Chapter 4: Advanced IR Models

Key Idea of Latent Concept Models

Objective:

Transformation of document vectors from
high-dimensional term vector space into
lower-dimensional **topic vector space** with

- exploitation of term correlations
  (e.g. „Web“ and „Internet“ frequently occur in together)
- implicit differentiation of polysems that exhibit
  different term correlations for different meanings
  (e.g. „Java“ with „Library“ vs. „Java“ with „Kona Blend“ vs. „Java“ with „Borneo“)

mathematically:

given: m terms, n docs (usually n > m) and a
m×n term-document similarity matrix A,
needed: largely similarity-preserving mapping
of column vectors of A
into k-dimensional vector space (k << m) for given k

4.3.1 Foundations from Linear Algebra

A set S of vectors is called **linearly independent** if no
$x \in S$ can be written as a linear combination of other vectors in S.
The **rank** of matrix A is the maximal number of
linearly independent row or column vectors.
A **basis** of an n×n matrix A is a set S of row or column vectors
such that all rows or columns are linear combinations of vectors from S.
A set S of n×1 vectors is an **orthonormal basis** if for all x, y ∈ S:

$$\|x\|_2 := \sqrt{\sum_{i=1}^{n} X_i^2} = 1 = \|y\|_2 \quad and \quad x \cdot y = 0$$

Eigenvalues and Eigenvectors

Let A be a real-valued n×n matrix, x a real-valued n×1 vector, and λ a real-valued scalar. Solutions x and λ of the equation $A \times x = \lambda x$ are called an **Eigenvector** and **Eigenvalue** of A. Eigenvectors of A are vectors whose direction is preserved by the linear transformation described by A.

The Eigenvalues of A are the roots (Nullstellen) of the characteristic polynom f(λ) of A: $f(\lambda) = |A - \lambda I| = 0$

with the determinant (developing the i-th row):

$$|A| = \sum_{j=1}^{n} (-1)^{i+j} a_{ij} \left|A^{(ij)}\right|$$

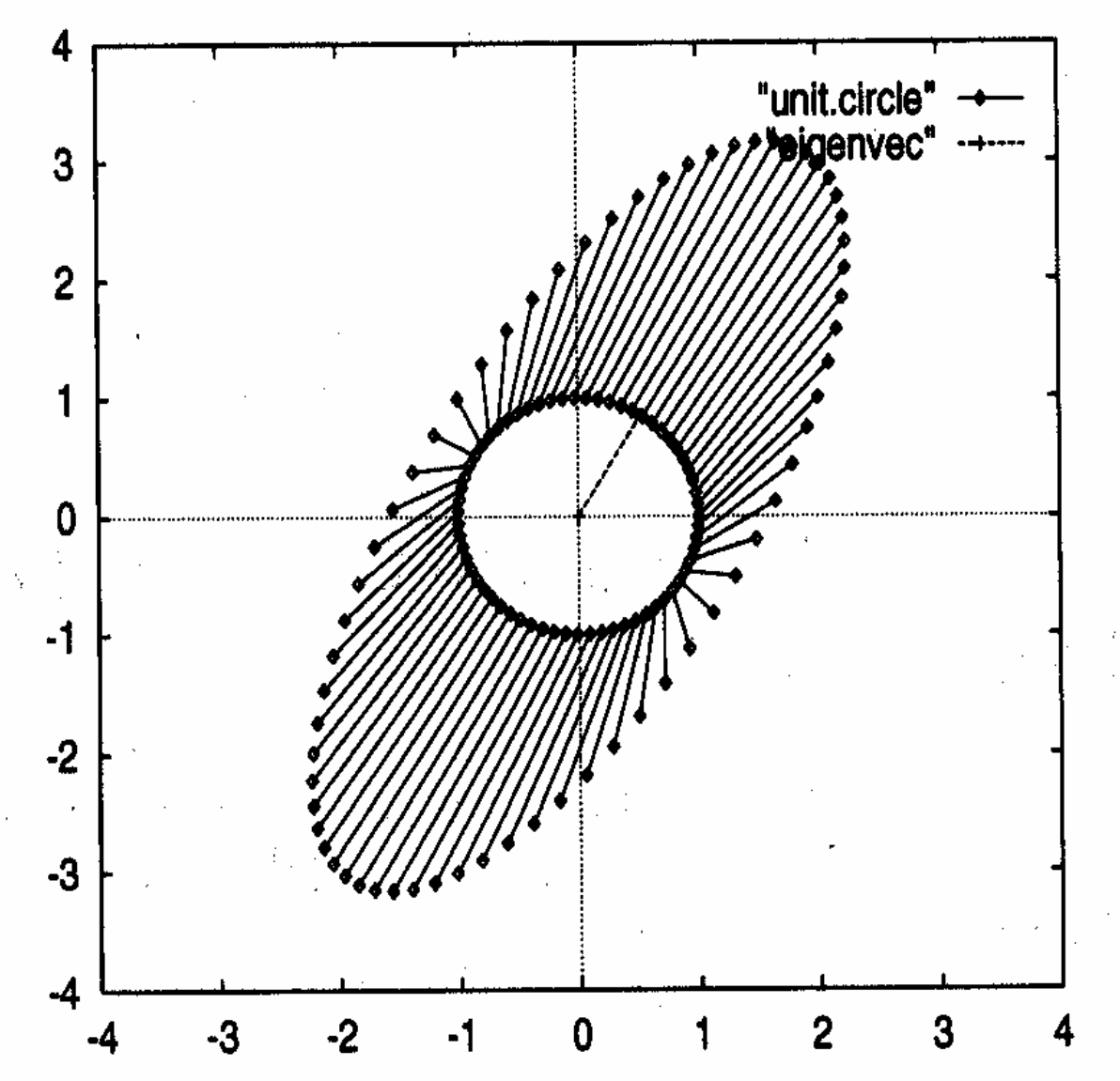
where matrix $A^{(ij)}$ is derived from A by removing the i-th row and the j-th column

The real-valued n×n matrix A is **symmetric** if $a_{ij}=a_{ji}$ for all i, j.
A is **positive definite** if for all n×1 vectors $x \neq \vec{0}$: $x^T \times A \times x > 0$.
If A is symmetric then all Eigenvalues of A are A real.
If A is symmetric and positive definite then all Eigenvalues are positive.

