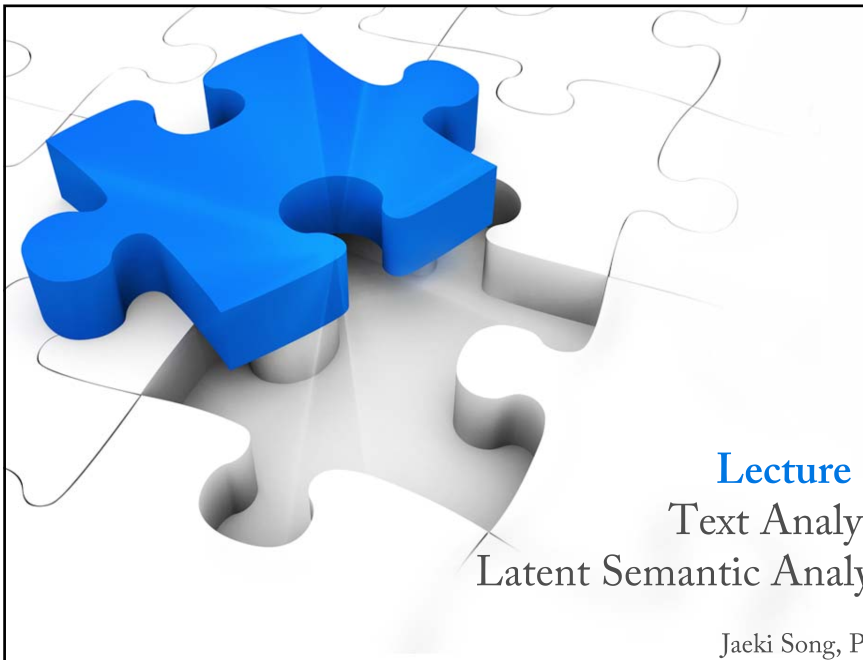


Lecture 11:
Text Analysis:
Latent Semantic Analysis

Jaeki Song, Ph.D.

Latent Semantic Analysis (LSA)

Latent Semantic Analysis (LSA) is a theory and method for extracting and representing the contextual-usage meaning of words by statistical computations applied to a large corpus of text (Landauer and Dumais, 1997).

Data Qualification		Core Analysis	Quantitative Analysis
Pre-Processing	Term Weighting	Singular Value Decomposition	Factor Analysis
Compile terms Correct typos Eliminate unique words Eliminate stop words Stemming	Importance of term in local and global document	reduce dimension of matrix	find meaningful topics

LSA Procedure

Latent Semantic Analysis (LSA)

- LSA is an algorithmically well-defined way of measuring lexical co-occurrence in some set of text
- The assumption is that co-occurrence says something about semantics: words about the same things are likely to occur in the same contexts

How is an LSA model constructed?

1. Build a term-document matrix with
    - rows representing words
    - columns representing documents

	Doc 1	Doc2	……
data			
example			
introduction			
package			
……			

How is an LSA model constructed?

2. Enter term frequency in each cell
   - How many times the word *i* appear in **the document *j***

	Doc 1	Doc2	……
data	1	3	
example	1	0	
introduction	0	1	
package	1	0	
……			

How is an LSA model constructed?

3. Transform the matrix (term weighting scheme)

$$weight_{i,j} = lw(tf_{i,j}) \times gw(tf_{i,j})$$

a) Control for word frequency (lw: local weight)
   - compress the effects of frequency

b) Control for the number of documents each word appeared in (gw: global weight)
   - Words that occur in few documents are more informative about those documents than words that appear in many different documents

Term Weighting Scheme: TF-IDF

$$weight_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

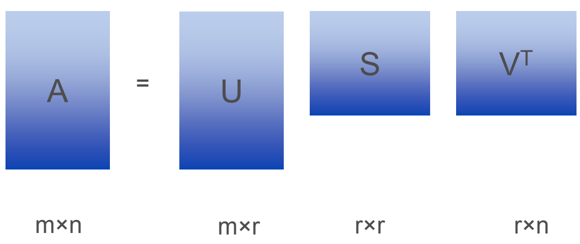
- $tf_{i,j}$ = *number of occurrences of term i in document j*
- df_i = *number of documents containing term i*
- N = *total number of documents*

