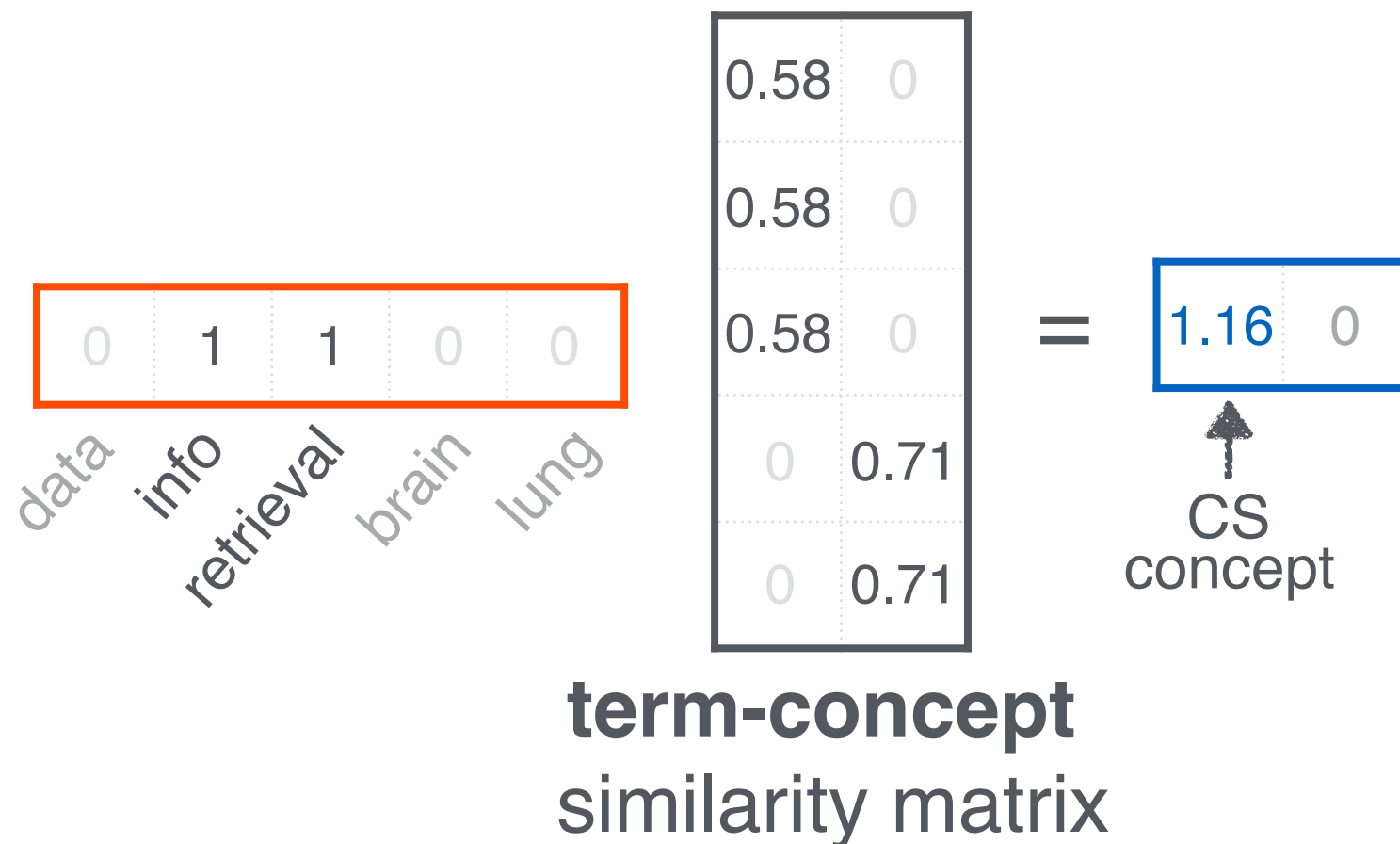
**How would the document
(‘information’, ‘retrieval’) be handled?**

Case Study

**How would the document
(‘information’, ‘retrieval’) be handled?**

SAME!