Illustration of Eigenvectors

Matrix $A = \begin{pmatrix} 2 & 1 \\ 1 & 3 \end{pmatrix}$

describes
affine transformation
$x \mapsto Ax$



Eigenvector
x1 = $(0.52 \;\; 0.85)^T$
for Eigenvalue $\lambda 1 = 3.62$

Eigenvector
x2 = $(0.85 \;\; -0.52)^T$
for Eigenvalue $\lambda 2 = 1.38$

Principal Component Analysis (PCA)

Spectral Theorem:

(PCA, Karhunen-Loewe transform):

Let A be a symmetric n×n matrix with Eigenvalues λ1, ..., λn and Eigenvectors x1, ..., xn such that $\|x_i\|_2 = 1$ for all i.
The Eigenvectors form an orthonormal basis of A.
Then the following holds:

$$D = Q^T \times A \times Q,$$

where D is a diagonal matrix with diagonal elements λ1, ..., λn and Q consists of column vectors x1, ..., xn.

often applied to covariance matrix of n-dim. data points

Singular Value Decomposition (SVD)

Theorem:

Each real-valued m×n matrix A with rank r can be decomposed into the form $\mathbf{A = U \times \Delta \times V^T}$
with an **m×r matrix U** with orthonormal column vectors,
an **r×r diagonal matrix Δ**, and
an **n×r matrix V** with orthonormal column vectors.
This decomposition is called singular value decomposition and is unique when the elements of Δ or sorted.

Theorem:

In the singular value decomposition $A = U \times \Delta \times V^T$ of matrix A the matrices U, Δ, and V can be derived as follows:

- Δ consists of the singular values of A,
  i.e. the positive roots of the Eigenvalues of $A^T \times A$,
- the columns of U are the Eigenvectors of $A \times A^T$,
- the columns of V are the Eigenvectors of $A^T \times A$.

SVD for Regression

Theorem:

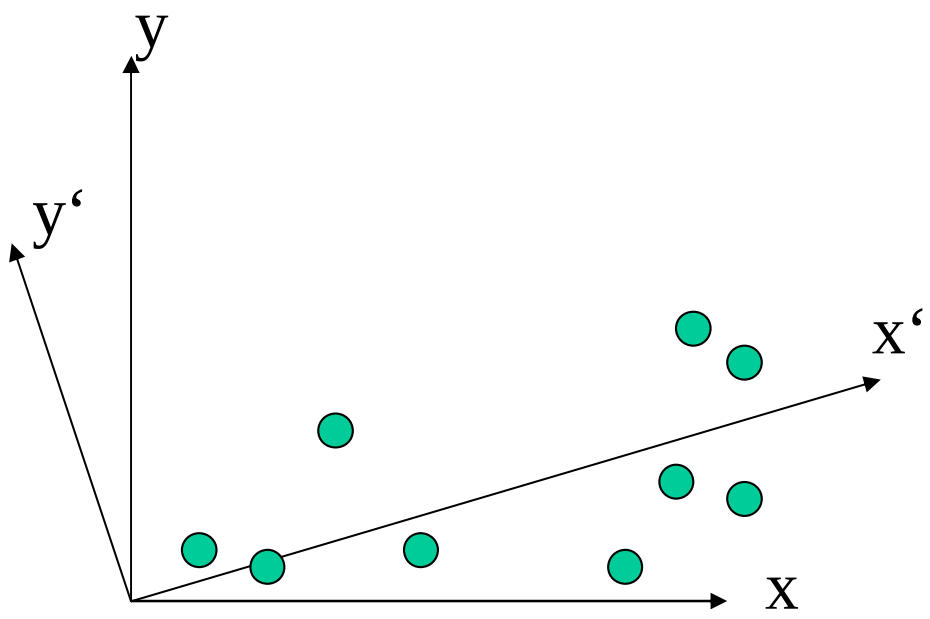
Let A be an m×n matrix with rank r, and let $\mathbf{A_k = U_k \times \Delta_k \times V_k^T}$, where the k×k diagonal matrix Δ_k contains the k largest singular values of A and the m×k matrix U_k and the n×k matrix V_k contain the corresponding Eigenvectors from the SVD of A.

Among all m×n matrices C with rank at most k

A_k is the matrix that minimizes the Frobenius norm

$$\|A-C\|_F^2 = \sum_{i=1}^{m}\sum_{j=1}^{n}(A_{ij}-C_{ij})^2$$

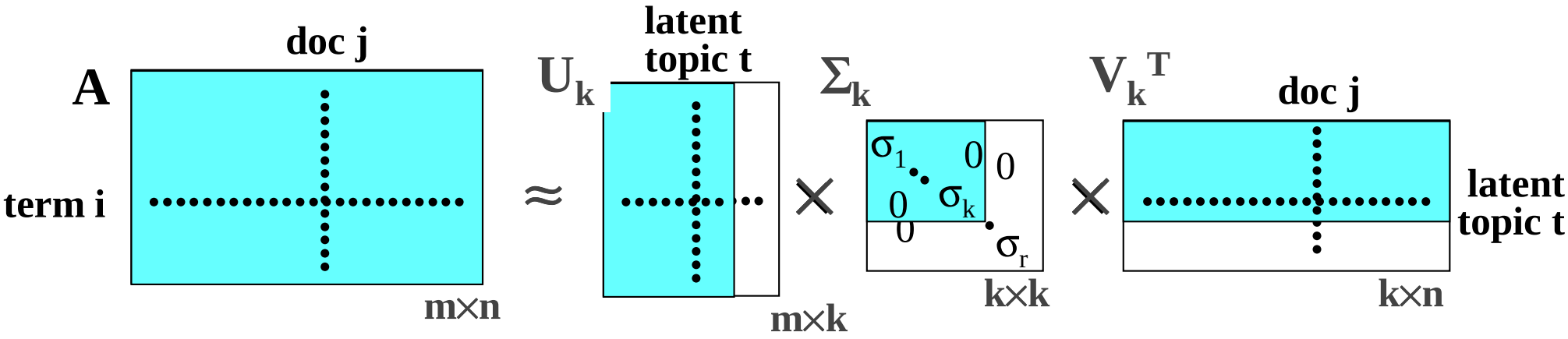
Example:
m=2, n=8, k=1
projection onto x‘ axis
minimizes „error“ or
maximizes „variance“
in k-dimensional space



