INF 141
IR METRICS
LATENT SEMANTIC ANALYSIS AND INDEXING

Crista Lopes

Outline

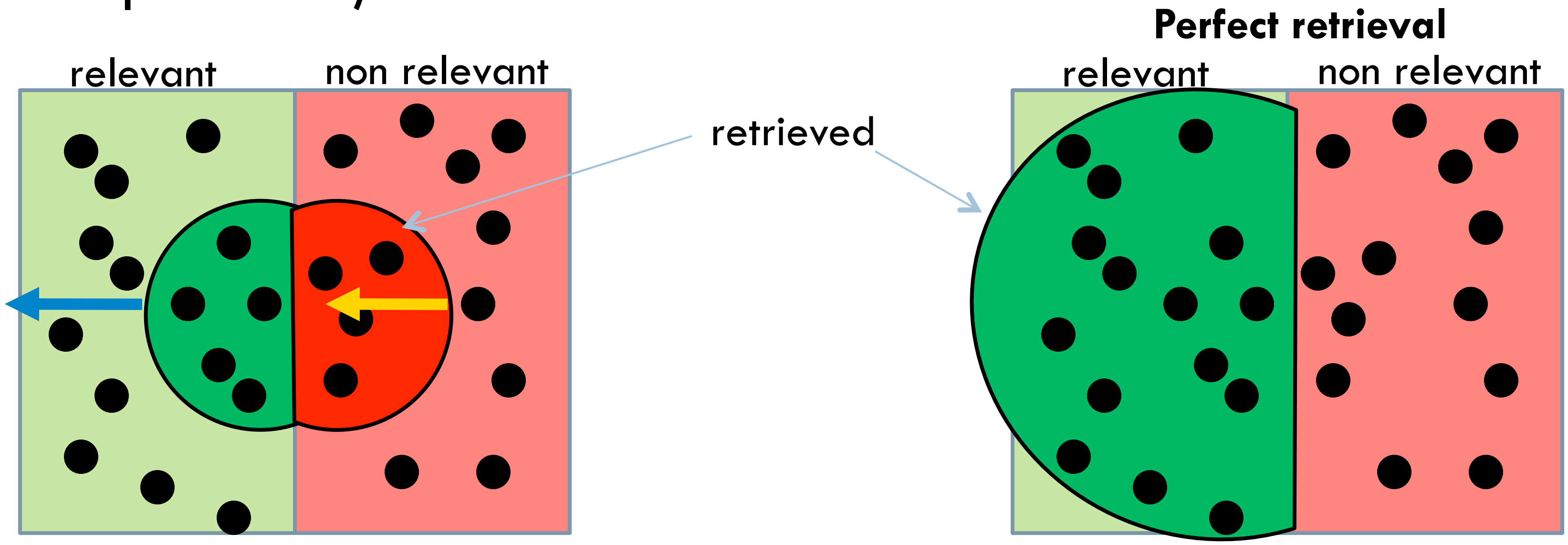
- Precision and Recall
- The problem with indexing so far
- Intuition for solving it
- Overview of the solution
- The Math

How to measure

- Given the enormous variety of possible retrieval schemes, how do we measure how good they are?

Standard IR Metrics

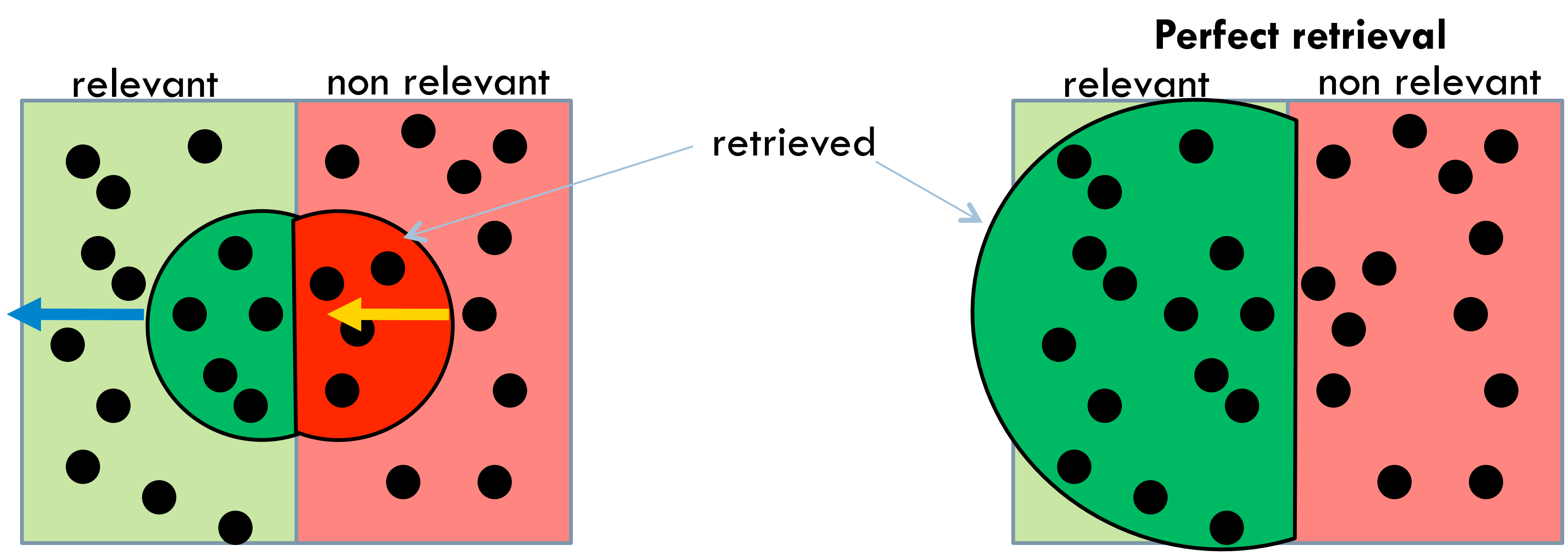
- **Recall:** portion of the relevant documents that the system retrieved (blue arrow points in the direction of higher recall)
- **Precision:** portion of retrieved documents that are relevant (yellow arrow points in the direction of higher precision)