✓ Term Frequency (TF): importance of the term within that document
✓ Inverse Document Frequency (IDF): importance of the term in the corpus

- Word occurs in many documents is less useful; its IDF($\log\left(\frac{N}{df_i}\right)$)value is low

Singular Value Decomposition

- This reduces dimensionality by "projecting" the tens of thousands of dimensions onto a smaller number.
  - unique mathematical decomposition of a matrix into the product of three matrices:
    - two with orthonormal columns
    - one with singular values on the diagonal



{A}={U}{S}{V}^{T}

Singular Value Decomposition

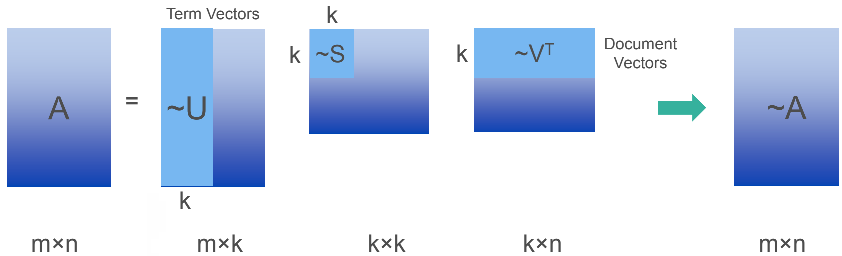
- Example

$$A = \begin{bmatrix} 3 & 1 & 1 \\ -1 & 3 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \sqrt{12} & 0 & 0 \\ 0 & 0 & \sqrt{10} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{6}} & \frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} \\ \frac{2}{\sqrt{5}} & \frac{-1}{\sqrt{5}} & 0 \\ \frac{1}{\sqrt{30}} & \frac{2}{\sqrt{30}} & \frac{-5}{\sqrt{30}} \end{bmatrix}$$

LSA - Singular Value Decomposition

- The “discarded” dimensions are those that are least informative, which are redundant
- The optimal SVD result contains the smallest number of dimensions but the most informative information.



$\{\sim A\}\sim=\{\sim U\}\{\sim S\}\{\sim V\}^T$

How is the LSA model used?

- To get a measure of how related a word is to another word, measure the distance between the columns containing the two words.
  - This gives you a measure of how different the contexts of the two words were: that is, how often a word occurred a different number of times in each context
- You can also take the distance between two document vectors to get a measure of how related they are.
- You can measure distance by taking the cosine between two vectors

A Small Example

Technical Memo Titles

c1: *Human* machine *interface* for 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

m1: The generation of random, binary, ordered *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*