4.3.2 Latent Semantic Indexing (LSI) [Deerwester et al. 1990]: Applying SVD to Vector Space Model

A is the m×n term-document similarity matrix. Then:

- U and U_k are the m×r and m×k term-topic similarity matrices,
- V and V_k are the n×r and n×k document-topic similarity matrices,
- $A \times A^T$ and $A_k \times A_k^T$ are the m×m term-term similarity matrices,
- $A^T \times A$ and $A_k^T \times A_k$ are the n×n document-document similarity matrices



mapping of m×1 vectors into latent-topic space: $d_j \mapsto U_k^T \times d_j =: d_j'$

$q \mapsto U_k^T \times q =: q'$

scalar-product similarity in latent-topic space: $d_j{}'^T \times q' = ((\Delta_k V_k^T)_{*j})^T \times q'$

Indexing and Query Processing

- The matrix $\Delta_k \, V_k^T$ corresponds to a **„topic index“** and
  is stored in a suitable data structure.
  Instead of $\Delta_k \, V_k^T$ the simpler **index V_k^T** could be used.
- Additionally the **term-topic mapping U_k** must be stored.
- A **query q** (an m×1 column vector) in the term vector space
  is transformed into query **$q' = U_k^T \times q$** (a k×1 column vector)
  and evaluated in the topic vector space (i.e. V_k)
  (e.g. by scalar-product similarity $V_k^T \times q'$ or cosine similarity)
- A **new document d** (an m×1 column vector) is transformed into
  $d' = U_k^T \times d$ (a k ×1 column vector) and
  appended to the „index“ V_k^T as an additional column **(„folding-in“)**

Example 1 for Latent Semantic Indexing

m=5 (interface, library, Java, Kona, blend), n=7

$$A = \begin{pmatrix} 1&2&1&5&0&0&0 \\ 1&2&1&5&0&0&0 \\ 1&2&1&5&0&0&0 \\ 0&0&0&0&2&3&1 \\ 0&0&0&0&2&3&1 \end{pmatrix} = \underbrace{\begin{pmatrix} 0.58 & 0.00 \\ 0.58 & 0.00 \\ 0.58 & 0.00 \\ 0.00 & 0.71 \\ 0.00 & 0.71 \end{pmatrix}}_{U} \times \underbrace{\begin{pmatrix} 9.64 & 0.00 \\ 0.00 & 5.29 \end{pmatrix}}_{\Delta} \times \underbrace{\begin{pmatrix} 0.18&0.36&0.18&0.90&0.00&0.00&0.00 \\ 0.00&0.00&0.00&0.00&0.53&0.80&0.27 \end{pmatrix}}_{V^T}$$

query $q = (0\ 0\ 1\ 0\ 0)^T$ is transformed into
$q' = U^T \times q = (0.58\ \ 0.00)^T$ and evaluated on V^T

the new document $d8 = (1\ 1\ 0\ 0\ 0)^T$ is transformed into
$d8' = U^T \times d8 = (1.16\ \ 0.00)^T$ and appended to V^T

Example 2 for Latent Semantic Indexing

m=6 terms
- t1: bak(e,ing)
- t2: recipe(s)
- t3: bread
- t4: cake
- t5: pastr(y,ies)
- t6: pie

n=5 documents
- d1: How to bake bread without recipes
- d2: The classic art of Viennese Pastry
- d3: Numerical recipes: the art of scientific computing
- d4: Breads, pastries, pies and cakes: quantity baking recipes
- d5: Pastry: a book of best French recipes

$$A = \begin{pmatrix} 0.5774 & 0.0000 & 0.0000 & 0.4082 & 0.0000 \\ 0.5774 & 0.0000 & 1.0000 & 0.4082 & 0.7071 \\ 0.5774 & 0.0000 & 0.0000 & 0.4082 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.4082 & 0.0000 \\ 0.0000 & 1.0000 & 0.0000 & 0.4082 & 0.7071 \\ 0.0000 & 0.0000 & 0.0000 & 0.4082 & 0.0000 \end{pmatrix}$$

Example 2 for Latent Semantic Indexing (2)

$$A = \begin{pmatrix} 0.2670 & -0.2567 & 0.5308 & -0.2847 \\ 0.7479 & -0.3981 & -0.5249 & 0.0816 \\ 0.2670 & -0.2567 & 0.5308 & -0.2847 \\ 0.1182 & -0.0127 & 0.2774 & 0.6394 \\ 0.5198 & 0.8423 & 0.0838 & -0.1158 \\ 0.1182 & -0.0127 & 0.2774 & 0.6394 \end{pmatrix} \quad \text{U}$$

$$\times \begin{pmatrix} 1.6950 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 1.1158 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.8403 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.4195 \end{pmatrix} \quad \Delta$$

$$\times \begin{pmatrix} 0.4366 & 0.3067 & 0.4412 & 0.4909 & 0.5288 \\ -0.4717 & 0.7549 & -0.3568 & -0.0346 & 0.2815 \\ 0.3688 & 0.0998 & -0.6247 & 0.5711 & -0.3712 \\ -0.6715 & -0.2760 & 0.1945 & 0.6571 & -0.0577 \end{pmatrix} \quad \text{V}^{\text{T}}$$

Example 2 for Latent Semantic Indexing (3)

$$A_3 = \begin{pmatrix} 0.4971 & -0.0330 & 0.0232 & 0.4867 & -0.0069 \\ 0.6003 & 0.0094 & 0.9933 & 0.3858 & 0.7091 \\ 0.4971 & -0.0330 & 0.0232 & 0.4867 & -0.0069 \\ 0.1801 & 0.0740 & -0.0522 & 0.2320 & 0.0155 \\ -0.0326 & 0.9866 & 0.0094 & 0.4402 & 0.7043 \\ 0.1801 & 0.0740 & -0.0522 & 0.2320 & 0.0155 \end{pmatrix} = U_3 \times \Delta_3 \times V_3^T$$

Example 2 for Latent Semantic Indexing (4)

query q: baking bread
$q = (1\ 0\ 1\ 0\ 0\ 0)^T$

transformation into topic space with k=3
$q' = U_k^T \times q = (0.5340\ \ -0.5134\ \ 1.0616)^T$

scalar product similarity in topic space with k=3:
$sim(q, d1) = Vk_{*1}^T \times q' \approx 0.86$ $sim(q, d2) = Vk_{*2}^T \times q \approx -0.12$
$sim(q, d3) = Vk_{*3}^T \times q' \approx -0.24$ etc.