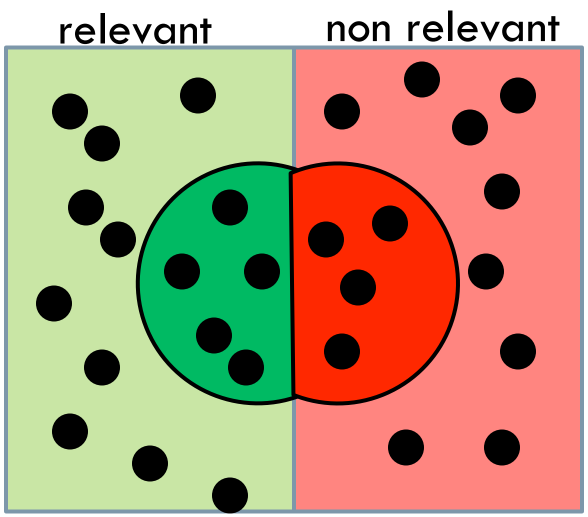
Definitions

$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}$$



Definitions



True positives

False negatives

True negatives

False positives

$$\text{Precision} = \frac{tp}{tp + fp} \qquad \text{Recall} = \frac{tp}{tp + fn}$$

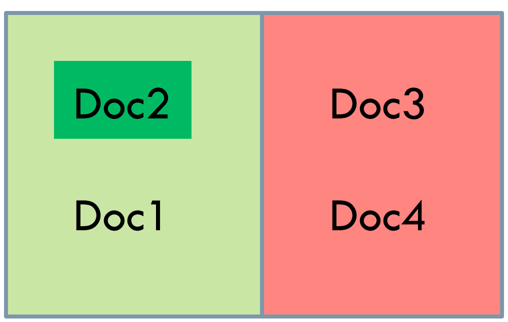
(same thing, different terminology)

Example

Doc1 = A comparison of the newest models of cars (keyword: car)
Doc2 = Guidelines for automobile manufacturing (keyword: automobile)
Doc3 = The car function in Lisp (keyword: car)
Doc4 = Flora in North America

Query: “automobile”

Retrieval scheme A



$$\text{Precision} = \frac{tp}{tp + fp}$$

Precision = 1/1 = 1

$$\text{Recall} = \frac{tp}{tp + fn}$$

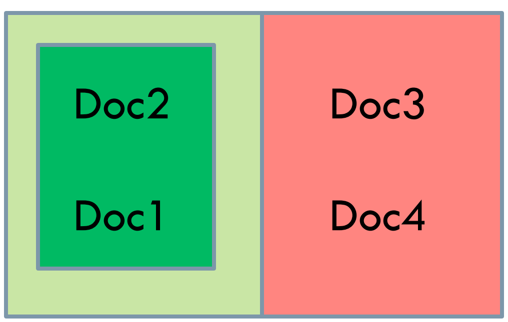
Recall = 1/2 = 0.5

Example

Doc1 = A comparison of the newest models of cars (keyword: car)
Doc2 = Guidelines for automobile manufacturing (keyword: automobile)
Doc3 = The car function in Lisp (keyword: car)
Doc4 = Flora in North America

Query: “automobile”

Retrieval scheme B



$$\text{Precision} = \frac{tp}{tp + fp}$$

Precision = 2/2 = 1

$$\text{Recall} = \frac{tp}{tp + fn}$$

Recall = 2/2 = 1

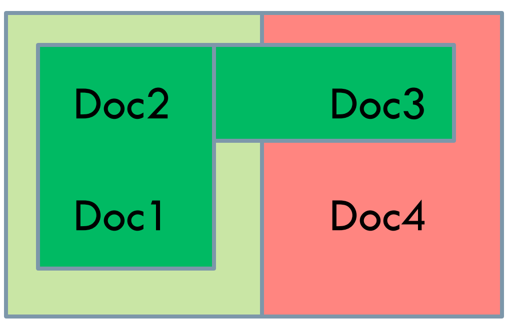
Perfect!

Example

Doc1 = A comparison of the newest models of cars (keyword: car)
Doc2 = Guidelines for automobile manufacturing (keyword: automobile)
Doc3 = The car function in Lisp (keyword: car)
Doc4 = Flora in North America

Query: "automobile"

Retrieval scheme C



$$\text{Precision} = \frac{tp}{tp + fp}$$

Precision = 2/3 = 0.67

$$\text{Recall} = \frac{tp}{tp + fn}$$

Recall = 2/2 = 1

Example

- Clearly scheme B is the best of the 3.
- A vs. C: which one is better?
  - Depends on what you are trying to achieve

- Intuitively for people:
  - Low precision leads to low trust in the system – too much noise!
    (e.g. consider precision = 0.1)
  - Low recall leads to unawareness
    (e.g. consider recall = 0.1)

F-measure

- Combines precision and recall into a single number

$$F = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}$$

More generally,

$$F_\beta = (1 + \beta^2) \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\beta^2 \cdot \mathrm{precision} + \mathrm{recall}}$$

Typical values:

$\beta = 2$ ➔ gives more weight to recall

$\beta = 0.5$ ➔ gives more weight to precision

F-measure

$$F(\text{scheme A}) = 2 * (1 * 0.5)/(1+0.5) = 0.67$$

$$F(\text{scheme B}) = 2 * (1 * 1)/(1+1) = 1$$

$$F(\text{scheme C}) = 2 * (0.67 * 1)/(0.67+1) = 0.8$$

Test Data

- In order to get these numbers, we need data sets for which we know the relevant and non-relevant documents for test queries
  - Requires human judgment

Outline

- The problem with indexing so far
- Intuition for solving it
- Overview of the solution
- The Math

Part of these notes were adapted from:

[1] An Introduction to Latent Semantic Analysis, Melanie Martin
http://www.slidefinder.net/I/Introduction_Latent_Semantic_Analysis_Melanie/26158812

