http://poloclub.gatech.edu/cse6242

CSE6242 / CX4242: Data & Visual Analytics

Text Analytics (Text Mining)

Concepts, Algorithms, LSI/SVD

Duen Horng (Polo) Chau

Associate Professor

Associate Director, MS Analytics

Machine Learning Area Leader, College of Computing

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

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

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:
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" ➡ [1, 1, 0, 1]

"I like data" ➡ [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** / #document containing that **term**)
(penalize "common" words appearing in almost any documents)

Final score = TF * IDF
(higher score ➡ more "characteristic")

Example: http://en.wikipedia.org/wiki/Tf–idf#Example_of_tf.E2.80.93idf

Vector Space Model

Why?

Each document ➡ vector

Each query ➡ vector

Search for documents ➡ find “similar” vectors

Cluster documents ➡ 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

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

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

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.

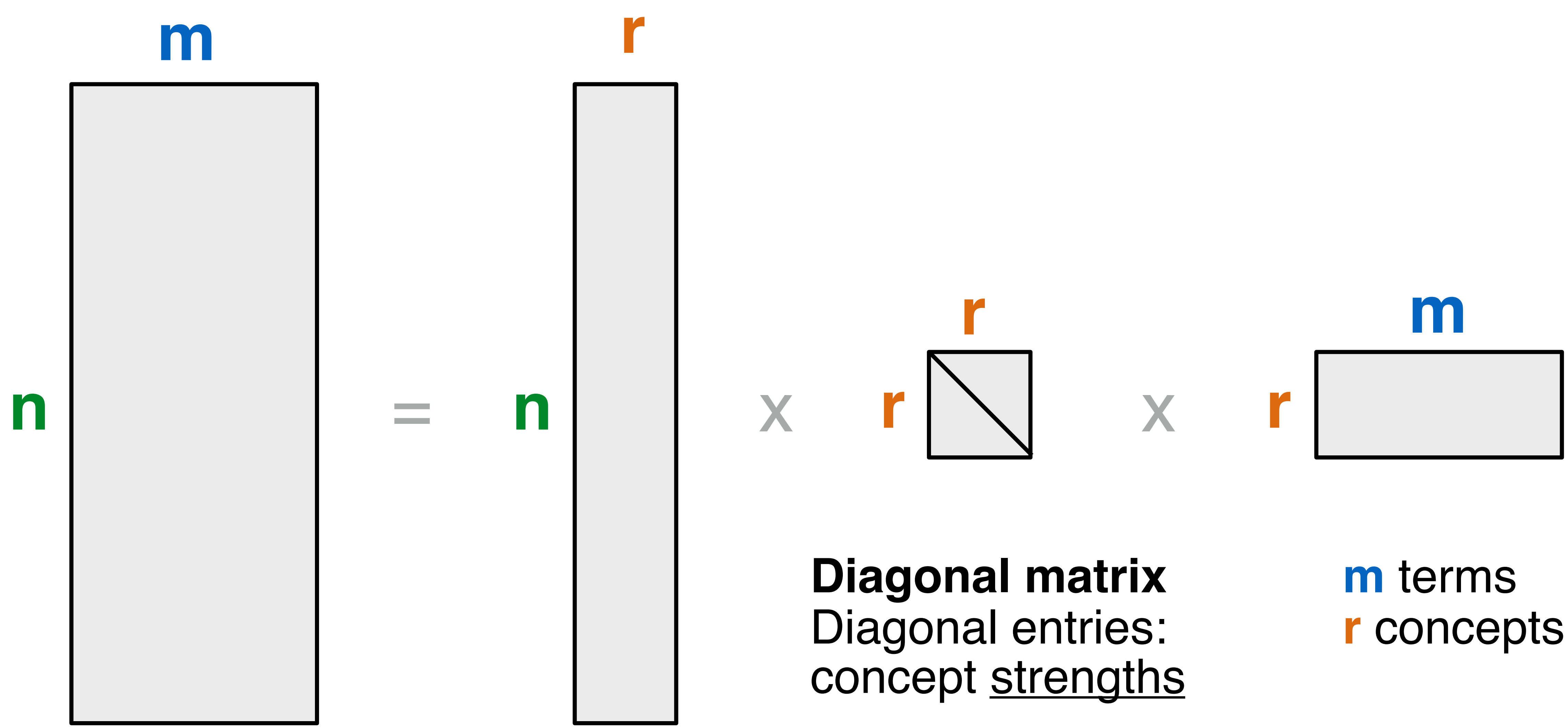
Songs / Movies / Products

Customers

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

SVD Definition (pictorially)

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



n documents
m terms

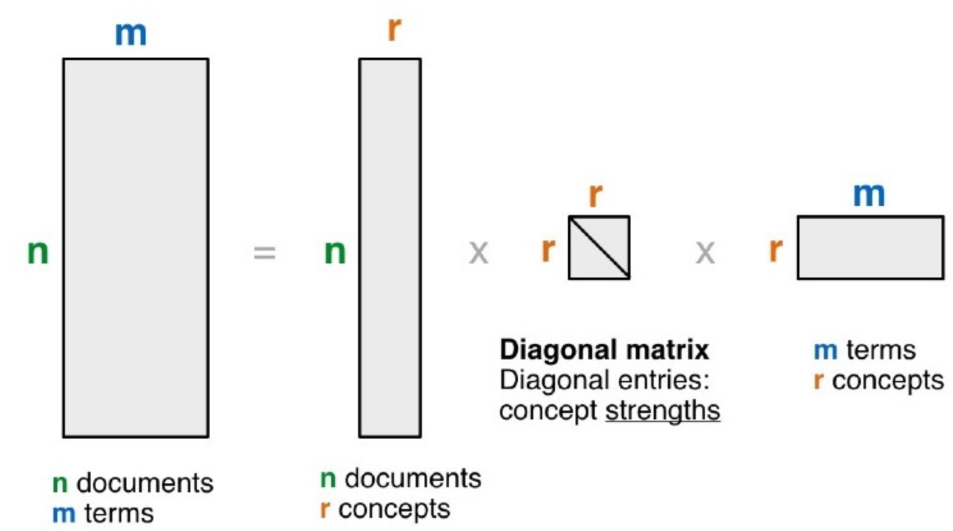
n documents
r concepts

Diagonal matrix
Diagonal entries:
concept strengths

m terms
r concepts

SVD Definition (in words)