Folding-in of a new document d6:
algorithmic recipes for the computation of pie
$d6 = (0\ \ 0.7071\ \ 0\ \ 0\ \ 0\ \ 0.7071)^T$

transformation into topic space with k=3
$d6' = U_k^T \times d6 \approx (0.5\ \ -0.28\ \ -0.15)$

d6‘ is appended to Vk^T as a new column

Multilingual Retrieval with LSI

- Construct LSI model (U_k, Δ_k, V_k^T) from training documents that are available in multiple languages:
  - consider all language variants of the same document as a single document and
  - extract all terms or words for all languages.
- Maintain index for further documents by „folding-in“, i.e. mapping into topic space and appending to V_k^T.
- Queries can now be asked in any language, and the query results include documents from all languages.

Example:

d1: How to bake bread without recipes.
Wie man ohne Rezept Brot backen kann.

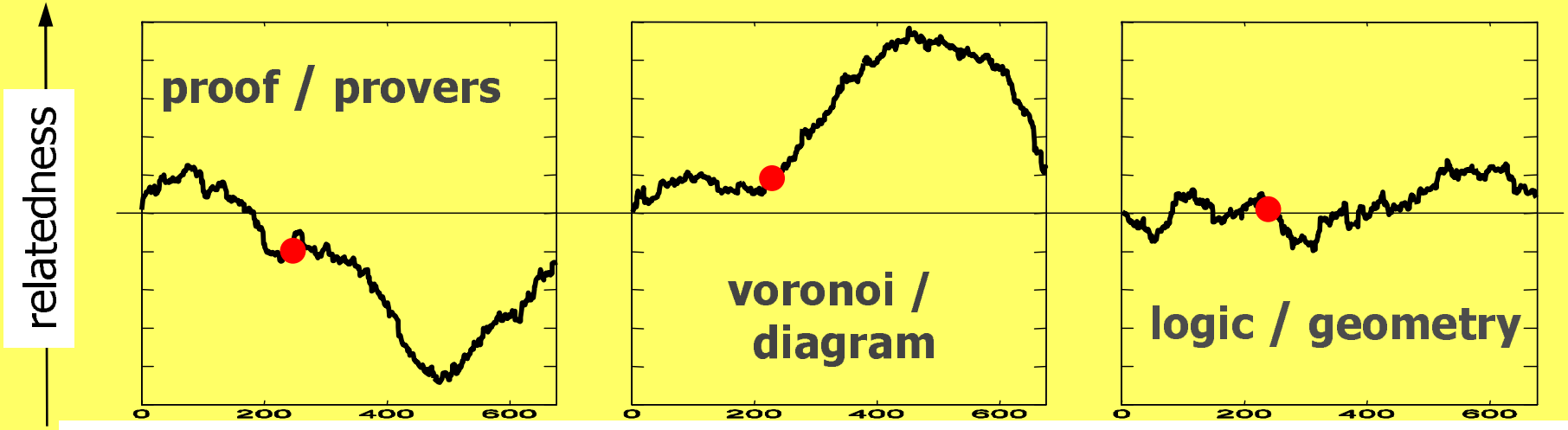
d2: Pastry: a book of best French recipes.
Gebäck: eine Sammlung der besten französischen Rezepte.

Terms are e.g. bake, bread, recipe, backen, Brot, Rezept, etc.

Documents and terms are mapped into compact topic space.

Towards Self-tuning LSI [Bast et al. 2005]

- Project data to its top k eigenvectors (SVD): $A \approx U_k \times \Sigma_k \times V^T_k$
  → latent concepts (LSI)
- This discovers hidden term relations in $U_k \times U_k^T$:
  - proof / provers: -0.68
  - voronoi / diagram: 0.73
  - logic / geometry: -0.12
- Central question: which k is the best?



Assess the shape of the graph, not specific values!

→ new „dimension-less“ variant of LSI:
use 0-1-rounded expansion matrix $U_k \times U_k^T$ to expand docs
→ outperforms standard LSI