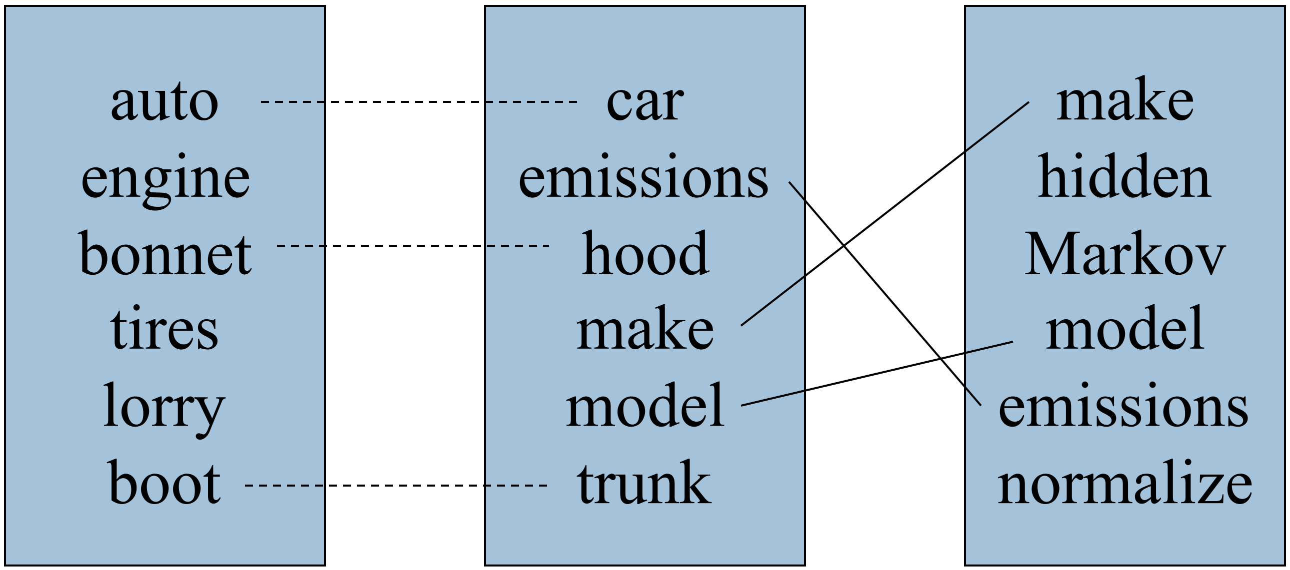
Indexing so far

- Given a collection of documents:
  - retrieve documents that are relevant to a given query
- **Match terms in documents to terms in query**
- Vector space method
  - term (rows) by document (columns) matrix, based on occurrence
  - translate into vectors in a vector space
    - one vector for each document + query
  - cosine to measure distance between vectors (documents)
    - small angle → large cosine → similar
    - large angle → small cosine → dissimilar

Two problems

- **synonymy**: many ways to refer to the same thing, e.g. car and automobile
  - Term matching leads to poor recall
- **polysemy**: many words have more than one meaning, e.g. model, python, chip
  - Term matching leads to poor precision

Two problems



Synonymy

Will have small cosine

but are related

Polysemy

Will have large cosine

but not truly related

Solutions

- Use dictionaries
  - Fixed set of word relations
  - Generated with years of human labour
  - Top-down solution
- Use latent semantics methods
  - Word relations emerge from the corpus
  - Automatically generated
  - Bottom-up solution

Dictionaries

- WordNet
  - http://wordnet.princeton.edu/
  - Library and Web API

Latent Semantic Indexing (LSI)

- First non-dictionary solution to these problems
- developed at Bellcore (now Telcordia) in the late 1980s (1988). It was patented in 1989.
- http://lsi.argreenhouse.com/lsi/LSI.html

LSI pubs

- Dumais, S. T., Furnas, G. W., Landauer, T. K. and Deerwester, S. (1988), "Using latent semantic analysis to improve information retrieval." In Proceedings of CHI'88: Conference on Human Factors in Computing, New York: ACM, 281-285.
- Deerwester, S., Dumais, S. T., Landauer, T. K., Furnas, G. W. and Harshman, R.A. (1990) "Indexing by latent semantic analysis." Journal of the Society for Information Science, 41(6), 391-407.
- Foltz, P. W. (1990) "Using Latent Semantic Indexing for Information Filtering". In R. B. Allen (Ed.) Proceedings of the Conference on Office Information Systems, Cambridge, MA, 40-47.

LSI (Indexing) vs. LSA (Analysis)

- LSI: the use of latent semantic methods to build a more powerful index (for info retrieval)
- LSA: the use latent semantic methods for document/ corpus analysis

Basic Goal of LS methods

		D_1	D_2	D_3	…	D_M
	$Term_1$	$tdidf_{1,1}$	$tdidf_{1,2}$	$tdidf_{1,3}$	…	$tdidf_{1,M}$
(e.g. car)	$Term_2$	$tdidf_{2,1}$	$tdidf_{2,2}$	$tdidf_{2,3}$	…	$tdidf_{2,M}$
	$Term_3$	$tdidf_{3,1}$	$tdidf_{3,2}$	$tdidf_{3,3}$	…	$tdidf_{3,M}$
(e.g. automobile)	$Term_4$	$tdidf_{4,1}$	$tdidf_{4,2}$	$tdidf_{4,3}$	…	$tdidf_{4,M}$
	$Term_5$	$tdidf_{5,1}$	$tdidf_{5,2}$	$tdidf_{5,3}$	…	$tdidf_{5,M}$
	$Term_6$	$tdidf_{6,1}$	$tdidf_{6,2}$	$tdidf_{6,3}$	…	$tdidf_{6,M}$
	$Term_7$	$tdidf_{7,1}$	$tdidf_{7,2}$	$tdidf_{7,3}$	…	$tdidf_{7,M}$
	$Term_8$	$tdidf_{8,1}$	$tdidf_{8,2}$	$tdidf_{8,3}$	…	$tdidf_{8,M}$
	…					
	$Term_N$	$tdidf_{N,1}$	$tdidf_{N,2}$	$tdidf_{N,3}$	…	$tdidf_{N,M}$