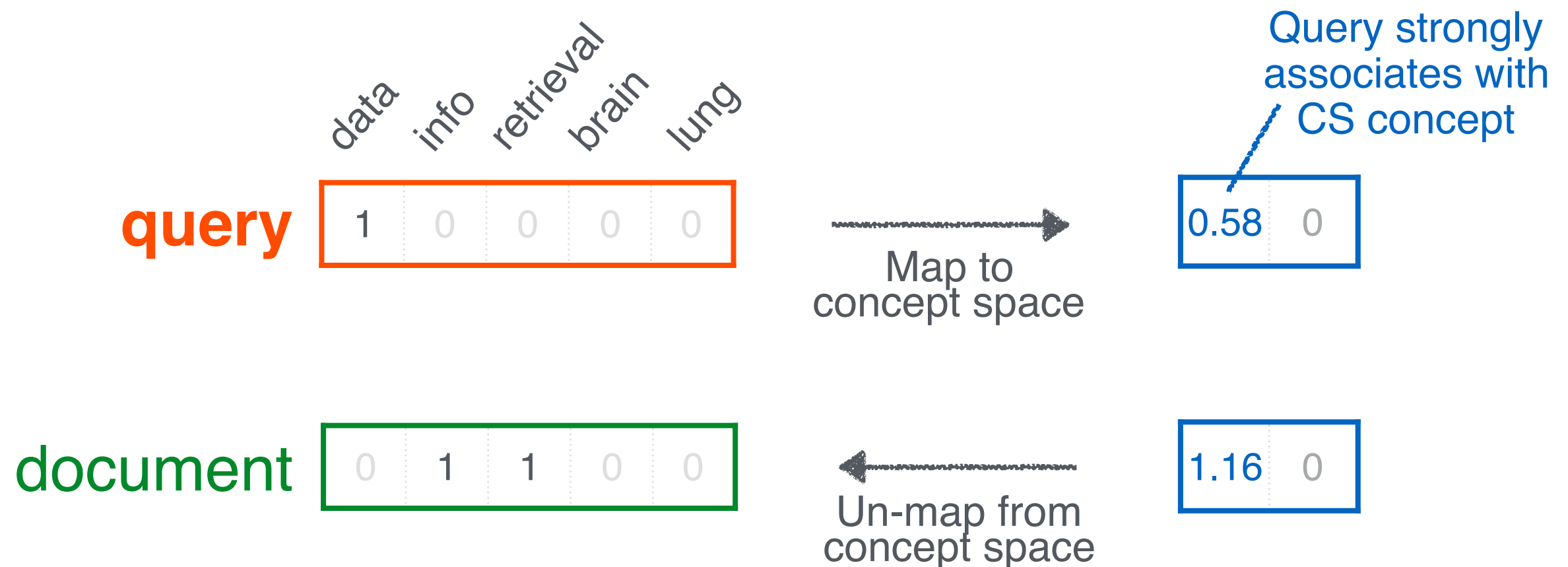
$$\mathbf{d} \mathbf{V} = \mathbf{d}_{\text{concept}}$$



Case Study

Observation

Document ('information', 'retrieval') will be retrieved by **query** ('data'), even though it does not contain 'data'!!



Case study - LSI

Q1: How to do queries with LSI?

➡ Q2: multi-lingual IR (english query, on spanish text?)

Case study - LSI

- Problem:
 - given many documents, translated to both languages (e.g., English and Spanish)
 - answer queries across languages

Case study - LSI

- Solution: \sim LSI

Diagram illustrating a neural network architecture for a classification task, showing input, hidden, and output layers with associated labels and connections.

Input Layer (Left):

- Nodes are arranged in two columns of five.
- Labels above the first column: **data**, **inf.**
- Labels above the second column: **retrieval**, **brain**
- Labels to the left: **CS** (Contextual Signal) and **MD** (Modulatory Drive)

Output Layer (Right):

- Nodes are arranged in two columns of five.
- Labels above the first column: **lung**, **datos**
- Labels above the second column: **informacion**, **datos**
- Labels to the right: **MD** (Modulatory Drive)

Connections:

- A vertical dashed line separates the input and output layers.
- Horizontal arrows indicate connections from the input layer to the output layer.
- Vertical arrows indicate connections from the **CS** and **MD** nodes to the input layer.
- A vertical arrow indicates a connection from the **MD** node to the output layer.

