An R Package for Latent Semantic Analysis

GfKl, March 9th 2006, Berlin

Fridolin Wild
Vienna University of Economics and Business Administration

Structure of the Talk

- Introduction to Latent Semantic Analysis
- LSA Process & Driving Parameters
- The Package
- Demo I: Essay Scoring with LSA
- Demo II: The Geometry of Meaning
- Benchmarking Effectiveness
- Conclusion & Future Plans

<2>

Input (e.g., documents)

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*

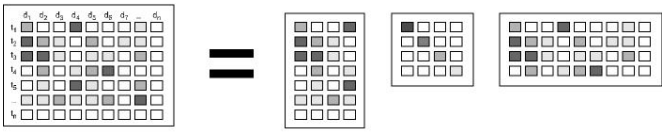
Only the red terms appear in more than one document, so strip the rest.

{ M } =

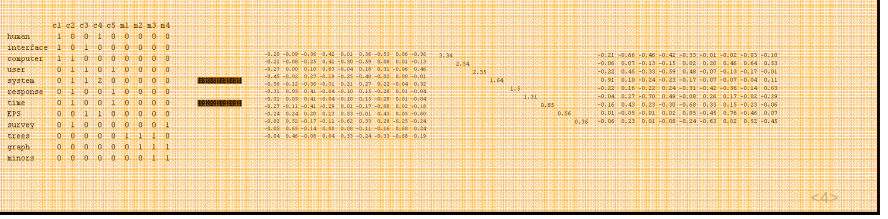
	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors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

Deerwester, Dumais, Furnas, Landauer, and Harshman (1990): Indexing by Latent Semantic Analysis, In: Journal of the American Society for Information Science, 41(6):391-407

Singular Value Decomposition

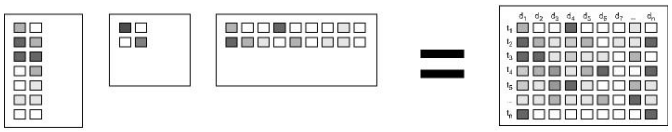


$$M = T\ S\ D^T$$



Latent Semantics

- Assumption: language utterances have a semantic structure
- However, this structure is obscured by word usage (noise, synonymy, polysemy, …)
- Proposed LSA Solution: map doc-term matrix using conceptual indices derived statistically (truncated SVD)



$$T_k \, S_k \, D_k^T = M_k$$

Reconstructed, Reduced Matrix

	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

m4: *Graph minors*: A *survey*

doc2doc - similarities

unreduced
- Based on $M = TSD'$
- Pearson Correlation over document vectors

	c1	c2	c3	c4	c5	m1	m2	m3	m4
c1	1.00								
c2	-0.19	1.00							
c3	0.00	0.00	1.00						
c4	0.00	0.00	0.47	1.00					
c5	-0.33	0.58	0.00	-0.31	1.00				
m1	-0.17	-0.30	-0.21	-0.16	-0.17	1.00			
m2	-0.26	-0.45	-0.32	-0.24	-0.26	0.67	1.00		
m3	-0.33	-0.58	-0.41	-0.31	-0.33	0.52	0.77	1.00	
m4	-0.33	-0.19	-0.41	-0.31	-0.33	-0.17	0.26	0.56	1.00

reduced
- based on $M_2 = TS_2D'$
- Pearson Correlation over document vectors

	c1	c2	c3	c4	c5	m1	m2	m3	m4
c1	1.00								
c2	0.91	1.00							
c3	1.00	0.91	1.00						
c4	1.00	0.88	1.00	1.00					
c5	0.85	0.99	0.85	0.81	1.00				
m1	-0.85	-0.56	-0.85	-0.88	-0.45	1.00			
m2	-0.85	-0.56	-0.85	-0.88	-0.44	1.00	1.00		
m3	-0.85	-0.56	-0.85	-0.88	-0.44	1.00	1.00	1.00	
m4	-0.81	-0.50	-0.81	-0.84	-0.37	1.00	1.00	1.00	1.00