Given N x M matrix

Basic Goal of LS methods

	D_1	D_2	D_3	…	D_M
$Concept_1$	$v_{1,1}$	$v_{1,2}$	$v_{1,3}$	…	$v_{1,M}$
$Concept_2$	$v_{2,1}$	$v_{2,2}$	$v_{2,3}$	…	$v_{2,M}$
$Concept_3$	$v_{3,1}$	$v_{3,2}$	$v_{3,3}$	…	$v_{3,M}$
$Concept_4$	$v_{4,1}$	$v_{4,2}$	$v_{4,3}$	…	$v_{4,M}$
$Concept_5$	$v_{5,1}$	$v_{5,2}$	$v_{5,3}$	…	$v_{5,M}$
$Concept_6$	$v_{6,1}$	$v_{6,2}$	$v_{6,3}$	…	$v_{6,M}$

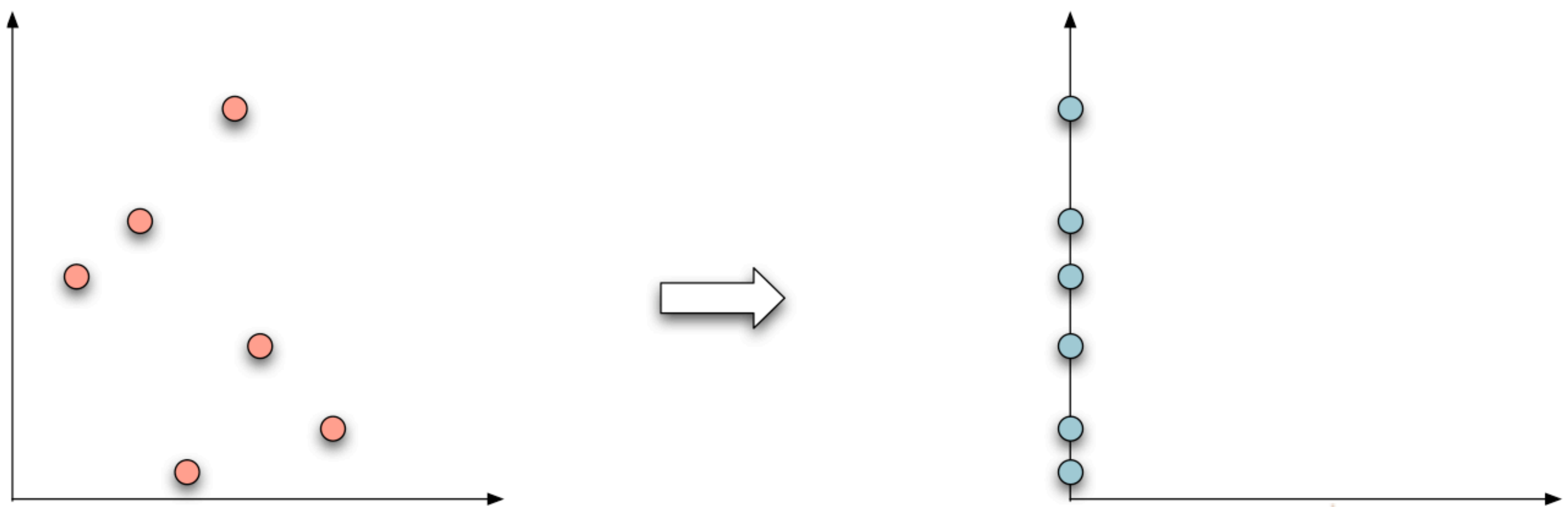
K=6

Squeeze terms such that they reflect **concepts**

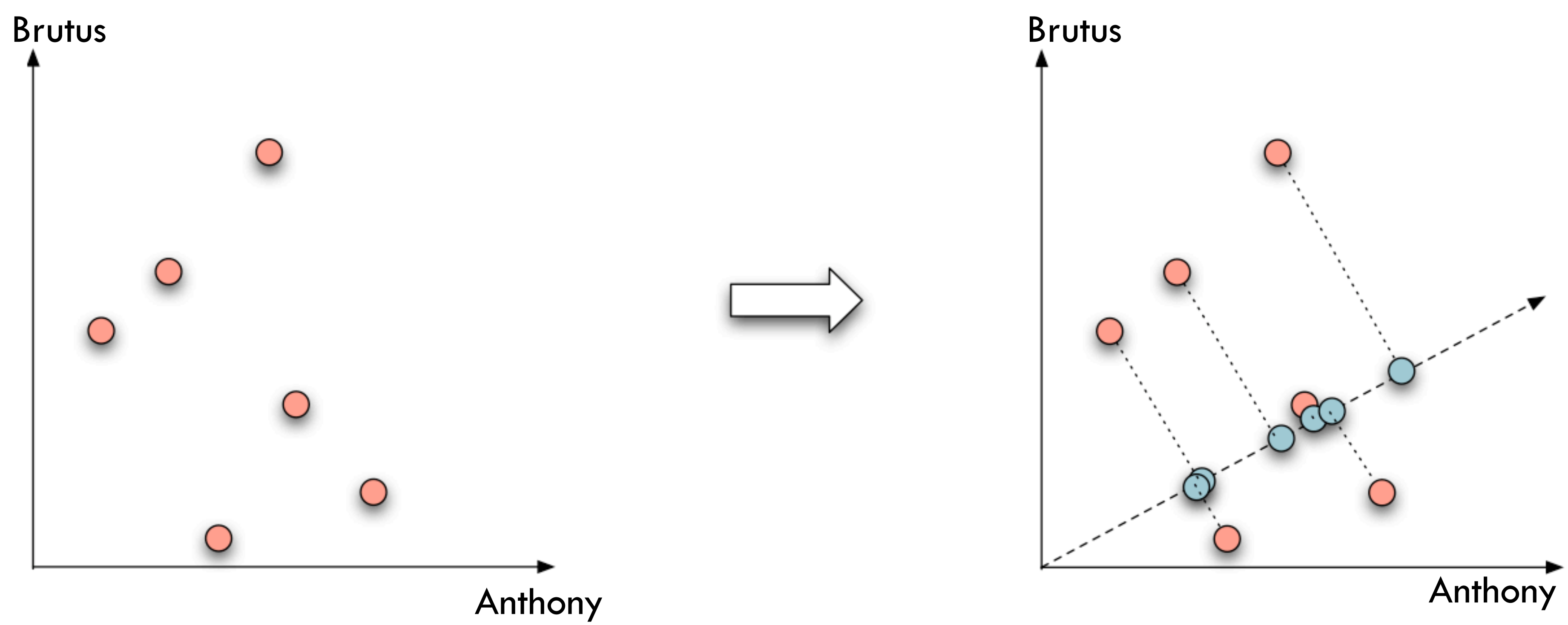
Query matching is performed in the concept space too