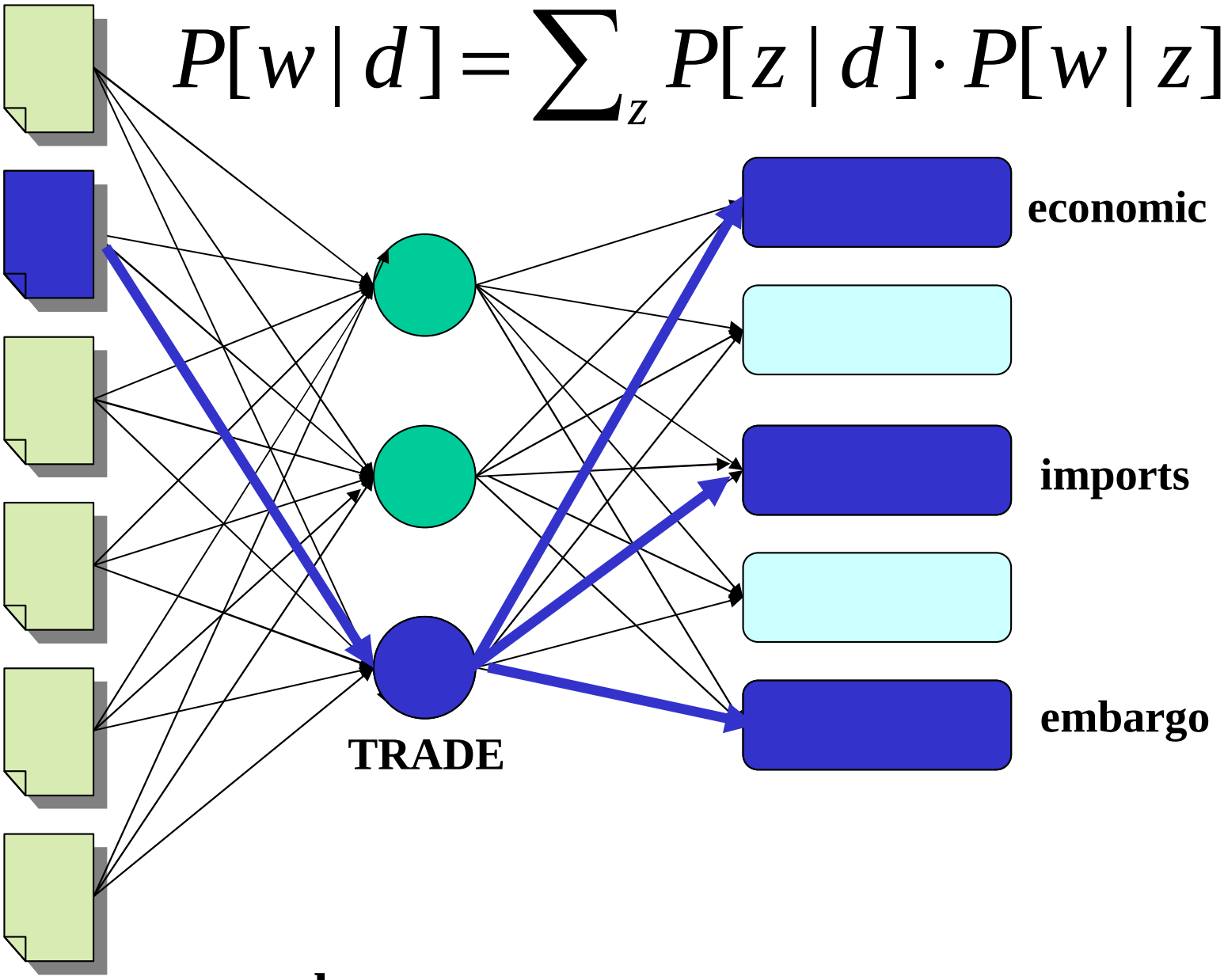
Summary of LSI

+ Elegant, mathematically well-founded model
+ „Automatic learning“ of term correlations
  (incl. morphological variants, multilingual corpus)
+ Implicit thesaurus (by correlations between synonyms)
+ Implicit discrimination of different meanings of polysems
  (by different term correlations)
+ Improved precision and recall on „closed“ corpora
  (e.g. TREC benchmark, financial news, patent databases, etc.)
  with empirically best k in the order of 100-200
– In general difficult choice of appropriate k
– Computational and storage overhead for very large (sparse) matrices
– No convincing results for Web search engines (yet)

4.3.3 Probabilistic LSI (pLSI)

$$P[w \mid d] = \sum_z P[z \mid d] \cdot P[w \mid z]$$

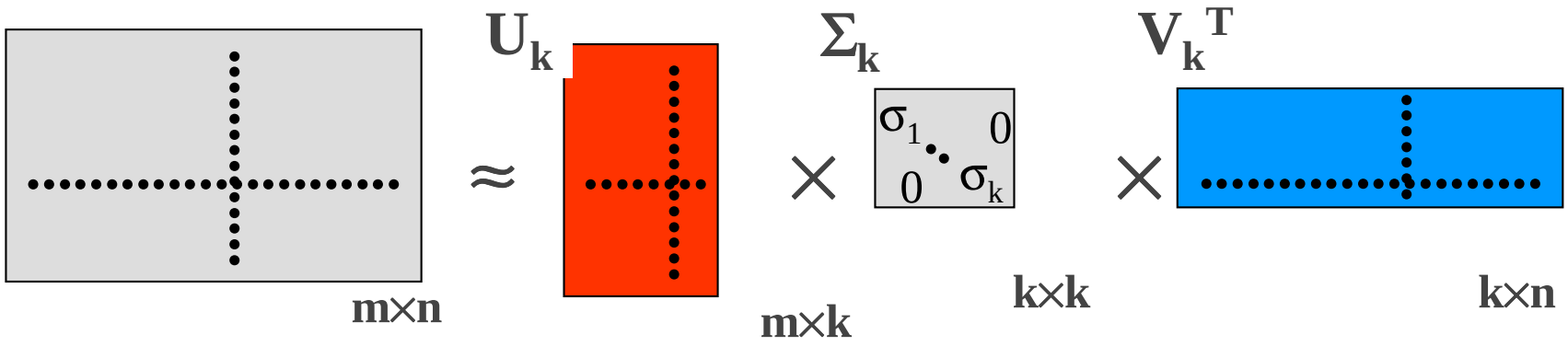
d and w conditionally independent given z



