An R Package for Latent Semantic Analysis
LSA-TEL, March 29th 2007, Heerlen, NL

Fridolin Wild
Vienna University of Economics and Business Administration
Structure of the Talk

- Concepts of the Package
- Analysis Process
- Driving Parameters
- Evaluating Algorithm Effectiveness
- Demo I: Essay Scoring
- Demo II: Coding Qualitative Interviews
- Demo III: The Geometry of Meaning
- Conclusion & Future Plans
Concepts used in the *lsa* Package for R
Singular Value Decomposition

\[ M = T S D^T \]
Latent Semantic Structure

- Assumption: ‘language’ utterances have a semantic structure, i.e. a certain form in their meaning
- However: this structure is obscured by word usage (noise, synonymy, polysemy, …)
- Proposed solution: map doc-term matrix using conceptual indices derived statistically (truncated SVD)
Concepts used in the Package

- **Term** = feature
- **Vocabulary** = ordered set of features
- **Corpus** = document collection
- **Textmatrix** = occurrence matrix (of terms in documents)
- **Weighting** = specifies the importance of certain features
- **Latent-Semantic Space** = SVD partial matrices
- **Dimension** = singular value
- **Folding In** = adding additional evidence
- **Distance / Similarity** = measure for semantic distance
- **Query** = pseudo document used to identify similar documents
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
Analysis Process
Core Workflow

- \( \text{tm} = \text{textmatrix}('dir/') \)
- \( \text{tm} = \text{lw} \text{ logtf}(\text{tm}) \times \text{gw_idf}(\text{tm}) \)
- \( \text{space} = \text{lsa}(\text{tm},\) \text{dims=dimcalc\_share()} \)
- \( \text{tm3} = \text{fold\_in}(\text{tm}, \text{space}) \)
- \( \text{as.textmatrix}(\text{tm}) \)
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!
Driving Parameters
LSA Process & Driving Parameters

### Textbase Selection
- documents
- chapters
- paragraphs
- sentences
- context bags
- number of docs

### Pre-processing
- stemming
- stopword filtering
- global or local frequency bandwidth channel
- controlled vocabulary
- raw

### Weighting
- local weights:
  - none (raw)
  - binary tf
  - log tf
- global weights:
  - none (raw)
  - normalisation
  - idf
  - 1+entropy

### Dimensionality
- singular values $k$:
  - coverage
  - $= 0.3, 0.4, 0.5$
  - $\geq \text{n docs}$
  - $1/30$
  - $1/50$
  - magic 10
  - none (vector m.)

### Similarity Measurement
- method:
  - best hit
  - mean of best
- correlation measure:
  - pearson
  - spearman
  - cosine
Term Weighting Schemes

\[ \text{weight}_{ij} = \text{lw}(tf_{ij}) \cdot \text{gw}(tf_{ij}) \]

- **Global Weights (GW)**
  - None (‘raw’ \( tf \))
  - Normalisation
    \[
    \text{norm}_i = \frac{1}{\sqrt{\sum_j tf_{ij}^2}}
    \]
  - Inverse Document Frequency (IDF)
    \[
    \text{idf}_i = \log_2 \left( \frac{\text{numdocs}}{\text{docfreq}(i)} \right) + 1
    \]
  - 1 + Entropy
    \[
    \text{entplusone}_i = 1 - \sum_j \frac{p_{ij} \log p_{ij}}{\log \text{numdocs}}, \text{ where } p_{ij} = \frac{tf_{ij}}{\sum_j tf_{ij}}
    \]

- **Local Weights (LW)**
  - None (‘raw’ \( tf \))
  - Binary Term Frequency
  - Logarithmized Term Frequency (log)
SVD-Dimensionality

- Fixed number $k$
  \[
  \text{lsa}( \text{tm}, \text{dims}=50 )
  \]

- Percentage of cumulated values (50%, 40%, 30%)
  \[
  \text{lsa}( \text{tm}, \text{dims} = \text{dimcalc\_share(share=0.5)} )
  \]