Dimensionality Reduction: Projection

- Latent Semantic Indexing always reduces the number of dimensions



Dimensionality Reduction: Projection

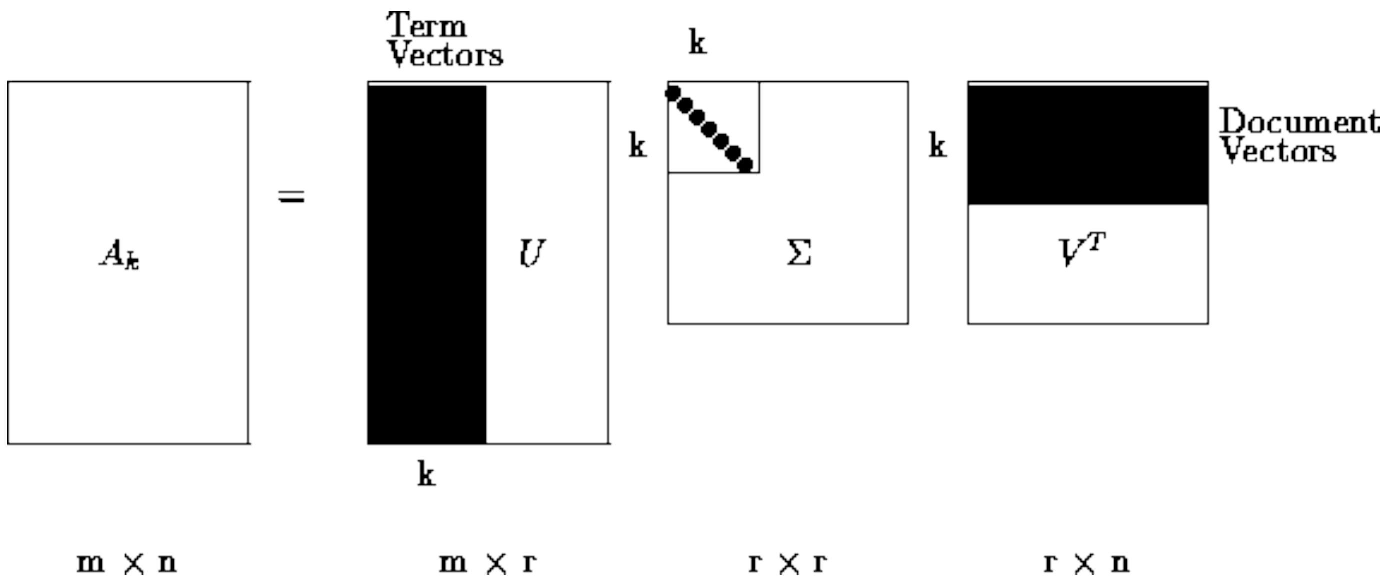


How can this be achieved?

- Math magic to the rescue
- Specifically, linear algebra
- Specifically, matrix decompositions
- Specifically, Singular Value Decomposition (SVD)
- Followed by dimension reduction
  - Honey, I shrunk the vector space!

Singular Value Decomposition

- Singular Value Decomposition
  $A=U\Sigma V^T$ (also $A=TSD^T$)
- Dimension Reduction
  $\sim A= \sim U \sim \Sigma \sim V^T$



SVD

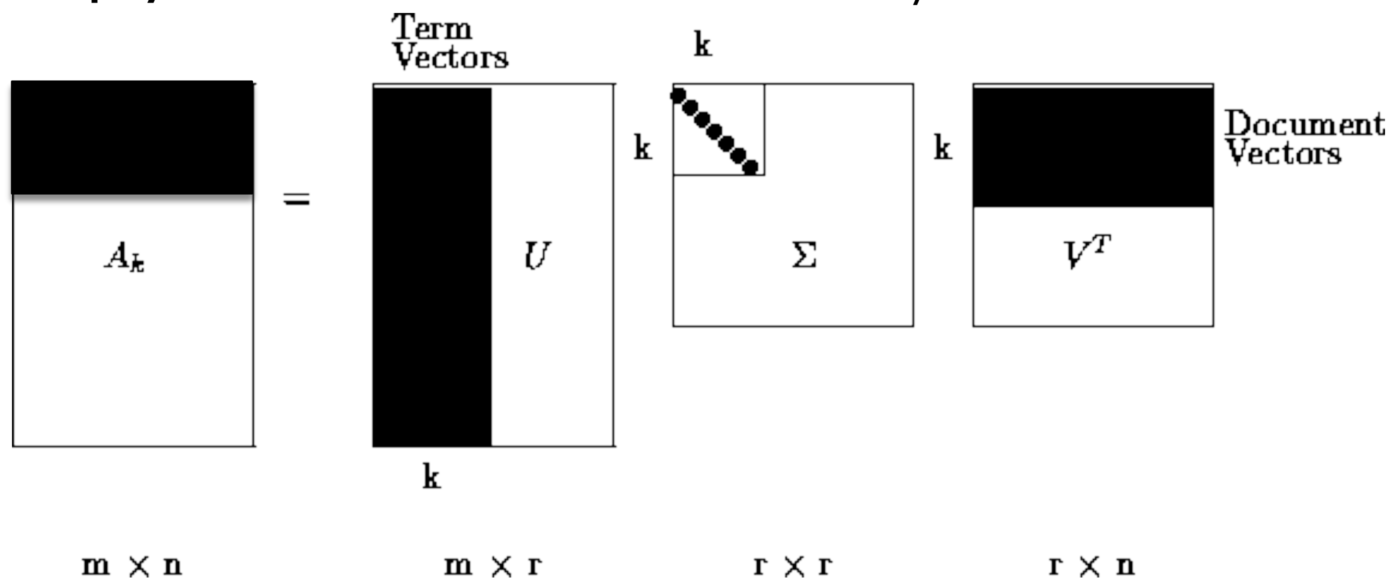
- $A=TSD^T$ such that
  - $TT^T=I$
  - $DD^T=I$
  - S = all zeros except diagonal (singular values); <u>singular values decrease along diagonal</u>