documents d **latent concepts z (aspects)** **terms w (words)**

Relationship of pLSI to LSI

$$P[d,w] = \sum_z P[d|z] \cdot P[z] \cdot P[w|z]$$

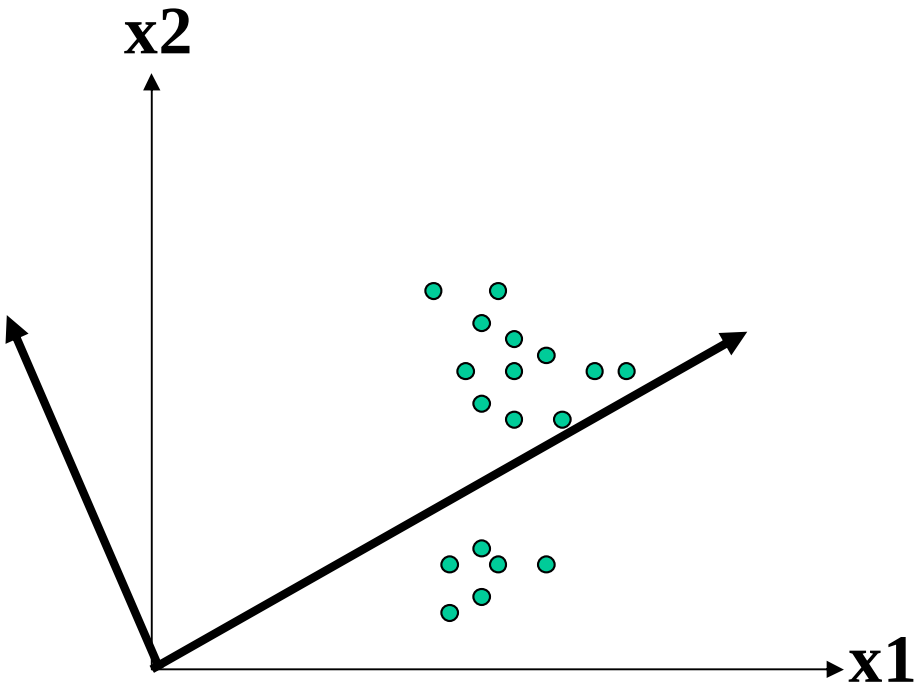


doc probs per concept | **concept probs** | **term probs per concept**

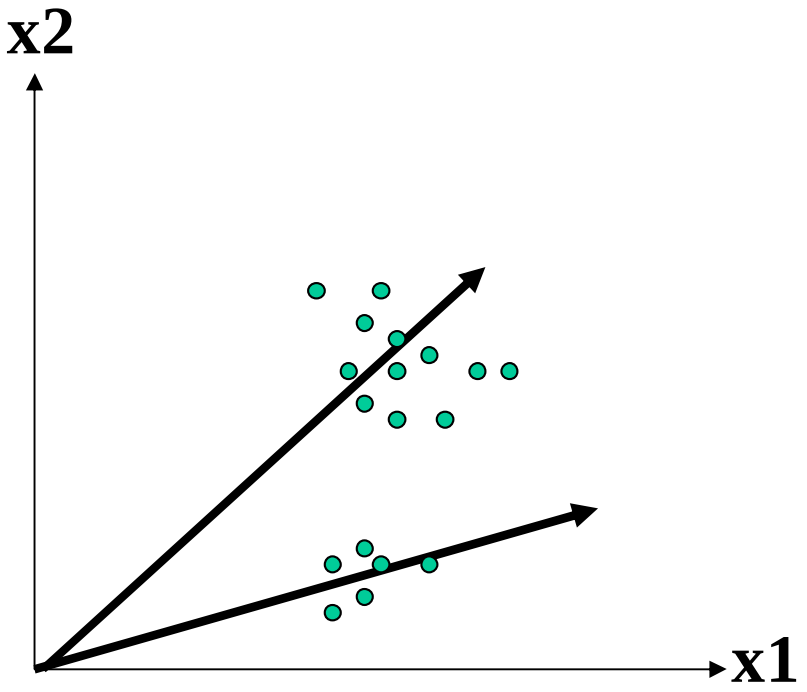
Key difference to LSI:
- **non-negative matrix decomposition**
- **with L1 normalization**

Key difference to LMs:
- **no generative model for docs**
- **tied to given corpus**

Power of Non-negative Matrix Factorization vs. SVD



SVD of data matrix A



NMF of data matrix A

Expectation-Maximization Method (EM)

Key idea:

when $L(\theta, X_1, ..., X_n)$ (where the X_i and θ are possibly multivariate)

is analytically intractable then

- introduce ***latent (hidden, invisible, missing) random variable(s) Z***

  such that

  - the ***joint distribution $J(X_1, ..., X_n, Z, \theta)$ of the „complete" data***

    is tractable (often with Z actually being $Z_1, ..., Z_n$)

- derive the incomplete-data likelihood $L(\theta, X_1, ..., X_n)$ by

  integrating (marginalization) J:

$$\hat{\theta} = arg\,max_{\theta} \sum_{z} J[\theta, X_1, ..., X_n, Z \mid Z = z] P[Z = z]$$

EM Procedure

Initialization: choose start estimate for $\theta^{(0)}$

Iterate (t=0, 1, …) until convergence:

E step (expectation):
estimate posterior probability of Z: $P[Z \mid X_1, \ldots, X_n, \theta^{(t)}]$
assuming θ were known and equal to previous estimate $\theta^{(t)}$,
and compute $E_{Z \mid X1, \ldots, Xn, \theta(t)}[\log J(X_1, \ldots, X_n, Z \mid \theta)]$
by integrating over values for Z

M step (maximization, MLE step):
Estimate $\theta^{(t+1)}$ by maximizing
$E_{Z \mid X1, \ldots, Xn, \theta(t)}[\log J(X_1, \ldots, X_n, Z \mid \theta)]$

convergence is guaranteed
(because the E step computes a lower bound of the true L function,
and the M step yields monotonically non-decreasing likelihood),
but may result in local maximum of log-likelihood function

EM at Indexing Time (pLSI Model Fitting)

observed data: **n(d,w) – absolute frequency of word w in doc d**

model params: **P[z|d], P[w|z] for concepts z, words w, docs d**

maximize log-likelihood $\sum_d \sum_w n(d,w) \cdot \log P[dw]$

E step: **posterior probability of latent variables**

$$P[z \mid d,w] = \frac{P[z \mid d]P[w \mid z]}{\sum_y P[y \mid d]P[w \mid y]}$$

prob. that occurrence of word w in doc d can be explained by concept z

M step: **MLE with completed data**

$$P[w \mid z] \sim \sum_d n(d,w)P[z \mid d,w] \qquad P[z \mid d] \sim \sum_w n(d,w)P[z \mid d,w]$$

freq. of w associated with z **freq. of d associated with z**

actual procedure „perturbs“ EM for „smoothing“ (avoidance of overfitting) → tempered annealing

EM Details (pLSI Model Fitting)

$$P[z \mid d,w] = \frac{P[z \mid d]P[w \mid z]}{\sum_y P[y \mid d]P[w \mid y]} \qquad \textbf{(E)}$$

$$P[w \mid z] = \frac{\sum_d n(d,w)P[z \mid d,w]}{\sum_{d,u} n(d,u)P[z \mid d,u]} \qquad \textbf{(M1)}$$

$$P[z \mid d] = \frac{\sum_w n(d,w)P[z \mid d,w]}{\sum_{w,y} n(d,w)P[y \mid d,w]} \qquad \textbf{(M2)}$$

or equivalently compute P[z], P[d|z], P[w|z] in M step (see S. Chakrabarti, pp. 110/111)

Folding-in of Queries

keep all estimated parameters of the pLSI model fixed

and treat query as a „new document“ to be explained

→ find concepts that most likely generate the query

(query is the only „document“, and P[w | z] is kept invariant)

→ EM for query parameters

$$P[z \mid q, w] = \frac{P[z \mid q]\hat{p}[w \mid z]}{\sum_y P[y \mid q]\hat{p}[w \mid y]}$$

$$P[z \mid q] = \frac{\sum_w n(q,w)P[z \mid q,w]}{\sum_{w,y} n(q,w)P[y \mid q,w]}$$

Query Processing

Once documents and queries are both represented as
probability distributions over k concepts
(i.e. k×1 vectors with L1 length 1),
we can use any convenient vector-space similarity measure
(e.g. scalar product or cosine or KL divergence).