<7>

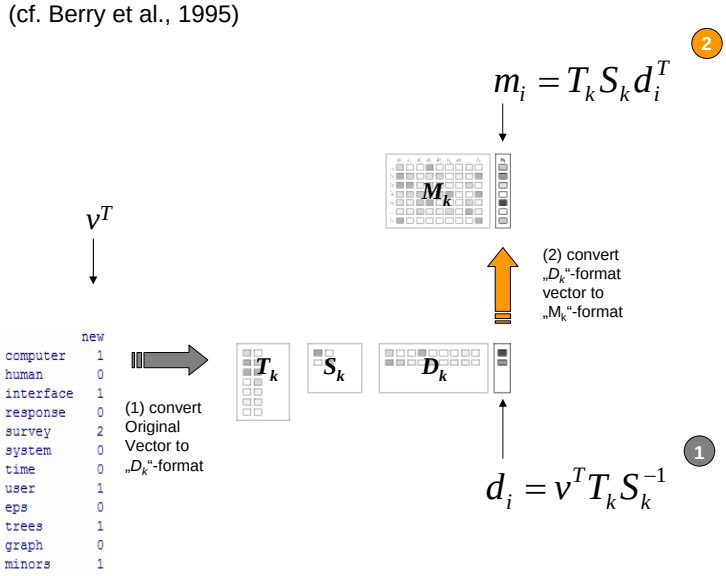
SVD-Updating: Folding-In

- SVD factor stability
  - SVD calculates factors over a given text base
  - Different texts – different factors
  - Challenge: avoid unwanted factor changes (e.g., bad essays)
  - Solution: folding-in of essays instead of recalculating
- SVD is computationally expensive
  - 14 seconds (300 docs textbase, this machine)
  - 10 minutes (3500 docs textbase, this machine)
  - … and rising!

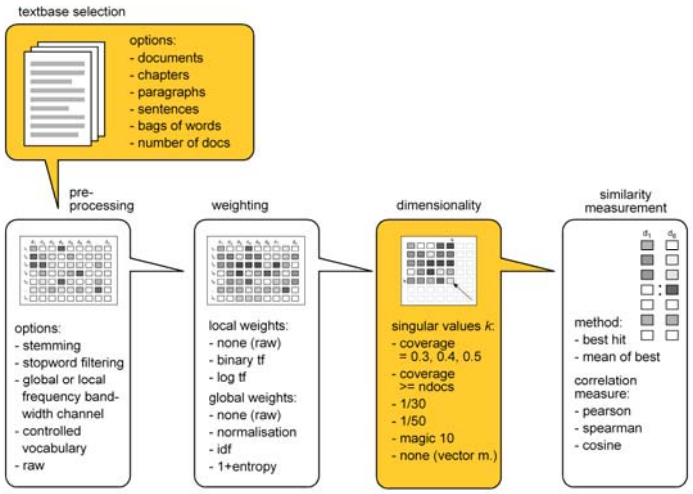
<8>

Folding-In in Detail

(cf. Berry et al., 1995)



LSA Process & Driving Parameters



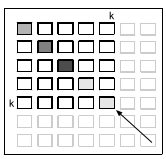
The Package

- Available via CRAN, e.g.:
  http://cran.at.r-project.org/src/contrib/Descriptions/lsa.html

- Higher-level Abstraction to Ease Use
  - Five core methods:
    `textmatrix() / query()`
    `lsa()`
    `fold_in()`
    `as.textmatrix()`
  - Supporting methods for term weighting, dimensionality calculation, correlation measurement, triple binding

SVD-Dimensionality

- Fixed number *k*
  `lsa( tm, dims=50 )`
- Percentage of cumulated values (50%, 40%, 30%)
  `lsa( tm, dims=dimcalc_share(share=0.5) )`
- Share of values = number of docs
  `lsa( tm, dims=dimcalc_ndocs(100) )`
- Absolute fraction of *k* (1/50 and 1/30)
  `lsa( tm, dims=round(k/50) )`
- ‘magic 10’
  `lsa( tm, dims=10 )`
- Kaiser Criterium
  `lsa( tm, dims=dimcalc_kaiser() )`
- Raw (= no LSA, pure vector model!)
  `lsa( tm, dims=dimcalc_raw() )`