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



A: n x m matrix

e.g., n documents, m terms

U: n x r matrix

e.g., n documents, r concepts

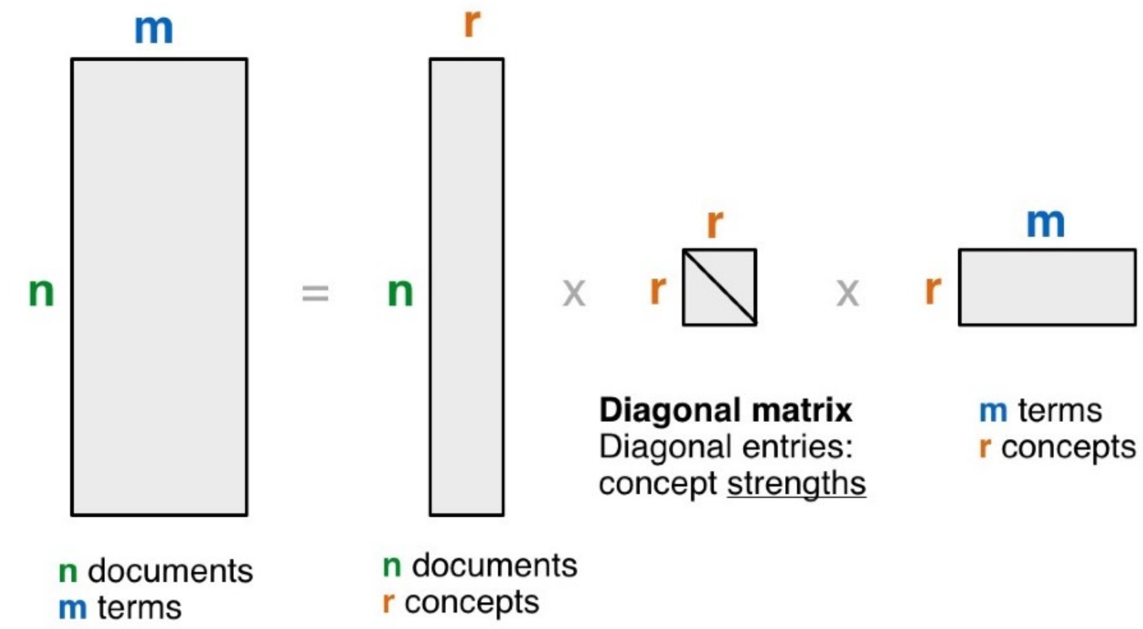
Λ: r x r diagonal matrix

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

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

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

U, **V**: column **orthonormal**

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

$\mathbf{U}^T \mathbf{U} = \mathbf{I}$

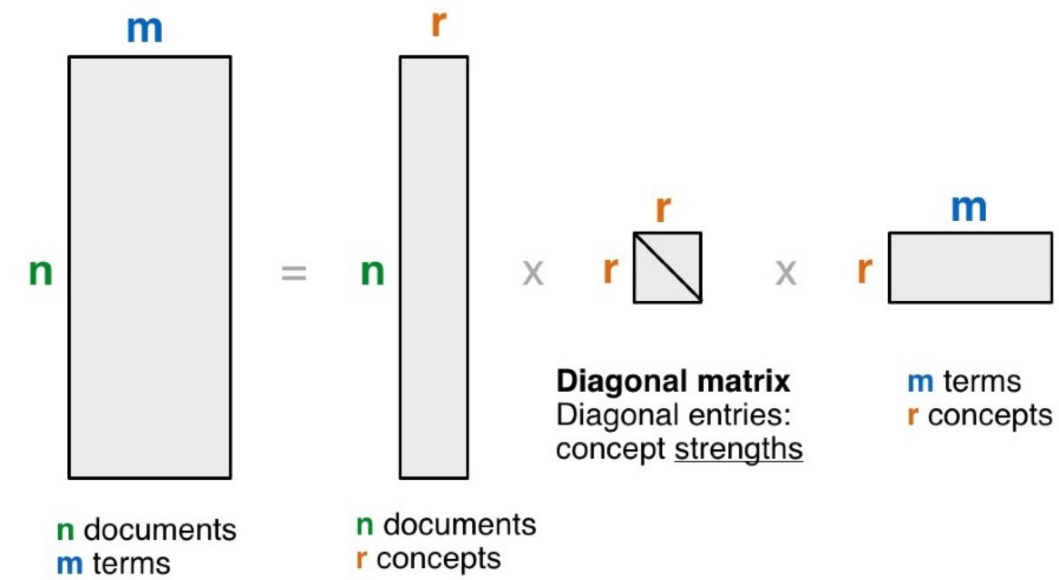
$\mathbf{V}^T \mathbf{V} = \mathbf{I}$

(**I**: identity matrix)

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

SVD - Example

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



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

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

=

CS concept	MD concept
0.18	0
0.36	0
0.18	0
0.90	0
0	0.53
0	0.80
0	0.27

document-concept
similarity matrix

x

“strength” of CS-concept

9.64	0
0	5.29

x

	data	info	retrieval	brain	lung
CS concept	0.58	0.58	0.58	0	0
MD concept	0	0	0	0.71	0.71

term-concept
similarity matrix

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 **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 $\mathbf{A}^T \mathbf{A}$?