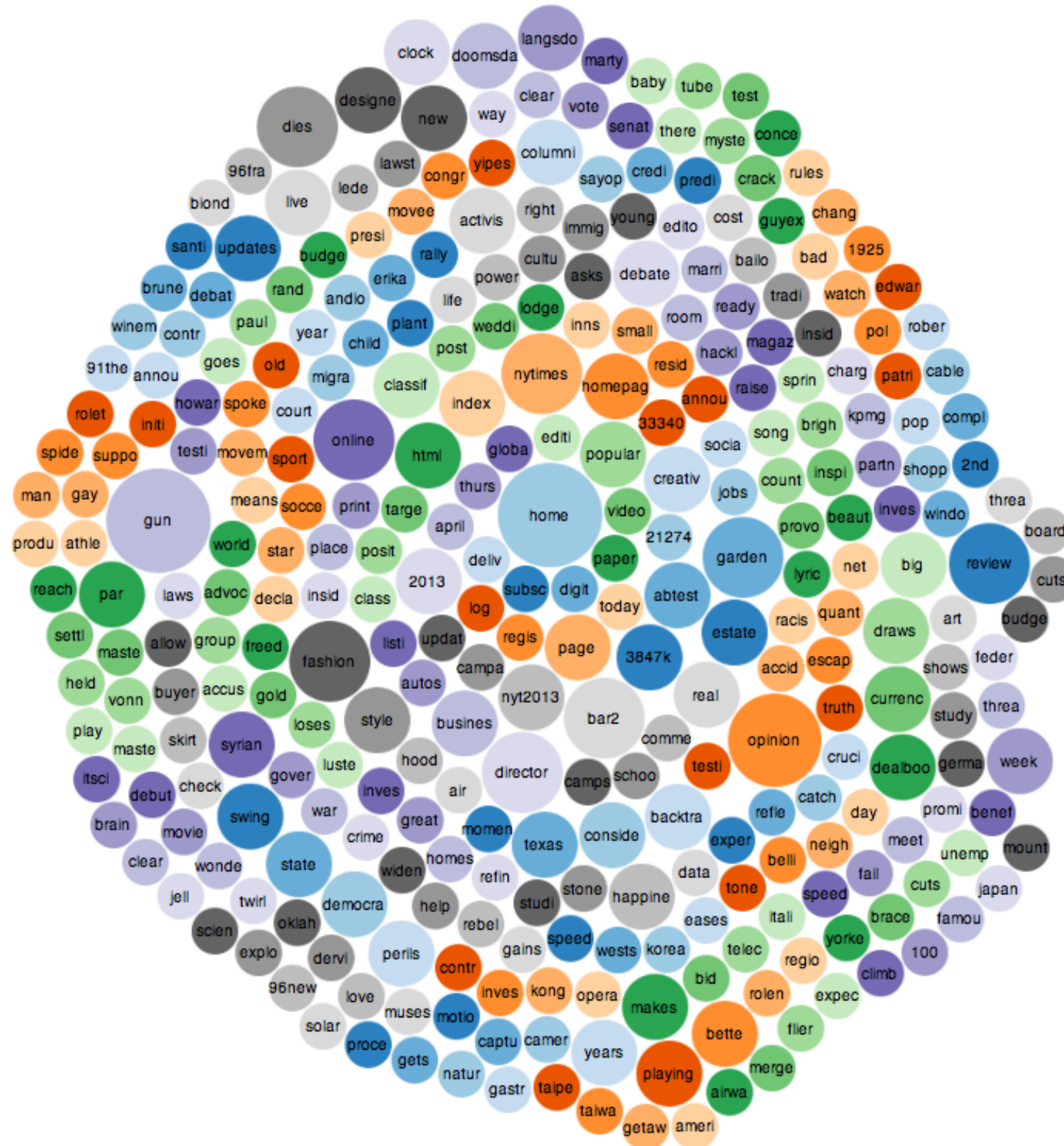
Switch Gear to **Text Visualization**

[illegible]

Twitter | [Tweets](#) | [Wikipedia](#) | [Custom](#)

Go!