Term Weighting Schemes

- Local Weights (LW)
  - None ('raw' *tf*)
  - Binary Term Frequency
  - Logarithmized Term Frequency (log)
- Global Weights (GW)
  - None ('raw' *tf*)
  - Normalisation

$$norm_i = \frac{1}{\sqrt{\sum_j^1 tf_{ij}^2}}$$

  - Inverse Document Frequency (IDF)

$$idf_i = \log_2\left(\frac{numdocs}{docfreq(i)}\right)+1$$

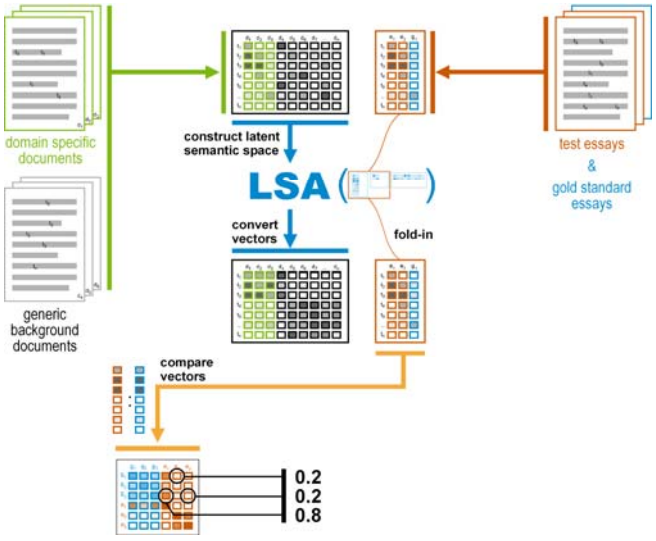
  - 1 + Entropy

$$entplusone_i = 1 - \sum_j^1 \frac{p_{ij}\log p_{ij}}{\log numdocs}, \text{ where } p_{ij} = \frac{tf_{ij}}{\sum_j^1 tf_{ij}}$$

- 12 Combinations

$$weight_{ij} = lw(tf_{ij}) \cdot gw(tf_{ij})$$

Demo I: Essay Scoring with LSA



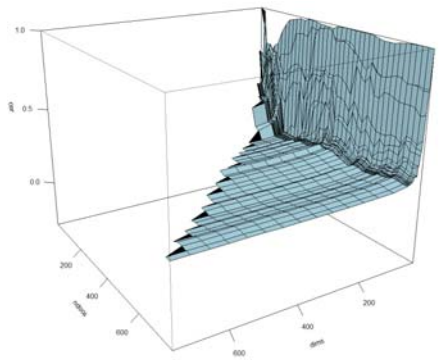
Essay Scoring (Code)

```
library( "lsa“ )                          # load package

# load training texts
trm = textmatrix( "trainingtexts/“ )
trm = lw_bintf( trm ) * gw_idf( trm )  # weighting
space = lsa( trm )                        # create an LSA space

# fold-in essays to be tested (including gold standard text)
tem = textmatrix( "testessays/", vocabulary=rownames(trm) )
tem_red = fold_in( tem, space )

# score an essay by comparing with
# gold standard text (very simple method!)
cor( tem_red[,"goldstandard.txt"], tem_red[,"E1.txt"] )
=> 0.7
```

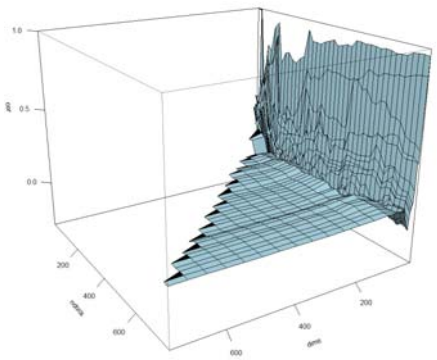

Demo II: Geometry of Meaning

Semantic Relations in a Latent Semantic Space

Pearson(eu, österreich)