A: term-to-term ([m x m]) similarity matrix

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

A: document-to-document ([n x n]) similarity matrix

SVD properties

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

$$\mathbf{A}^\mathsf{T}\mathbf{A} = \left(\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\mathsf{T}\right)^\mathsf{T}\left(\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\mathsf{T}\right) = \mathbf{V}\boldsymbol{\Sigma}^2\mathbf{V}^\mathsf{T}$$

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

$$\mathbf{A}\mathbf{A}^\mathsf{T} = \left(\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\mathsf{T}\right)\left(\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\mathsf{T}\right)^\mathsf{T} = \mathbf{U}\boldsymbol{\Sigma}^2\mathbf{U}^\mathsf{T}$$

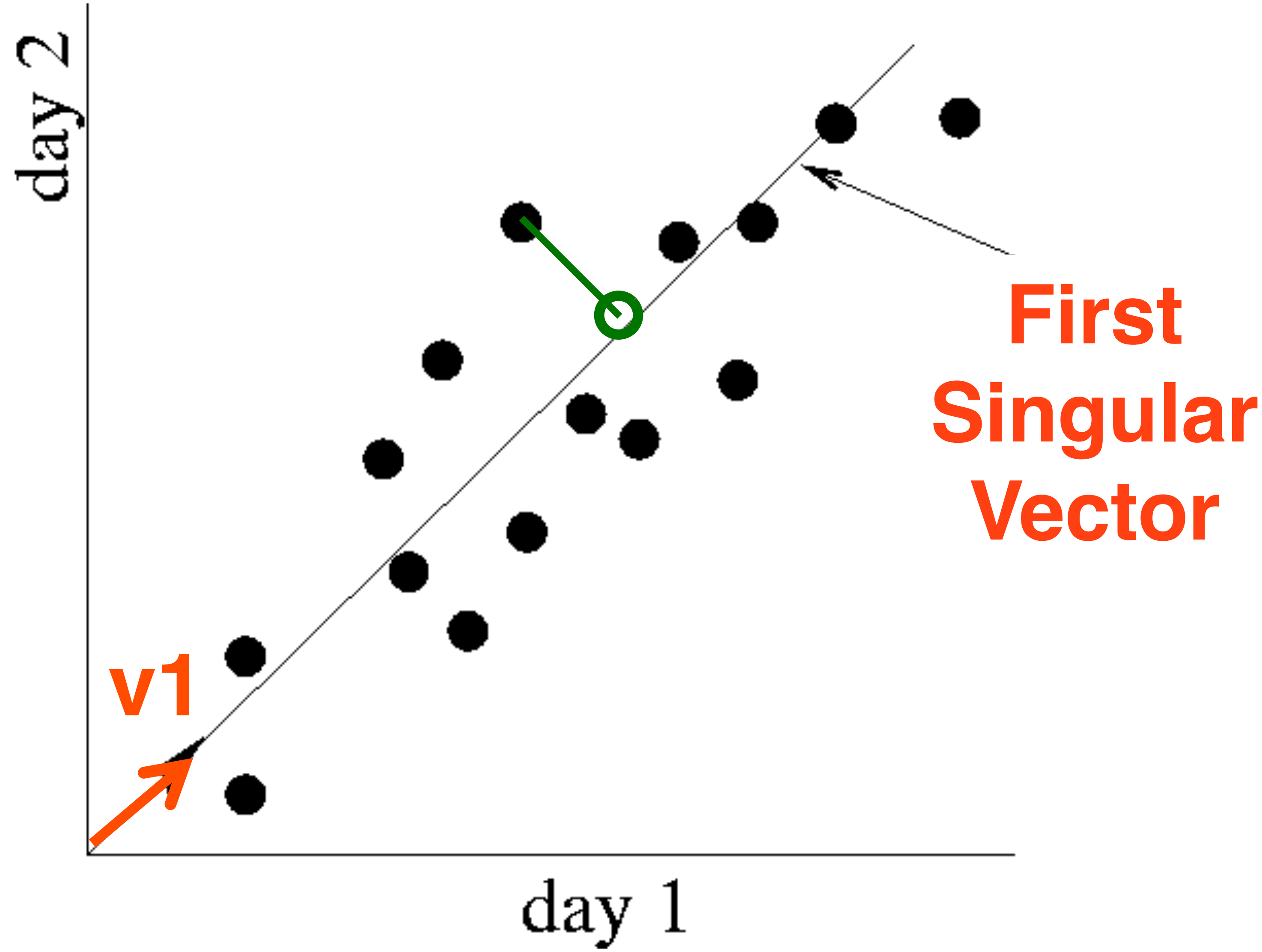
SVD is closely related to PCA, and can be numerically more stable.
For more info, see:

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.

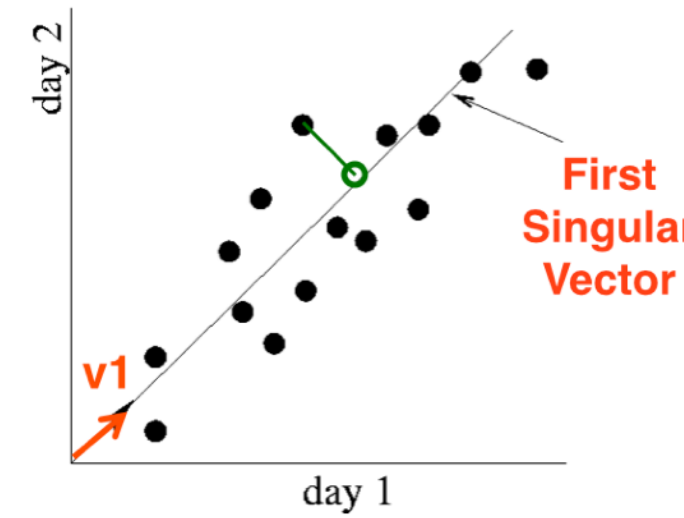
(‘best’ = min sum of squares of projection errors)



min RMS error

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

SVD - Interpretation #2



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

variance (‘spread’) on the v1 axis

9.64	0
0	5.29

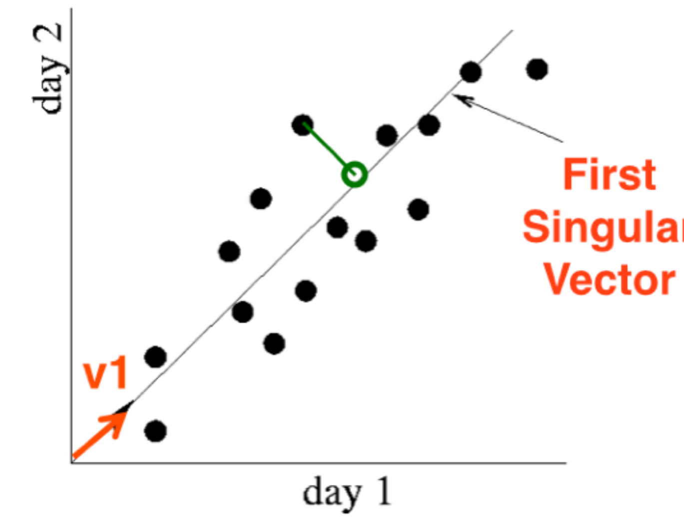
x

v1

0.58	0.58	0.58	0	0
0	0	0	0.71	0.71

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

SVD - Interpretation #2



U Λ gives the **coordinates** of the points in the projection axis

A =

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

U =

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

x Λ =

9.64	0
0	5.29

x V^T =

v1

0.58	0.58	0.58	0	0
0	0	0	0.71	0.71

A = U Λ V^T

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:

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

SVD - Interpretation #2

More details

Q: how exactly is dim. reduction done?