SVD examples

- http://people.revoledu.com/kardi/tutorial/LinearAlgebra/SVD.html
- http://users.telenet.be/paul.larmuseau/SVD.htm
- Many libraries available

Truncated SVD

- SVD is a means to the end goal.
- The end goal is dimension reduction, i.e. get another version of A computed from a reduced space in TSD^T
  - Simply zero S after a certain row/column k



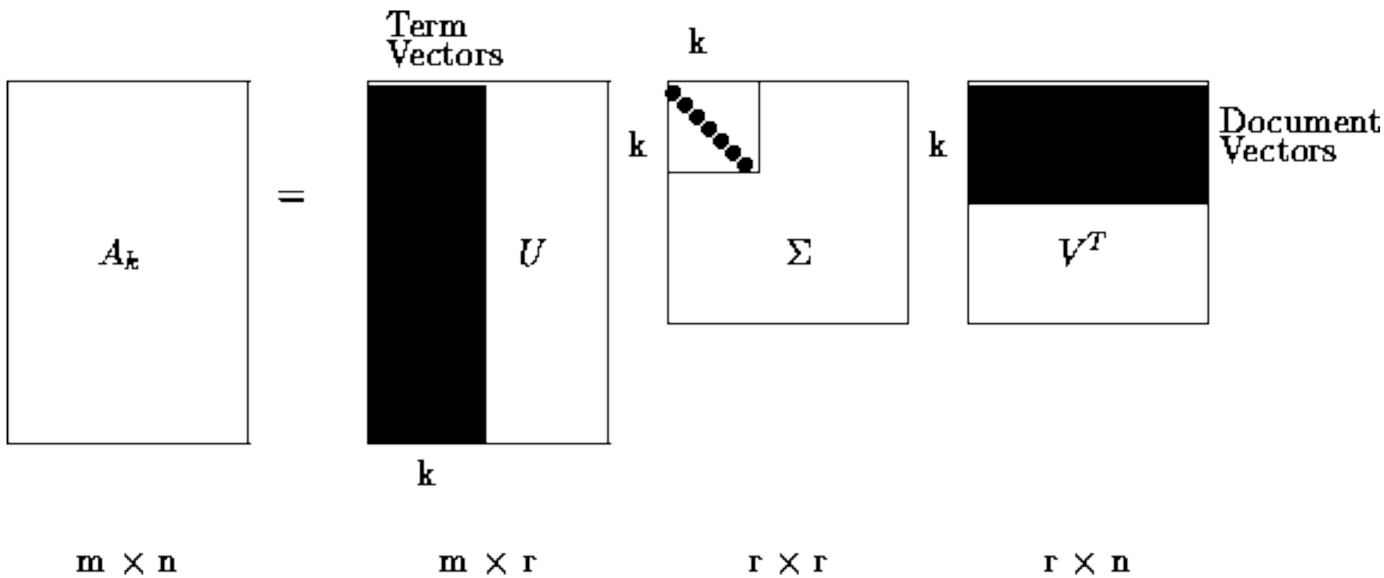
What is ∑ really?

- Remember, diagonal values are in decreasing order

64.9	0	0	0	0
0	29.06	0	0	0
0	0	18.69	0	0
0	0	0	4.84	0

- Singular values represent the strength of latent concepts in the corpus. Each concept emerges from word co-occurrences. (hence the word “latent”)
- By truncating, we are selecting the k strongest concepts
  - Usually in low hundreds
- When forced to squeeze the terms/documents down to a k-dimensional space, the SVD should bring together terms with similar co-occurrences.

SVD in LSI



Term x Document Matrix	Term x Factor Matrix	Singular Values Matrix	Factor x Document Matrix

Properties of LSI

- The computational cost of SVD is significant. This has been the biggest obstacle to the widespread adoption to LSI.
- As we reduce k, recall tends to increase, as expected.
- Most surprisingly, a value of k in the low hundreds can actually *increase* precision on some query benchmarks. This appears to suggest that for a suitable value of k, LSI addresses some of the challenges of synonymy.
- LSI works best in applications where there is little overlap between queries and documents.

Retrieval with LSI

- Query is placed in factor space as a pseudo-document
- Cosine distance to other documents