Pearson(jahr, wien)

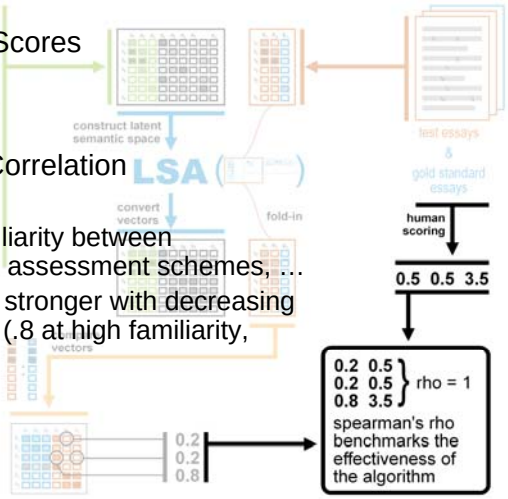


Demo II: Geometry of Meaning

```
# file in a corpus
mymatrix=textmatrix("kurier1200/", stopwords=stopwords_de)

# randomize document order
rnd_sample = sample(1:ncol(mymatrix))
sample_matrix = mymatrix[,rnd_sample]

# calculate term-to-term similarities with increasing
# matrix size and increasing number of dimensions
for (i in docseq) {
   m = sample_matrix[-(which(rowSums(sample_matrix[,1:i])==0)),1:i]
   for (dim in dimseq) {
        space = lsa(m, dims=dim)
        redm = as.textmatrix(space)
        ttsim = append( ttsim, cor(redm["eu",], redm["österreich",] )
   }
}
```

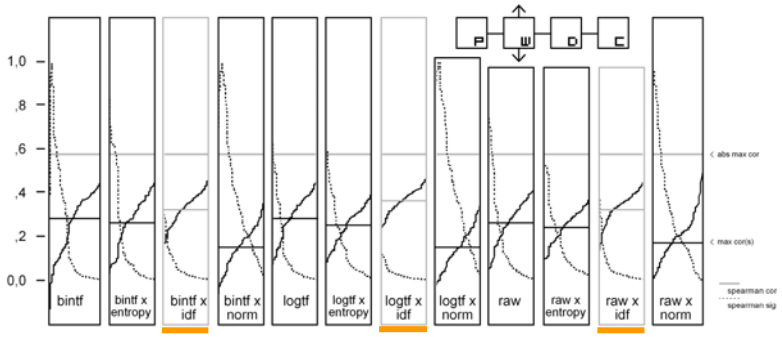

Benchmarking Effectiveness

- Compare Machine Scores with Human Scores
- Human-to-Human Correlation
  - Usually around .6
  - Increased by familiarity between assessors, tighter assessment schemes, ...
  - Scores vary even stronger with decreasing subject familiarity (.8 at high familiarity, worst test -.07)



•Test Collection: 43 German Essays, scored from 0 to 5 points (ratio scaled), average length: 56.4 words
•Training Collection: 3 'golden essays', plus 302 documents from a marketing glossary, average length: 56.1 words

Benchmarking Effectiveness



- Global Weights:
  - IDF overall best (.36 with logtf)
  - Normalisation worsens (.15 - .17)
  - 1+Entropy: nearly no effect
- Local Weights:
  - hardly any effect
  - raw and logtf squeeze curve

- Best 50:
  - 20 x bintf
  - 19 x logtf
  - 11 x raw
  - 26 x IDF
  - 13 x raw
  - 6 x normalisation
  - 5 x 1+entropy

(Wild et al., 2005)

<19>

Conclusion & Future Plans

- LSA/LSI fine in settings, where fuzziness is good (e.g., information retrieval, recommender systems)
- LSA/LSI sometimes worse than vector model in settings, where proper representation structures are needed (e.g., essay scoring, term relationship mining)
- Huge optimisation potential in corpus compilation – selection model? Training process?
- Huge optimisation potential in parameter tuning – model?
- Fast and simple predictors for parameter tuning?

<20>