A: set the smallest singular values to zero:

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

SVD - Interpretation #2

More details

Q: how exactly is dim. reduction done?

A: set the smallest singular values to zero:

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

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

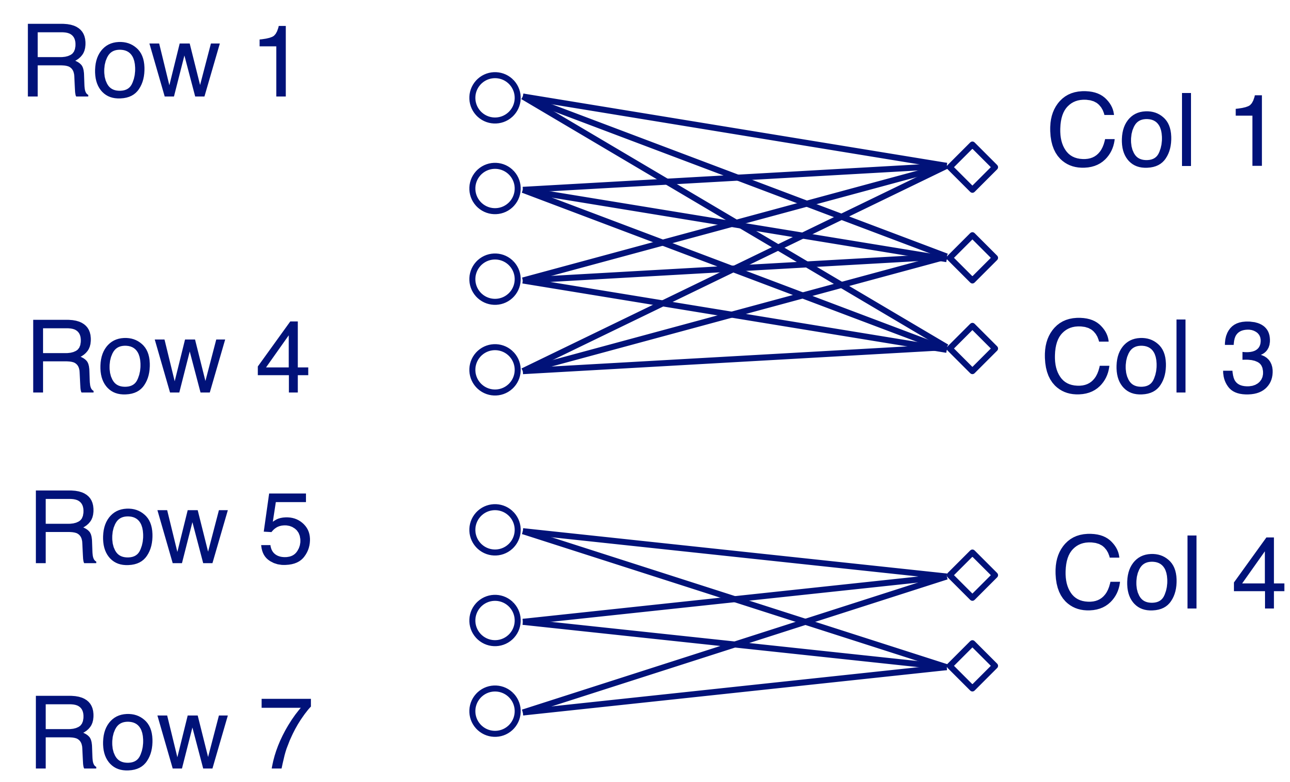
finds non-zero ‘blobs’ in a data matrix

$$
\left[\begin{array}{ccc|cc}
1 & 1 & 1 & 0 & 0 \\
2 & 2 & 2 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
\hline
0 & 0 & 0 & 2 & 2 \\
0 & 0 & 0 & 3 & 3 \\
0 & 0 & 0 & 1 & 1
\end{array}\right]
=
\begin{bmatrix}
0.18 & 0 \\
0.36 & 0 \\
0.18 & 0 \\
0.90 & 0 \\
0 & 0.53 \\
0 & 0.80 \\
0 & 0.27
\end{bmatrix}
\times
\begin{bmatrix}
9.64 & 0 \\
0 & 5.29
\end{bmatrix}
\times
\begin{bmatrix}
0.58 & 0.58 & 0.58 & 0 & 0 \\
0 & 0 & 0 & 0.71 & 0.71
\end{bmatrix}
$$

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



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’?

	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

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

Case Study

How to do queries with LSI?

For example, how to find documents with ‘data’?
A: map query vectors into ‘concept space’ – how?