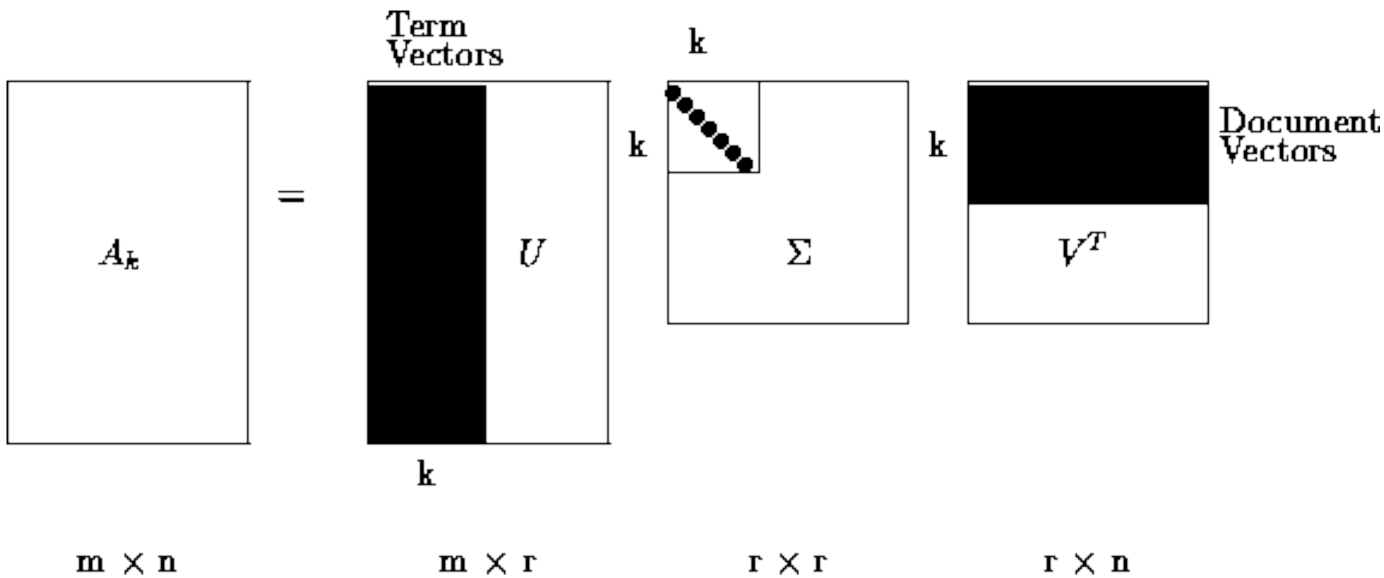
Retrieval with LSI – Example

HCI	c1:	*Human* machine *interface* for Lab ABC *computer* applications
	c2:	A *survey* of *user* opinion of *computer system response time*
	c3:	The *EPS user interface* management *system*
	c4:	*System* and *human system* engineering testing of *EPS*
	c5:	Relation of *user-perceived response time* to error measurement
Graph theory	m1:	The generation of random, binary, unordered *trees*
	m2:	The intersection *graph* of paths in *trees*
	m3:	*Graph minors* IV: Widths of *trees* and well-quasi-ordering
	m4:	*Graph minors*: A *survey*

Example – Term-Document Matrix

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

SVD in LSI



Term x Document Matrix

Term x Factor Matrix

Singular Values Matrix

Factor x Document Matrix

Online calculator

- http://www.bluebit.gr/matrix-calculator/

1	0	0	1	0	0	0	0	0
1	0	1	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0
0	1	1	0	1	0	0	0	0
0	1	1	2	0	0	0	0	0
0	1	0	0	1	0	0	0	0
0	1	0	0	1	0	0	0	0
0	0	1	1	0	0	0	0	0
0	1	0	0	0	0	0	0	1
0	0	0	0	0	1	1	1	0
0	0	0	0	0	0	1	1	1
0	0	0	0	0	0	0	1	1

Note from here on:
The result of this calculator
does not match exactly
the numbers shown in the LSI paper…
Some -/+ are the opposite…
But bottom line is not affected.

SVD – The T (term) Matrix

human	-0.221	-0.113	0.289	-0.415	-0.106	-0.341	-0.523	0.060	0.407
interface	-0.198	-0.072	0.135	-0.552	0.282	0.496	0.070	0.010	0.109
computer	-0.240	0.043	-0.164	-0.595	-0.107	-0.255	0.302	-0.062	-0.492
user	-0.404	0.057	-0.338	0.099	0.332	0.385	-0.003	0.000	-0.012
system	-0.644	-0.167	0.361	0.333	-0.159	-0.207	0.166	-0.034	-0.271
response	-0.265	0.107	-0.426	0.074	0.080	-0.170	-0.283	0.016	0.054
time	-0.265	0.107	-0.426	0.074	0.080	-0.170	-0.283	0.016	0.054
EPS	-0.301	-0.141	0.330	0.188	0.115	0.272	-0.033	0.019	0.165
survey	-0.206	0.274	-0.178	-0.032	-0.537	0.081	0.467	0.036	0.579
trees	-0.013	0.490	0.231	0.025	0.594	-0.392	0.288	-0.255	0.225
graph	-0.036	0.623	0.223	0.001	-0.068	0.115	-0.160	0.681	-0.232
minors	-0.032	0.451	0.141	-0.009	-0.300	0.277	-0.339	-0.678	-0.183

$$A = \mathbf{T}SD^T$$

SVD – Singular Values Matrix

3.341	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	2.542	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	2.354	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	1.645	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	1.505	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	1.306	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.846	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.560	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.364

$$A = T\mathbf{S}D^T$$

SVD – The D^T (document) Matrix

C1	C2	C3	C4	C5	M1	M2	M3	M4
-0.197	-0.606	-0.463	-0.542	-0.279	-0.004	-0.015	-0.024	-0.082
-0.056	0.166	-0.127	-0.232	0.107	0.193	0.438	0.615	0.530
0.110	-0.497	0.208	0.570	-0.505	0.098	0.193	0.253	0.079
-0.950	-0.029	0.042	0.268	0.150	0.015	0.016	0.010	-0.025
0.046	-0.206	0.378	-0.206	0.327	0.395	0.349	0.150	-0.602
-0.077	-0.256	0.724	-0.369	0.035	-0.300	-0.212	0.000	0.362
-0.177	0.433	0.237	-0.265	-0.672	0.341	0.152	-0.249	-0.038
0.014	-0.049	-0.009	0.019	0.058	-0.454	0.762	-0.450	0.070
0.064	-0.243	-0.024	0.084	0.262	0.620	-0.018	-0.520	0.454

$$A = TS\mathbf{D}^{\mathbf{T}}$$

SVD with k=2

$TS_2 =$

Distribution of topics over terms

human	-0.738	-0.287	0.000	0.000	0.000	0.000	0.000	0.000	0.000
interface	-0.662	-0.183	0.000	0.000	0.000	0.000	0.000	0.000	0.000
computer	-0.802	0.109	0.000	0.000	0.000	0.000	0.000	0.000	0.000
user	-1.350	0.145	0.000	0.000	0.000	0.000	0.000	0.000	0.000
system	-2.152	-0.425	0.000	0.000	0.000	0.000	0.000	0.000	0.000
response	-0.885	0.272	0.000	0.000	0.000	0.000	0.000	0.000	0.000
time	-0.885	0.272	0.000	0.000	0.000	0.000	0.000	0.000	0.000
EPS	-1.006	-0.358	0.000	0.000	0.000	0.000	0.000	0.000	0.000
survey	-0.688	0.697	0.000	0.000	0.000	0.000	0.000	0.000	0.000
trees	-0.043	1.246	0.000	0.000	0.000	0.000	0.000	0.000	0.000
graph	-0.120	1.584	0.000	0.000	0.000	0.000	0.000	0.000	0.000
minors	-0.107	1.146	0.000	0.000	0.000	0.000	0.000	0.000	0.000

SVD -- A_2

$S_2 D^T =$

Distribution of topics over documents

C1	C2	C3	C4	C5	M1	M2	M3	M4
-0.658	-2.025	-1.547	-1.811	-0.932	-0.013	-0.050	-0.080	-0.274
-0.142	0.422	-0.323	-0.590	0.272	0.491	1.113	1.563	1.347
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Query

"Human computer interaction"

?