Experimental Results: Example

- Concepts (10 of 128) extracted from Science Magazine articles (12K)

P(w|z)

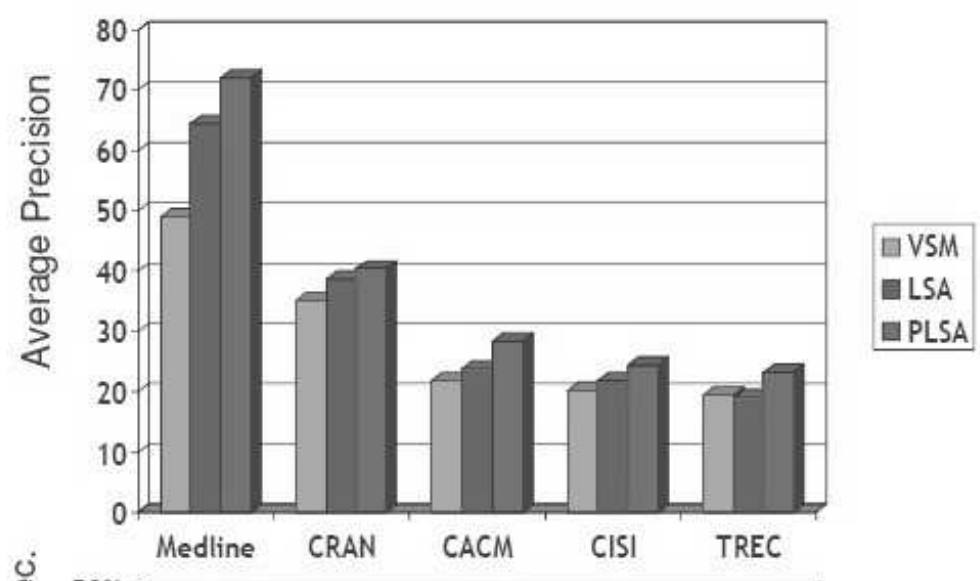
universe	0.0439	drug	0.0672	cells	0.0675	sequence	0.0818	years	0.156
galaxies	0.0375	patients	0.0493	stem	0.0478	sequences	0.0493	million	0.0556
clusters	0.0279	drugs	0.0444	human	0.0421	genome	0.033	ago	0.045
matter	0.0233	clinical	0.0346	cell	0.0309	dna	0.0257	time	0.0317
galaxy	0.0232	treatment	0.028	gene	0.025	sequencing	0.0172	age	0.0243
cluster	0.0214	trials	0.0277	tissue	0.0185	map	0.0123	year	0.024
cosmic	0.0137	therapy	0.0213	cloning	0.0169	genes	0.0122	record	0.0238
dark	0.0131	trial	0.0164	transfer	0.0155	chromosome	0.0119	early	0.0233
light	0.0109	disease	0.0157	blood	0.0113	regions	0.0119	billion	0.0177
density	0.01	medical	0.00997	embryos	0.0111	human	0.0111	history	0.0148

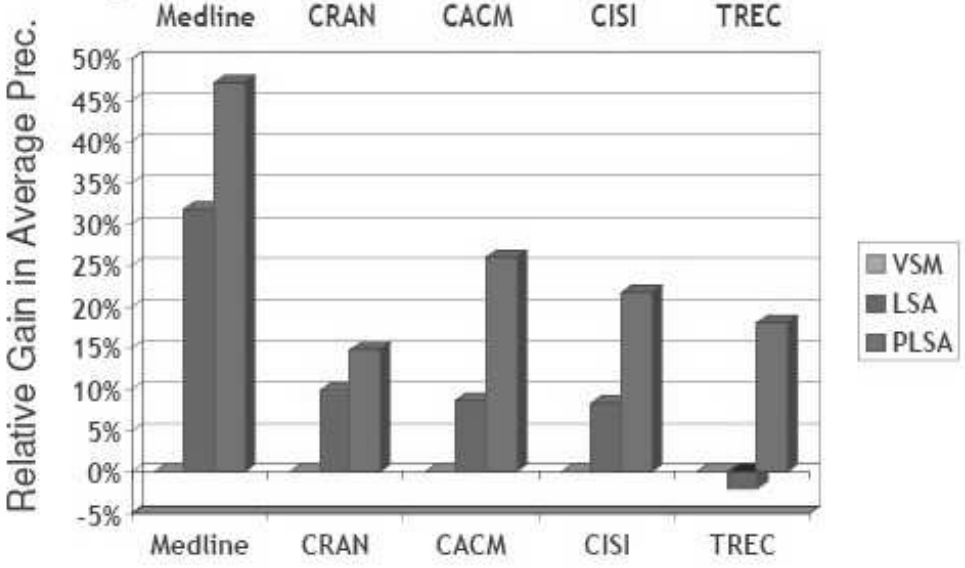
P(w|z)

bacteria	0.0983	male	0.0558	theory	0.0811	immune	0.0909	stars	0.0524
bacterial	0.0561	females	0.0541	physics	0.0782	response	0.0375	star	0.0458
resistance	0.0431	female	0.0529	physicists	0.0146	system	0.0358	astrophys	0.0237
coli	0.0381	males	0.0477	einstein	0.0142	responses	0.0322	mass	0.021
strains	0.025	sex	0.0339	university	0.013	antigen	0.0263	disk	0.0173
microbiol	0.0214	reproductive	0.0172	gravity	0.013	antigens	0.0184	black	0.0161
microbial	0.0196	offspring	0.0168	black	0.0127	immunity	0.0176	gas	0.0149
strain	0.0165	sexual	0.0166	theories	0.01	immunology	0.0145	stellar	0.0127
salmonella	0.0163	reproduction	0.0143	aps	0.00987	antibody	0.014	astron	0.0125
resistant	0.0145	eggs	0.0138	matter	0.00954	autoimmune	0.0128	hole	0.00824

Source: Thomas Hofmann, Tutorial at ADFOCS 2004

Experimental Results: Precision





Summary of quantitative evaluation

- Consistent improvements of retrieval accuracy
- Relative improvements of average precision **15-45%**
- On **TREC3**: **18%** improvement compared to SMART retrieval metric

VSM: simple tf-based vector space model (no idf)

Source: Thomas Hofmann, Tutorial „Machine Learning in Information Retrieval“, presented at Machine Learning Summer School (MLSS) 2004, Berder Island, France