	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

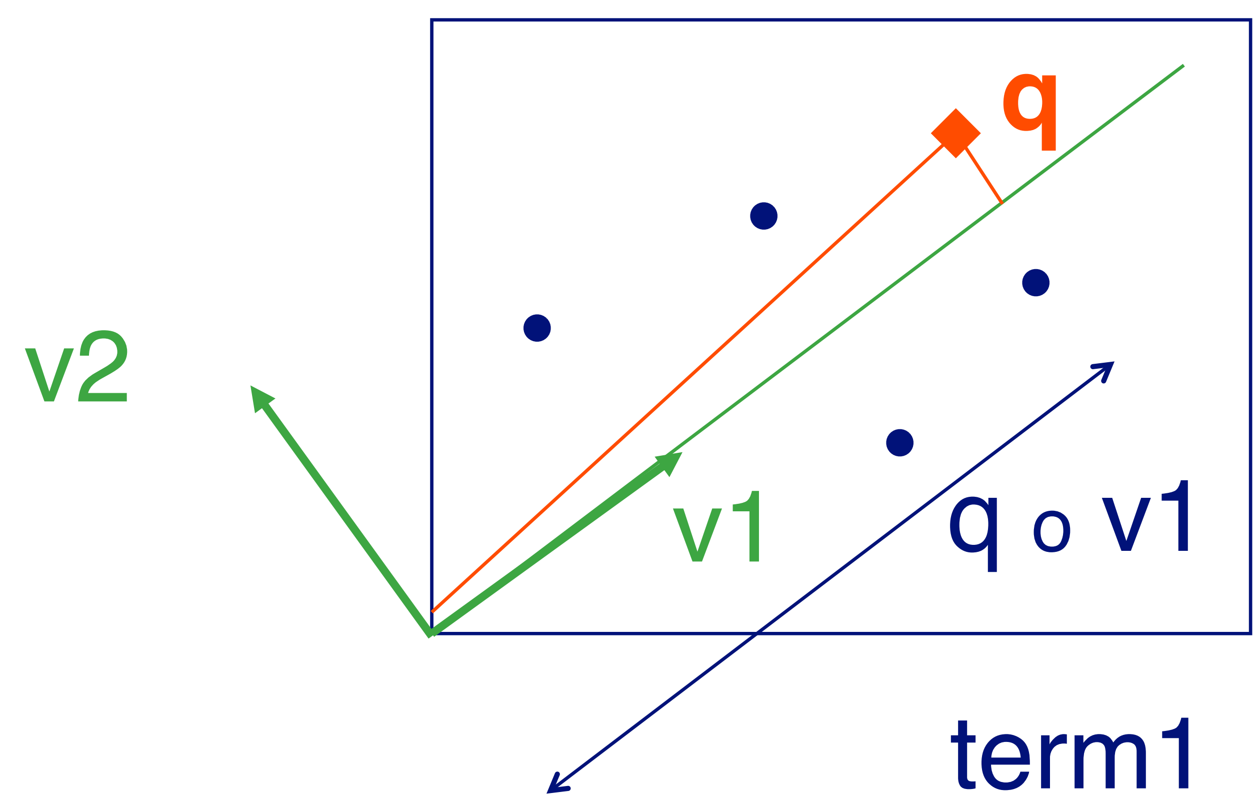
Case Study

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**

q =

data	info	retrieval	brain	lung
1	0	0	0	0

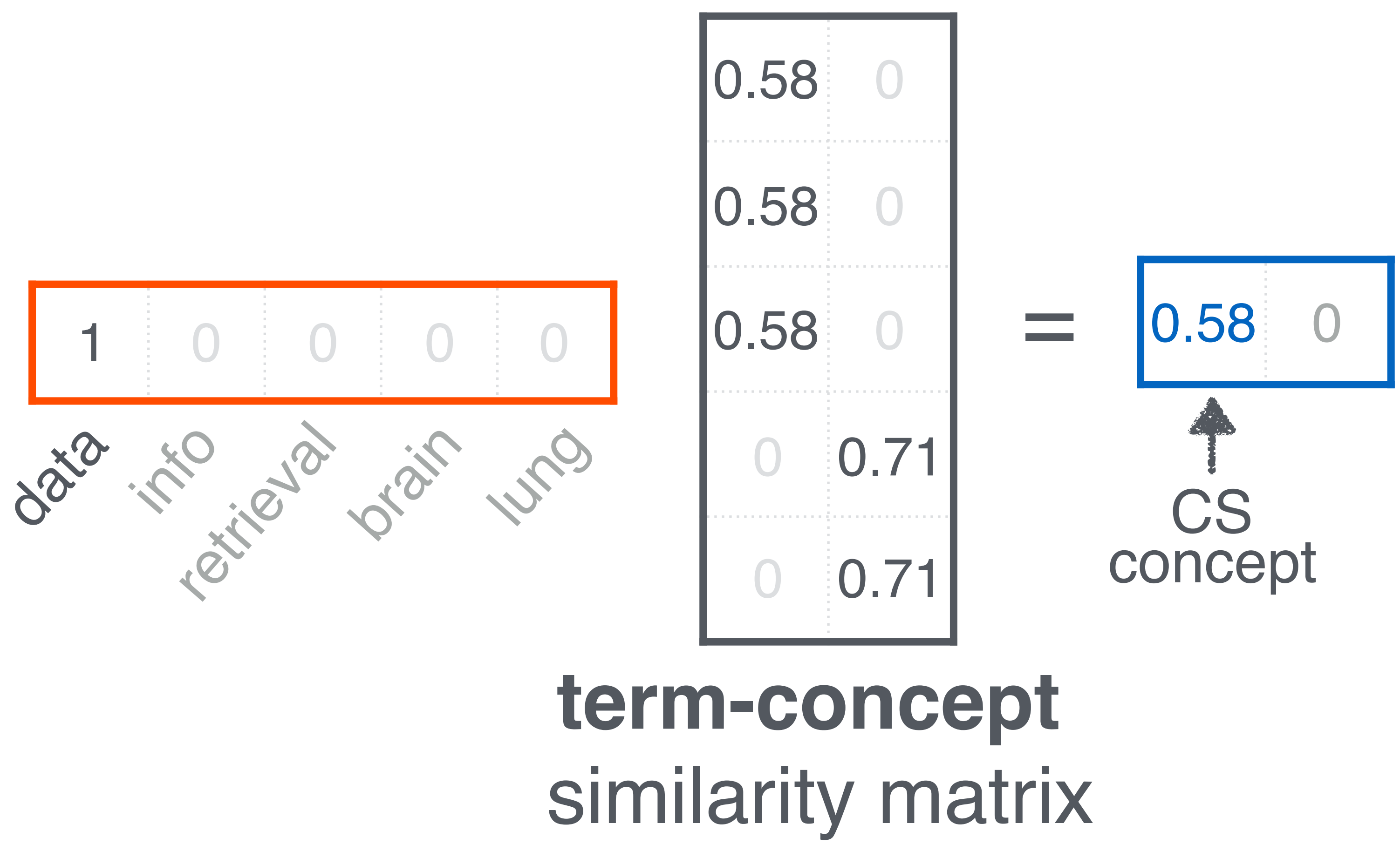


Case Study

How to do queries with LSI?