<u>Query is placed at the centroid of the query terms in the concept space</u>

Plot in the first 2 dimensions

$T_{1,2}$

human	-0.221	-0.113
interface	-0.198	-0.072
computer	-0.240	0.043
user	-0.404	0.057
system	-0.644	-0.167
response	-0.265	0.107
time	-0.265	0.107
EPS	-0.301	-0.141
survey	-0.206	0.274
trees	-0.013	0.490
graph	-0.036	0.623
minors	-0.032	0.451

$D^T_{1,2}$

C1	C2	C3	C4	C5	M1	M2	M3	M4
-0.197	-0.606	-0.463	-0.542	-0.279	-0.004	-0.015	-0.024	-0.082
-0.056	0.166	-0.127	-0.232	0.107	0.193	0.438	0.615	0.530

2d plot just for illustration purposes.
In practice it's still a 9d space

placed at the centroid of its terms

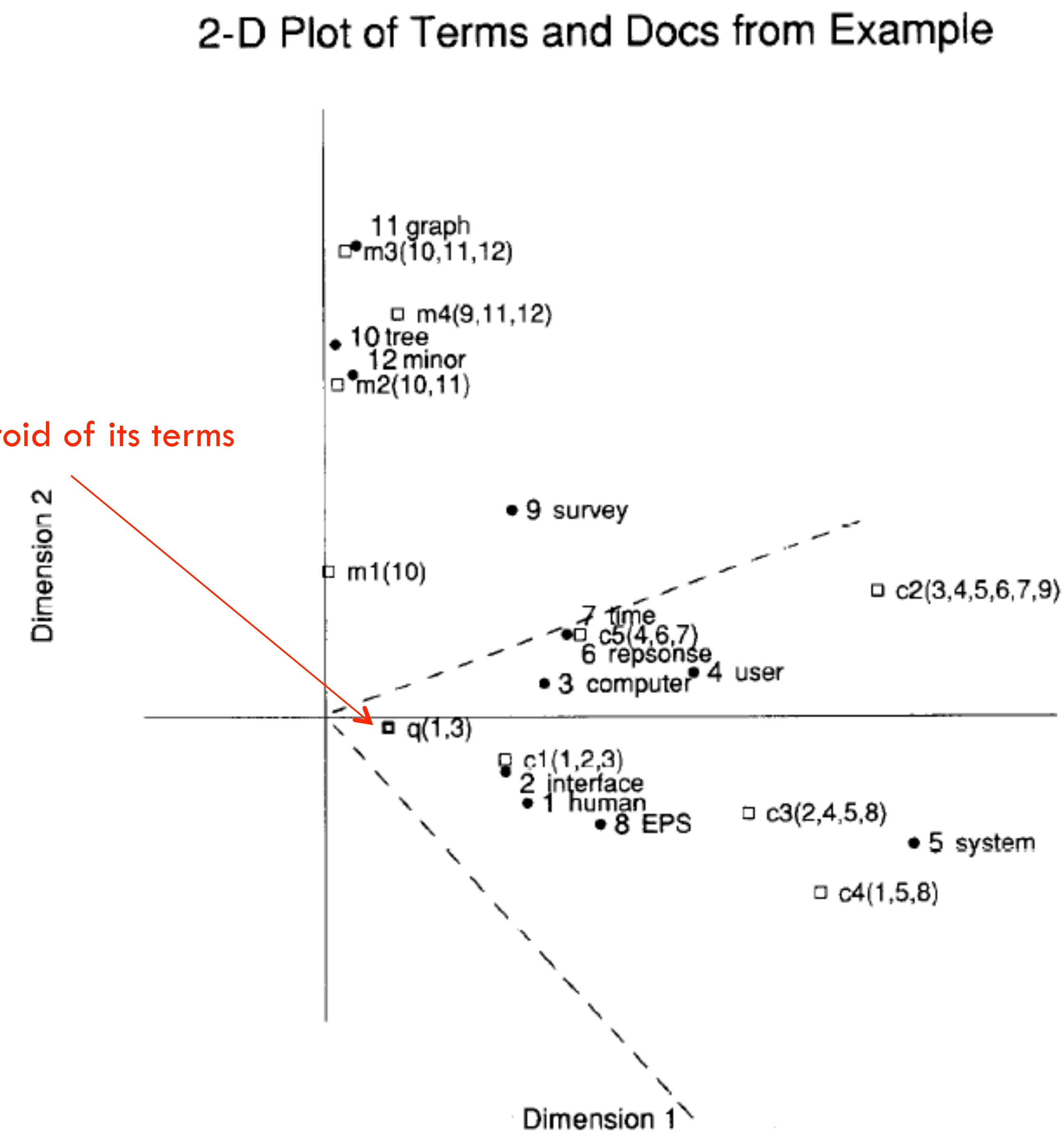


FIG. 1. A two-dimensional plot of 12 Terms and 9 Documents from the sampe TM set. Terms are represented by filled circles. Documents are shown as open squares, and component terms are indicated parenthetically. The query ("human computer interaction") is represented as a pseudo-document at

Centroid (geometric center)

$$C = (x_1 + x_2 + \ldots + x_n) / N$$

$$C_q = [-0.23 \quad 0.035]$$