A Small Example

X =	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

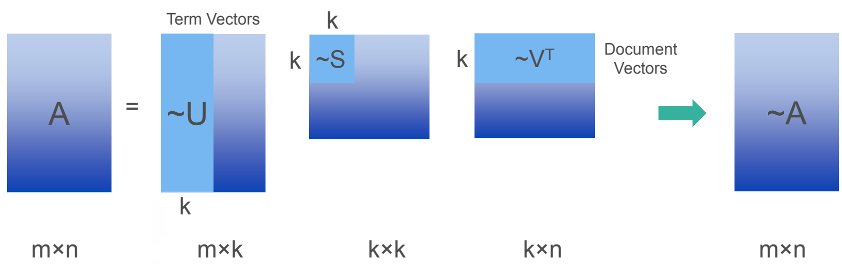
A Small Example

- Singular Value Decomposition

$$\{A\}=\{U\}\{S\}\{V\}^T$$

- Dimension Reduction

$$\{\sim A\}\sim=\{\sim U\}\{\sim S\}\{\sim V\}^T$$



A Small Example – 4

- {U} =

0.22	-0.11	0.29	-0.41	-0.11	-0.34	0.52	-0.06	-0.41
0.20	-0.07	0.14	-0.55	0.28	0.50	-0.07	-0.01	-0.11
0.24	0.04	-0.16	-0.59	-0.11	-0.25	-0.30	0.06	0.49
0.40	0.06	-0.34	0.10	0.33	0.38	0.00	0.00	0.01
0.64	-0.17	0.36	0.33	-0.16	-0.21	-0.17	0.03	0.27
0.27	0.11	-0.43	0.07	0.08	-0.17	0.28	-0.02	-0.05
0.27	0.11	-0.43	0.07	0.08	-0.17	0.28	-0.02	-0.05
0.30	-0.14	0.33	0.19	0.11	0.27	0.03	-0.02	-0.17
0.21	0.27	-0.18	-0.03	-0.54	0.08	-0.47	-0.04	-0.58
0.01	0.49	0.23	0.03	0.59	-0.39	-0.29	0.25	-0.23
0.04	0.62	0.22	0.00	-0.07	0.11	0.16	-0.68	0.23
0.03	0.45	0.14	-0.01	-0.30	0.28	0.34	0.68	0.18

A Small Example – 5

- {S} =

3.34								
	2.54							
		2.35						
			1.64					
				1.50				
					1.31			
						0.85		
							0.56	
								0.36

A Small Example – 6

- {V} =

0.20	0.61	0.46	0.54	0.28	0.00	0.01	0.02	0.08
-0.06	0.17	-0.13	-0.23	0.11	0.19	0.44	0.62	0.53
0.11	-0.50	0.21	0.57	-0.51	0.10	0.19	0.25	0.08
-0.95	-0.03	0.04	0.27	0.15	0.02	0.02	0.01	-0.03
0.05	-0.21	0.38	-0.21	0.33	0.39	0.35	0.15	-0.60
-0.08	-0.26	0.72	-0.37	0.03	-0.30	-0.21	0.00	0.36
0.18	-0.43	-0.24	0.26	0.67	-0.34	-0.15	0.25	0.04
-0.01	0.05	0.01	-0.02	-0.06	0.45	-0.76	0.45	-0.07
-0.06	0.24	0.02	-0.08	-0.26	-0.62	0.02	0.52	-0.45

A Small Example – 7

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	0.16	0.40	0.38	0.47	0.18	-0.05	-0.12	-0.16	-0.09
interface	0.14	0.37	0.33	0.40	0.16	-0.03	-0.07	-0.10	-0.04
computer	0.15	0.51	0.36	0.41	0.24	0.02	0.06	0.09	0.12
user	0.26	0.84	0.61	0.70	0.39	0.03	0.08	0.12	0.19
system	0.45	1.23	1.05	1.27	0.56	-0.07	-0.15	-0.21	-0.05
response	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
time	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
EPS	0.22	0.55	0.51	0.63	0.24	-0.07	-0.14	-0.20	-0.11
survey	0.10	0.53	0.23	0.21	0.27	0.14	0.31	0.44	0.42
trees	-0.06	0.23	-0.14	-0.27	0.14	0.24	0.55	0.77	0.66
graph	-0.06	0.34	-0.15	-0.30	0.20	0.31	0.69	0.98	0.85
minors	-0.04	0.25	-0.10	-0.21	0.15	0.22	0.50	0.71	0.62

A Small Example – 2 reprise

	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

LSA in R

- Required package: library(tm), library(lsa), library(Matrix)
- TF-IDF:

```
corpus.tdm.weig <- weightTfIdf(corpus.tdm, normalize = TRUE)
```

- SVD:

```
#Specify the dimensions.
userdimension=2
# Create LSA
corpus.tdm.weig.lsa <- lsa( corpus.tdm.weig, dims=userdimension)
# term matrix
tk<-as.matrix(corpus.tdm.weig.lsa$tk)
# diagnoal matrix
sk<-Diagonal(n=userdimension, as.matrix(corpus.tdm.weig.lsa$sk))
# doc matrix
dk<-as.matrix(corpus.tdm.weig.lsa$dk)
```

LSA in R

- Term loading and document loading

```
# term loading
termloading <- tk %*% sk
#write the term loading
write.csv(as.matrix(termloading), file="term_loading.csv")
# document loading
docloading <- dk %*% sk
#write the document loading
write.csv(as.matrix(docloading), file="doc_loading.csv")
```

- Rotation

```
# term loading after rotation
termloading.rot <- varimax(as.matrix(termloading), normalize = TRUE, eps = 1e-5)
#document loading after the rotation
docloading.rot<-as.matrix(docloading) %*% as.matrix(termloading.rot[2]$rotmat)
```