Compactly, we have:

$$\mathbf{q}\ \mathbf{V} = q_{concept}$$



Case Study

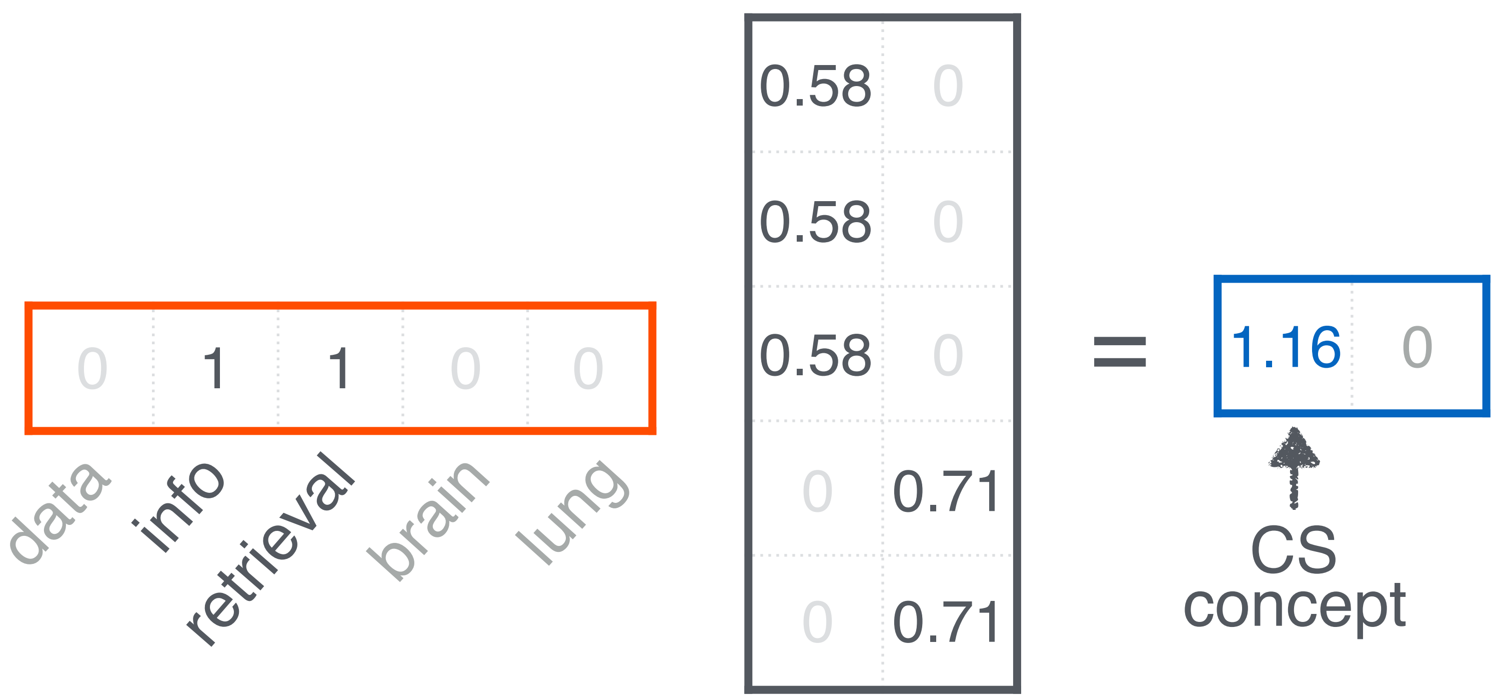
How would the document (‘information’, ‘retrieval’) be handled?

Case Study

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

SAME!