- Share of values = number of docs
  \[
  \text{lsa}( \text{tm}, \text{dims} = \text{dimcalc\_ndocs(100)} )
  \]

- Absolute fraction of $k$ (1/50 and 1/30)
  \[
  \text{lsa}( \text{tm}, \text{dims} = \text{round(k/50)} )
  \]

- ‘magic 10’
  \[
  \text{lsa}( \text{tm}, \text{dims}=10 )
  \]

- Kaiser Criterium
  \[
  \text{lsa}( \text{tm}, \text{dims} = \text{dimcalc\_kaiser()} )
  \]

- Raw (= no LSA, pure vector model!)
  \[
  \text{lsa}( \text{tm}, \text{dims} = \text{dimcalc\_raw()} )
  \]
Similarity / Distance Measures

- Cosine (part of the package)
- Pearson (cor)
- Spearman’s R (cor)
- Kendall (cor)
- ...

- R package in planning stage by David Meyer, Kurt Hornik, …
Binding of Triples

- Solution to the symbol-grounding problem ;)
- Triples (subject, predicate, object)
- Bind triples to documents

```python
setTriple(myMatrix, "c1", "category", "cats")
setTriple(myMatrix, "c1", "category", "dogs")
geTriple(myMatrix) => all
gtTriple(myMatrix, "c1", "category") => cats, dogs
```
Evaluating Algorithm Effectiveness
Evaluating Algorithm Effectiveness

- Compare Machine Scores with Human Scores

- Human-to-Human Correlation
  - Usually around 0.6
  - Increased by familiarity between assessors, tighter assessment schemes, ...
  - Scores vary even stronger with decreasing subject familiarity (0.8 at high familiarity, worst test -0.07)

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)
Demonstrations
Demo I: Essay Scoring with LSA

domain specific documents

construct latent semantic space

LSA

convert vectors

fold-in

test essays &
gold standard essays

generic background documents

compare vectors

0.2
0.2
0.8
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: Qualitative Interviews

Table 2
Correlations between LSA and human scores in the European mobile phone market.
library("lsa"); data(stopwords_de); # load package & German stopwords

# read interview texts
tm = textmatrix("A1", stopwords=stopwords_de)
space = lsa(tm, dims=dimcalc_share(share=0.5))
space.tm = as.textmatrix(space)

# read coding scheme & human scores
cs = readLines("codingscheme.txt")
humanscores = read.delim2("humanscores_A1.txt", sep="\n", header=F)

# calculate machine scores
scores = vector(mode="numeric",length=length(cs))
names(scores) = cs
cutoff = 0.5
for ( n in 1:length(cs) ) {
    q.cs = query(cs[n], termlist = rownames(space.tm) )
    q2.cs = fold_in(q.cs,space)
    space.query.tm = cbind(space.tm,q2.cs)
    correlation = cor(space.query.tm, method="pearson")
    scores[n] = length(which(correlation[135,1:134]>cutoff))*100/ncol(space.tm)
}

# evaluate effectiveness
cor.test(scores, humanscores, method="spearman", alternative="two.sided", exact=F)
Demo III: Geometry of Meaning
Semantic Relations in a Latent Semantic Space

Pearson(eu, österreich)  Pearson(jahr, wien)
# 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"] )
  }
}

Conclusion & Future Plans

- Pre-Processing chain can be better organised
- Better Error Handling
- Use Sparse Matrices
- Use partial SVD with Lanczos
- Huge optimisation potential in corpus compilation – selection model? Training process?
- Huge optimisation potential in parameter tuning – model?
- Fast and simple predictors for parameter tuning?
#eof.
Folding-In in Detail
(cf. Berry et al., 1995)

\[ m_i = T_k S_k d_i^T \]

(1) convert Original Vector to "D_k"-format

\[ d_i = \nu^T T_k S_k^{-1} \]

(2) convert "D_k"-format vector to "M_k"-format