Experimental Results: Perplexity

Perplexity measure (reflects generalization potential, as opposed to overfitting): $2^{H(freq(w,d),P[w|d])} = 2^{-\sum_{w,d} freq(w,d)\cdot \log_2 P[w|d]}$

with freq on new data

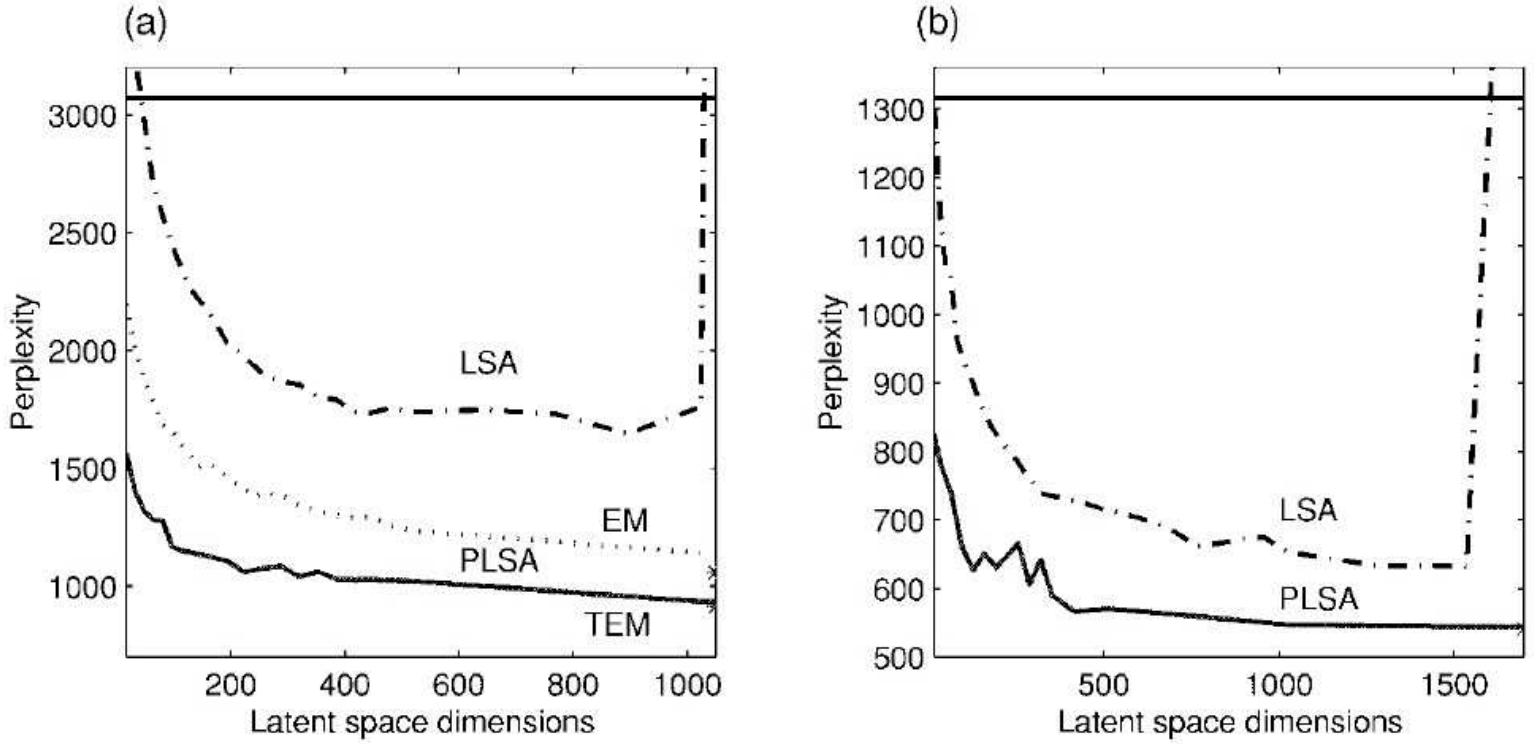


Figure 6. Perplexity results as a function of the latent space dimensionality for (a) the MED data (rank 1033) and (b) the LOB data (rank 1674). Plotted results are for LSA (dashed-dotted curve) and PLSA (trained by TEM = solid curve, trained by early stopping EM = dotted curve). The upper baseline is the unigram model corresponding to marginal independence. The star at the right end of the PLSA denotes the perplexity of the largest trained aspect models ($K = 2048$).

Source: T. Hofmann, Machine Learning 42 (2001)

pLSI Summary

+ Probabilistic variant of LSI
  (non-negative matrix factorization with L1 normalization)
+ Achieves better experimental results than LSI
+ Very good on „closed“, thematically specialized corpora,
  inappropriate for Web
– Computationally expensive (at indexing and querying time)
  → may use faster clustering for estimating P[d|z] instead of EM
  → may exploit sparseness of query to speed up folding-in
– pLSI does not have a generative model (rather tied to fixed corpus)
  → LDA model (Latent Dirichlet Allocation)
– number of latent concept remains model-selection problem
  → compute for different k, assess on held-out data, choose best

Additional Literature for Chapter 4

Latent Semantic Indexing:

- Grossman/Frieder Section 2.6
- Manning/Schütze Section 15.4
- M.W. Berry, S.T. Dumais, G.W. O'Brien: Using Linear Algebra for Intelligent Information Retrieval, SIAM Review Vol.37 No.4, 1995
- S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, R. Harshman: Indexing by Latent Semantic Analysis, JASIS 41(6), 1990
- H. Bast, D. Majumdar: Why Spectral Retrieval Works, SIGIR 2005
- W.H. Press: Numerical Recipes in C, Cambridge University Press, 1993, available online at http://www.nr.com/
- G.H. Golub, C.F. Van Loan: Matrix Computations, John Hopkins University Press, 1996

pLSI and Other Latent-Concept Models:

- Chakrabarti Section 4.4.4
- T. Hofmann: Unsupervised Learning by Probabilistic Latent Semantic Analysis, Machine Learning 42, 2001
- T. Hofmann: Matrix Decomposition Techniques in Machine Learning and Information Retrieval, Tutorial Slides, ADFOCS 2004
- D. Blei, A. Ng, M. Jordan: Latent Dirichlet Allocation, Journal of Machine Learning Research 3, 2003
- W. Xu, X. Liu, Y. Gong: Document Clustering based on Non-negative Matrix Factorization, SIGIR 2003