$$\mathbf{d}\,\mathbf{V} = d_{concept}$$

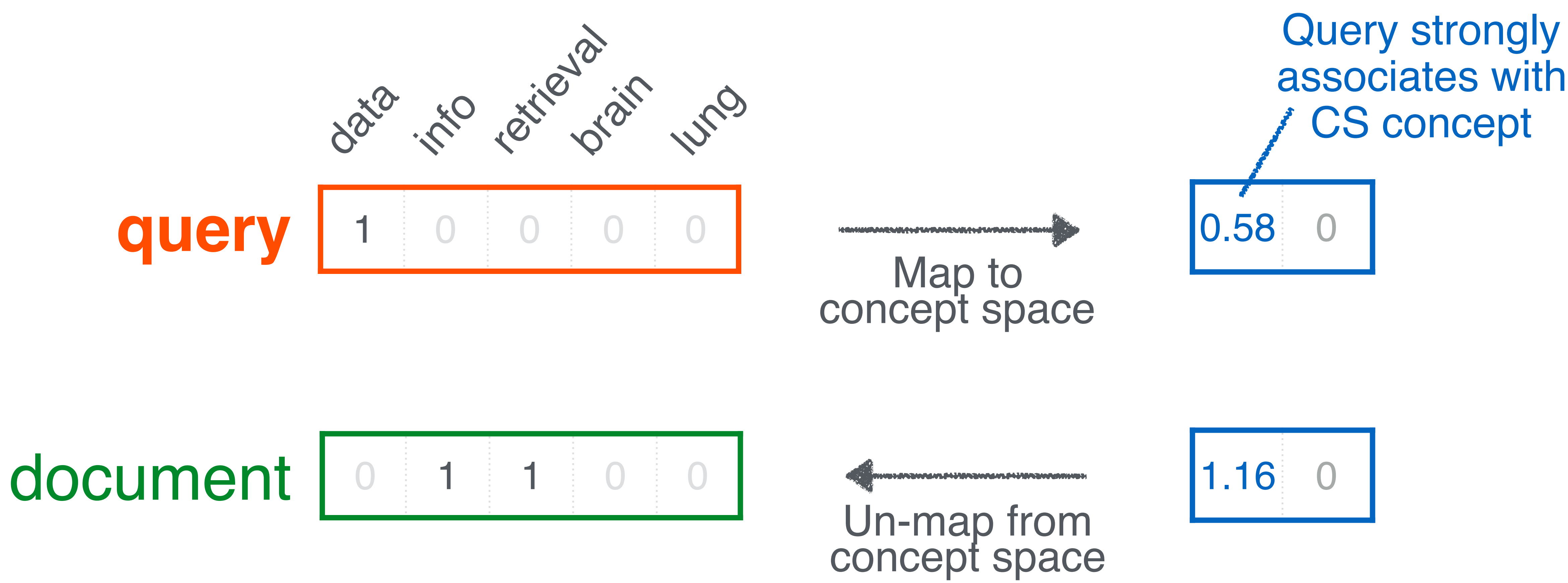


term-concept
similarity matrix

Case Study

Observation

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

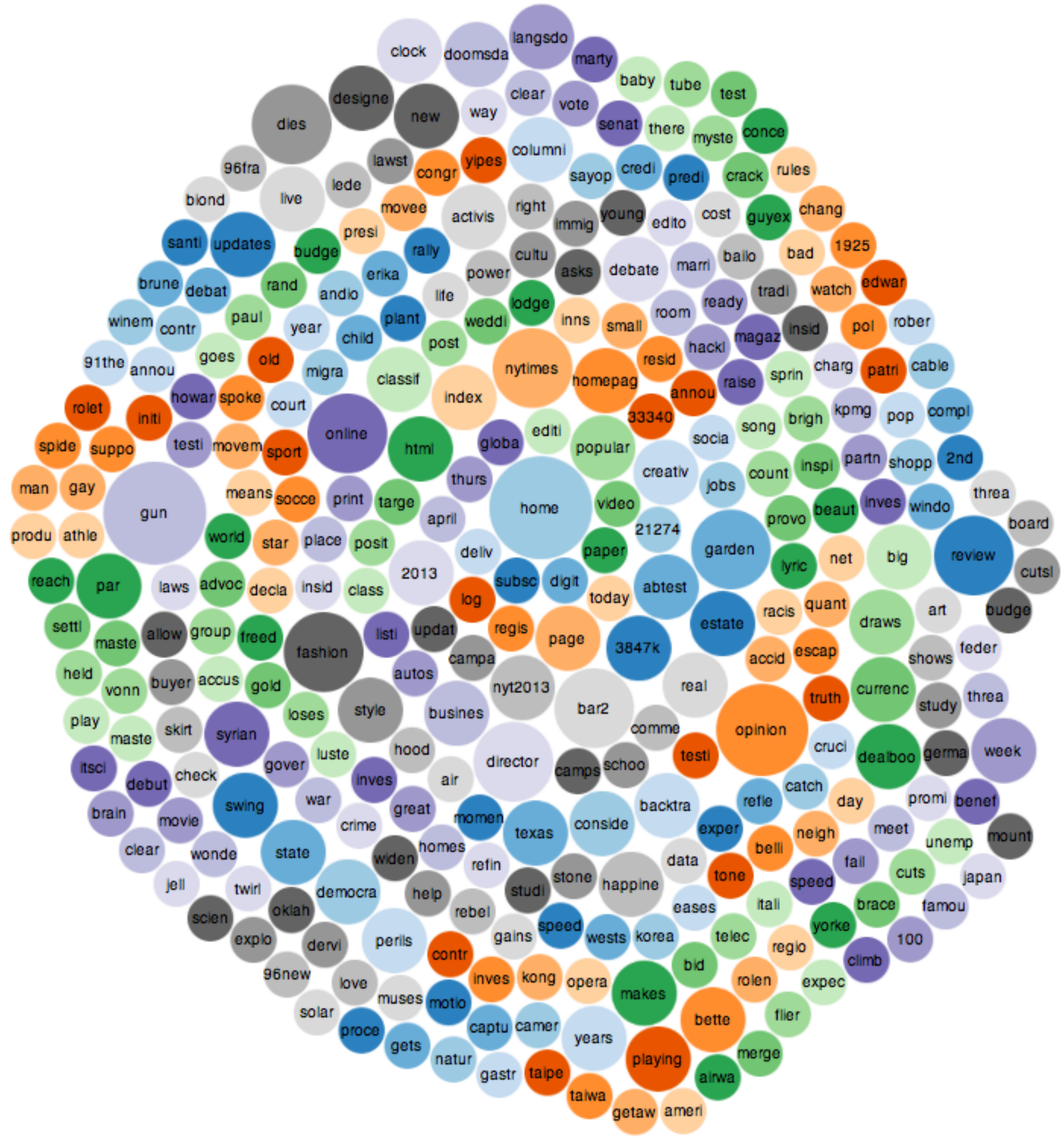


Switch Gear to **Text Visualization**

Word/Tag Cloud (still popular?)

http://www.wordle.net

Word Counts (words as bubbles)