Word Counts (words as bubbles)



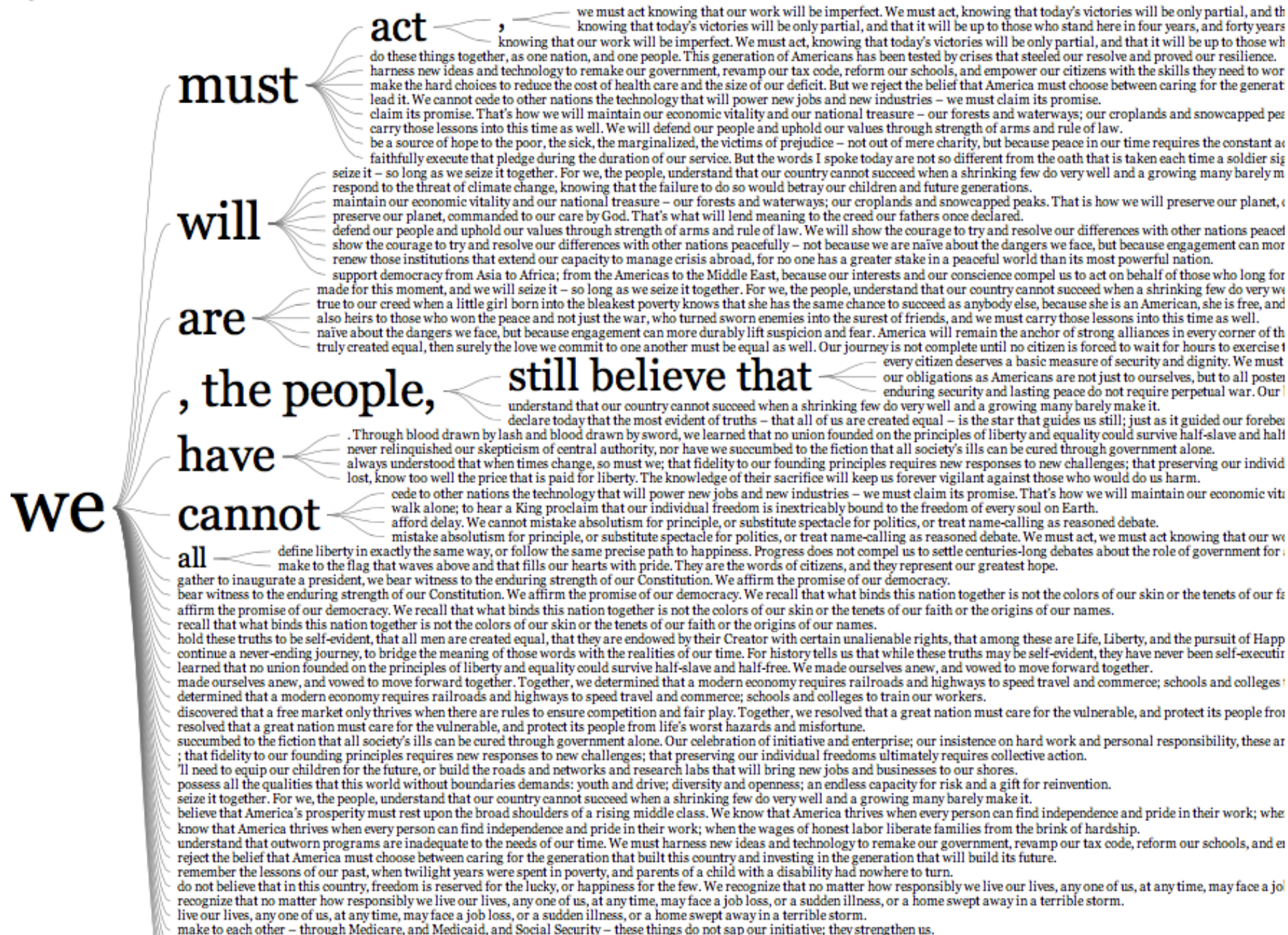
Word Tree

word tree

We

☐ reverse tree ☐ one phrase per line

Shift-click to make that word the root.



substitute spectacle for politics, or treat name-calling as reasoned debate. We must act, we must act knowing that our work will be imperfect. We must act, knowing that today's victories will be only partial, and that it will be up to those who stand here in four years, and forty years, and four hundred years hence to advance the timeless spirit once conferred to us in a spare Philadelphia hall.