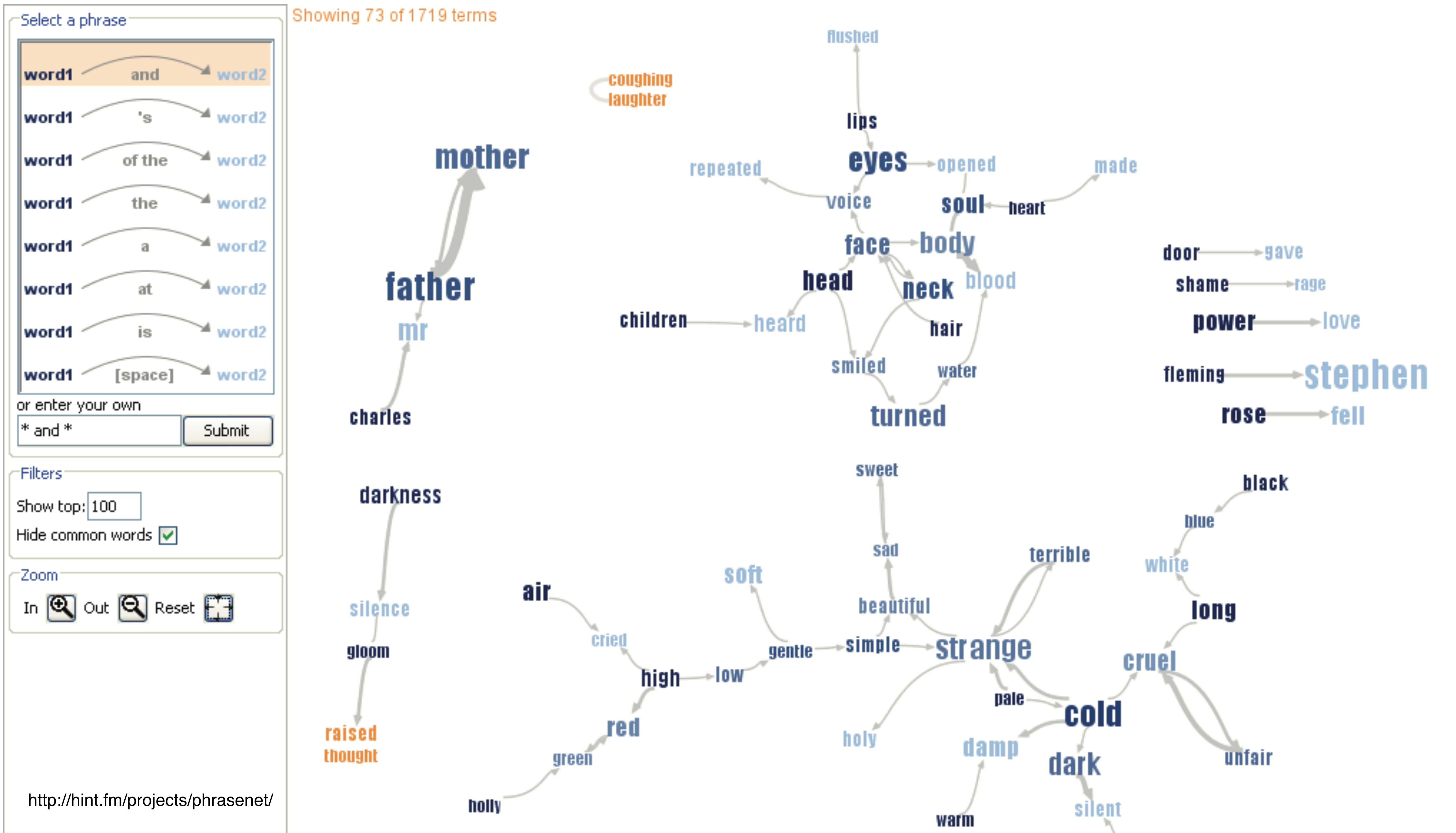


Word Tree

http://www.jasondavies.com/wordtree/

Phrase Net

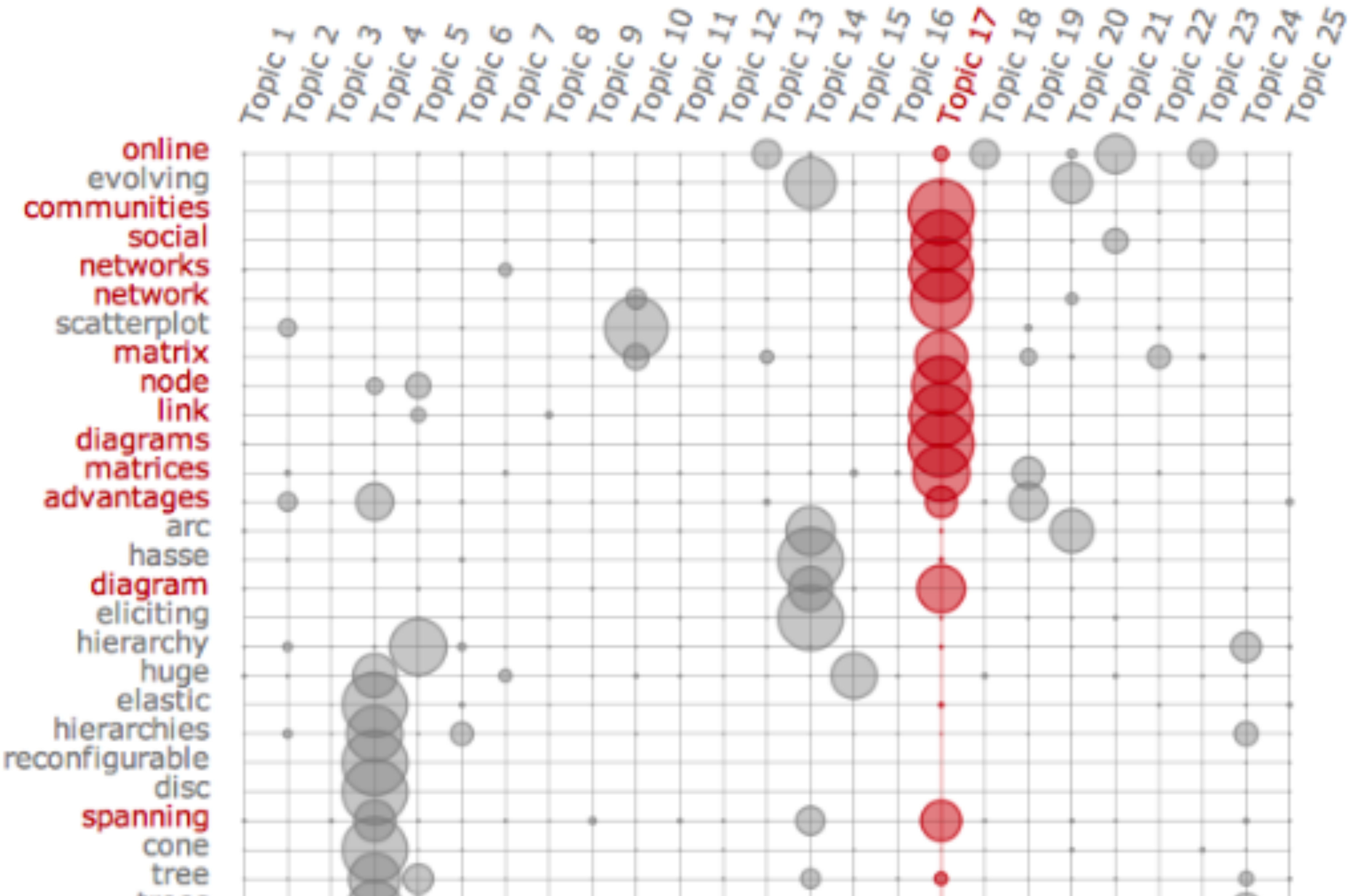
Visualize pairs of words satisfying a pattern (“X and Y”)



http://hint.fm/projects/phrasenet/

Termite: Topic Model Visualization

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



Termite: Topic Model Visualization

Analy

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