My fellow Americans, the oath I have sworn before you today, like the one recited by others who serve in this Capitol, was an oath to God and country, not party or faction - and we must faithfully execute that pledge during the duration of our service. But the words I spoke today are not so different from the oath that is taken each time a soldier signs up for duty, or an immigrant realizes her dream. My oath is not so

Phrase Net

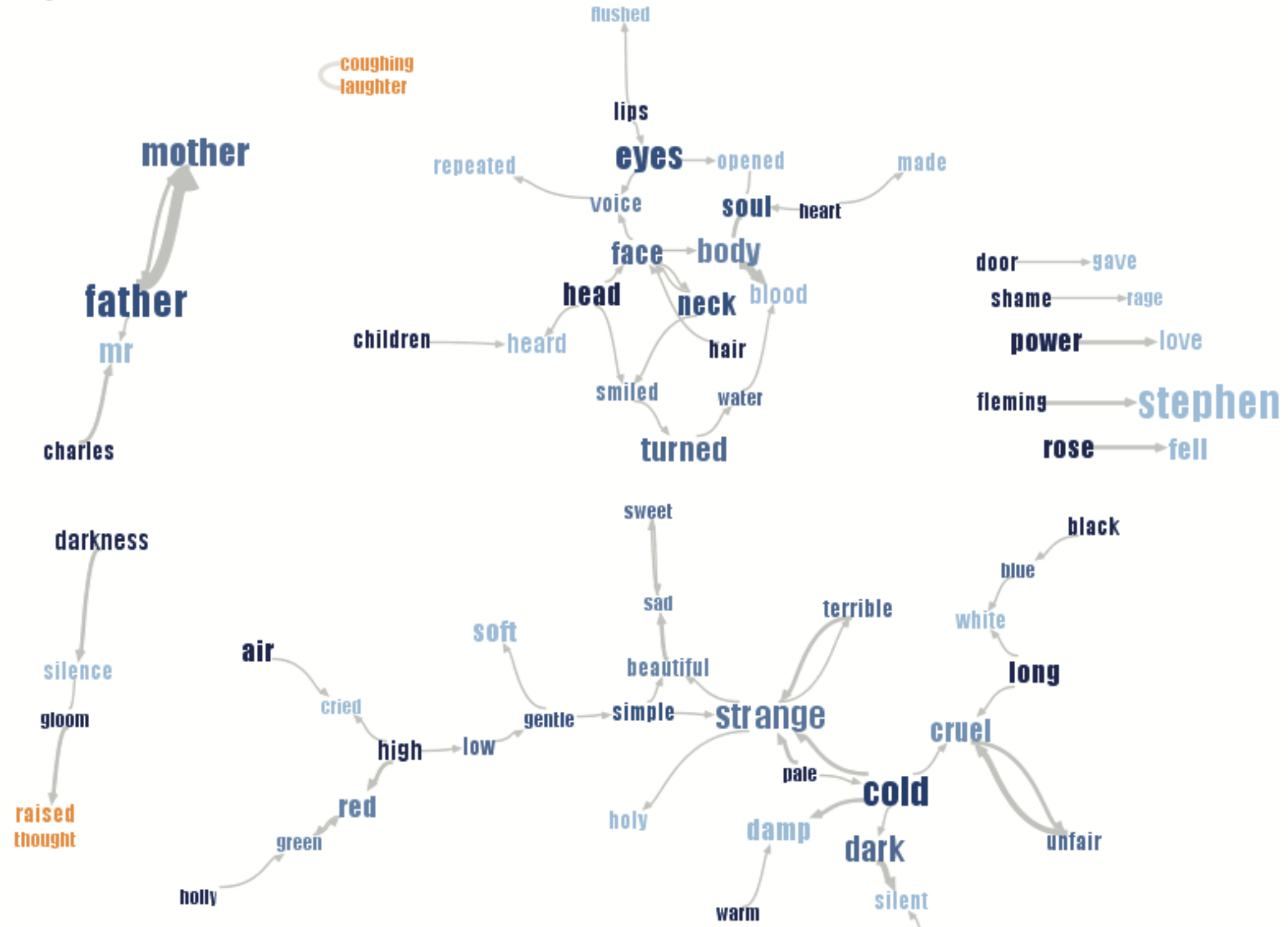
Visualize pairs of words satisfying a pattern (“X and Y”)

Select a phrase

word1	and	word2
word1	's	word2
word1	of the	word2
word1	the	word2
word1	a	word2
word1	at	word2
word1	is	word2
word1	[space]	word2

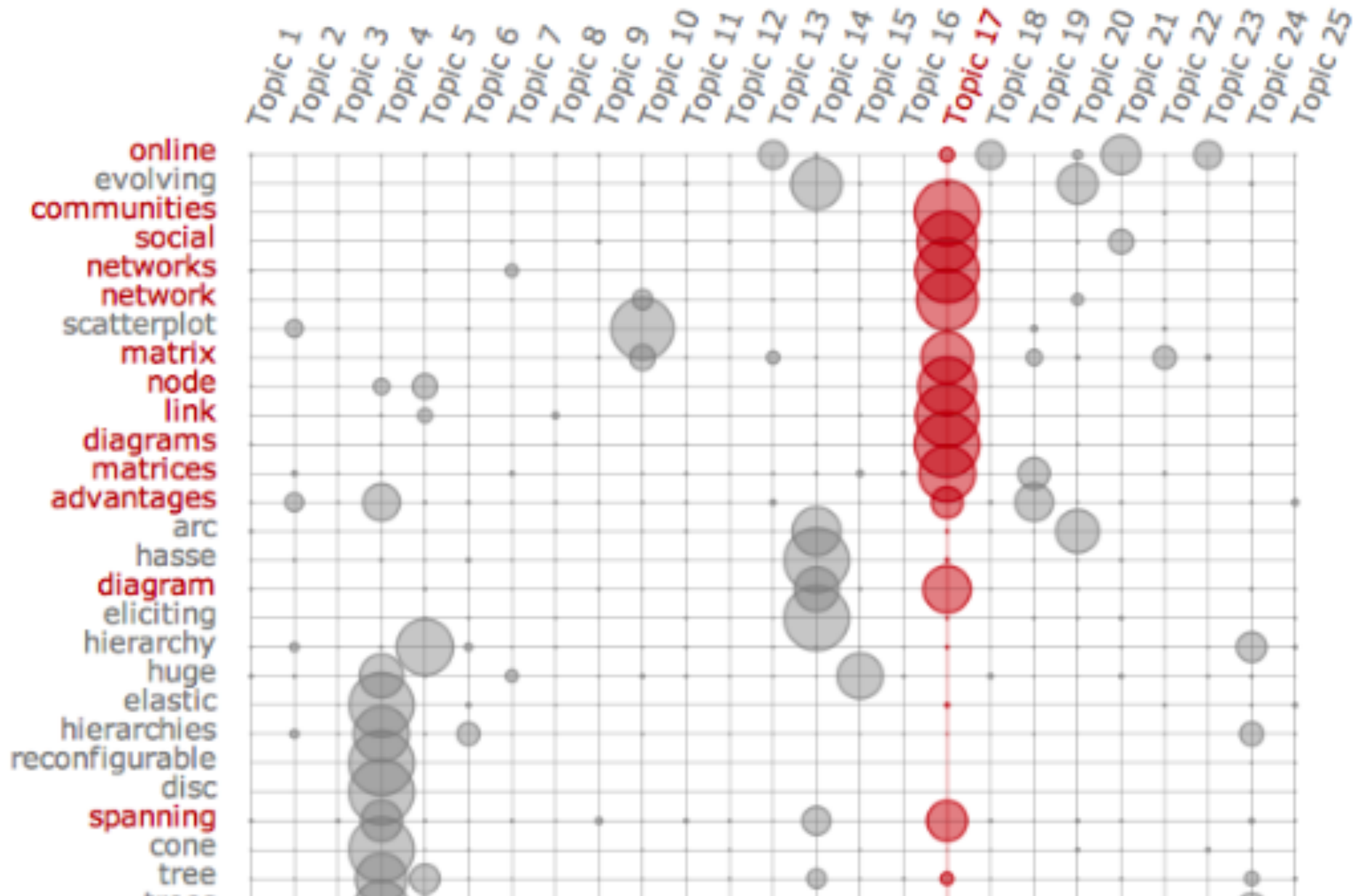
or enter your own

Showing 73 of 1719 terms



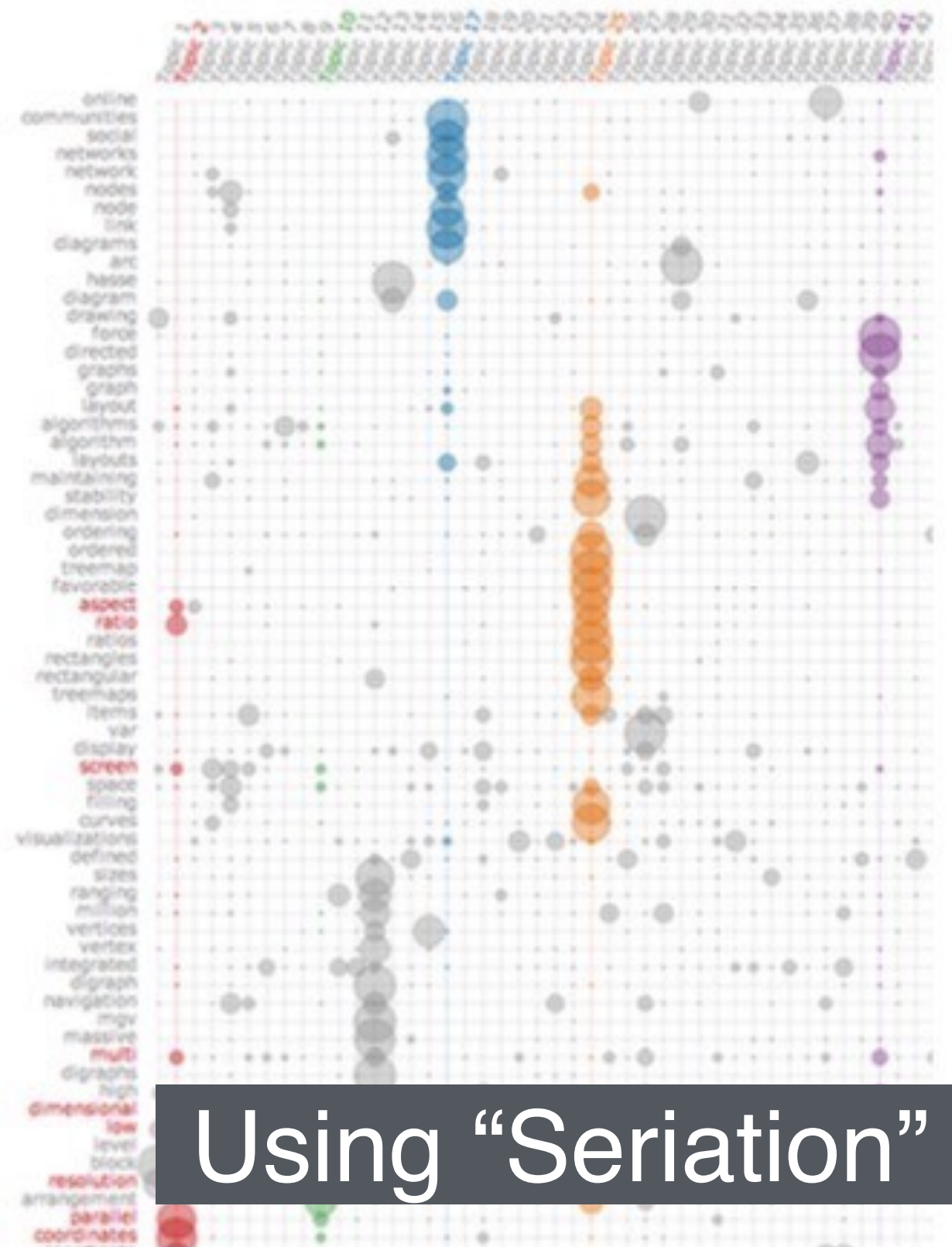
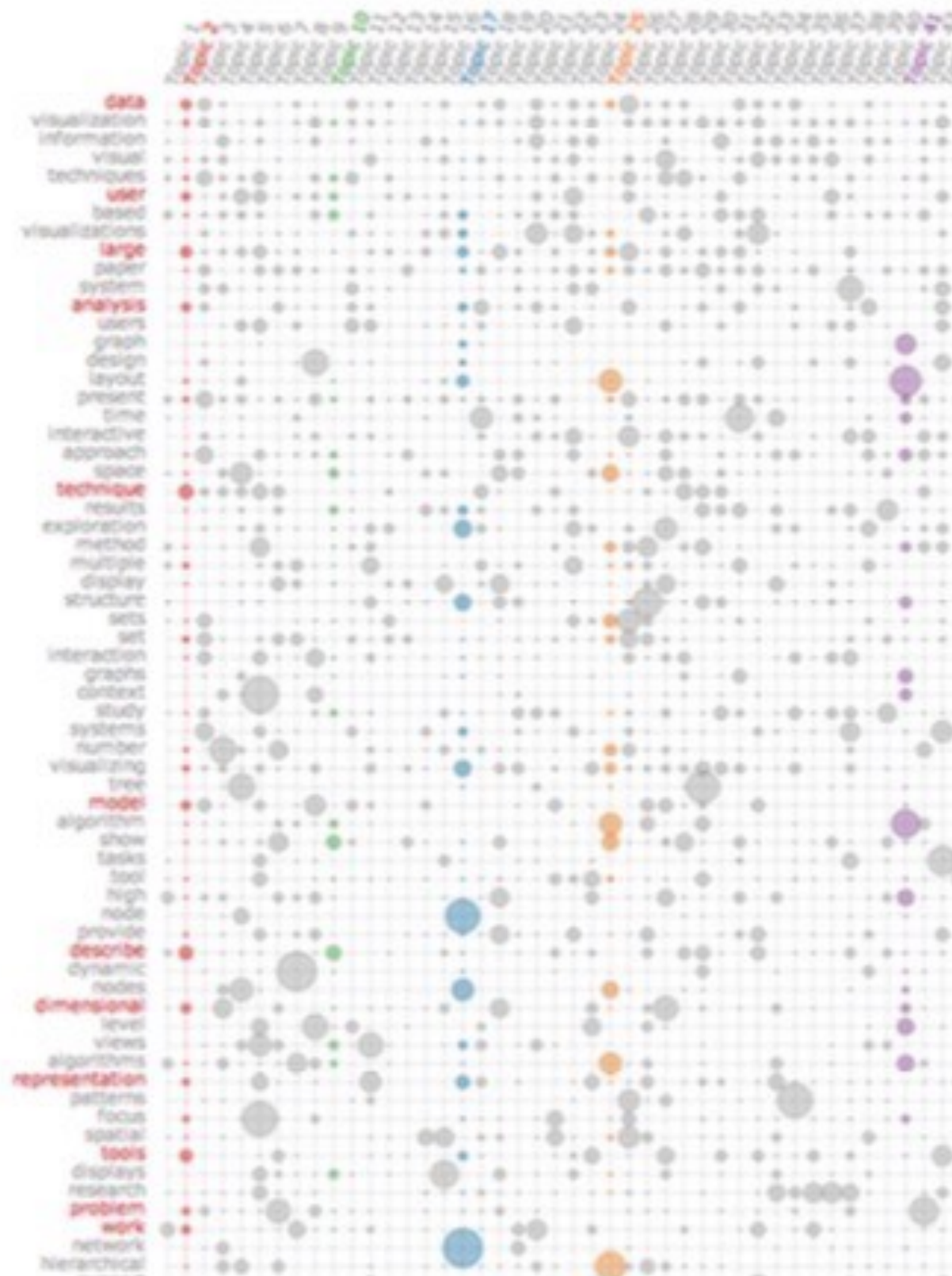
Termite: Topic Model Visualization

<http://vis.stanford.edu/papers/termite>



Termite: Topic Model Visualization

<http://vis.stanford.edu/papers/termite>



Using “Seriation”