3 Alternative IR-Modelle

- **Probabilistisches Retrieval:**
  Ranking aufgrund von Relevanzwahrscheinlichkeiten,
  die aus - geschätzten - Basisparametern abgeleitet werden.
- **Fuzzy-Set-Modell:**
  Queries (inkl. einzelner Terme) beschreiben Fuzzy-Mengen
  mit Dokumenten als Elementen vom Grad $\mu \in [0,1]$.
  Mengenoperationen verwenden Funktionen max, min, 1-μ.
- **Latent Semantic Indexing:**
  Berücksichtigung von Termkorrelationen durch
  Transformation des Term-Vektorraums in einen
  Themen-Vektorraum niedrigerer Dimensionalität
- **Neuronale Netze** und andere Inferenznetze
  zum Lernen von Termgewichten

Exkurs: Wahrscheinlichkeitstheorie

Ein **Wahrscheinlichkeitsraum** ist ein Tripel (Ω, E, P) mit
- einer Menge Ω elementarer Ereignisse,
- einer Familie E von Teilmengen von Ω mit $\Omega \in E$, die unter $\cap$, $\cup$ und – mit abzählbar vielen Operanden abgeschlossen ist (bei endlichem Ω ist in der Regel $E=2^{\Omega}$), und
- einem W.maß $P: E \rightarrow [0,1]$ mit $P[\Omega]=1$ und $P[\cup_i Ai] = \sum_i P[Ai]$ für abzählbar viele, paarweise disjunkte Ai

Eigenschaften von P:

$P[A] + P[\neg A] = 1$ $\qquad$ $P[\emptyset] = 0$

$P[A \cup B] = P[A] + P[B] - P[A \cap B]$ $\qquad$ $P[\Omega] = 1$

Eine **Zufallsvariable** X über einem W.raum (Ω, E, P) ist eine Funktion $X: \Omega \rightarrow M$ mit $M \subseteq R$, so daß $\{e \mid X(e) \leq x\} \in E$ für alle $x \in M$.

$F_X: M \rightarrow [0,1]$ mit $F_X(x) = P[X \leq x]$ heißt Verteilungsfunktion von X; bei abzählbarer Menge M heißt $f_X: M \rightarrow [0,1]$ mit $f_X(x) = P[X = x]$ Dichtefunktion von X, ansonsten ist $f_X(x)$ durch $F'_X(x)$ gegeben

Exkurs: Satz von Bayes

Zwei Ereignisse A, B eines W.raums heißen **unabhängig**, wenn gilt $P[A \cap B] = P[A]\,P[B]$.

Die **bedingte Wahrscheinlichkeit P[A | B]** von A unter der Bedingung (Hypothese) B ist definiert als: $$P[A \mid B] = \frac{P[A \cap B]}{P[B]}$$

Satz von der totalen Wahrscheinlichkeit:
Für eine Partitionierung von Ω in Ereignisse B1, ..., Bn gilt:

$$P[A] = \sum_{i=1}^{n} P[A \mid Bi]\,P[Bi]$$

Satz von Bayes: $$P[A \mid B] = \frac{P[B \mid A]\,P[A]}{P[B]}$$

A-Posteriori-W. von A

Probabilistisches Retrieval: Grundsätze

Ziel:

Ranking aufgrund von sim(d, q) =
P[R|d] = P [Dokument d ist für Query q relevant |
d hat den Termvektor X1, ..., Xm]

Annahmen:

- Relevante und nichtrelevante Dokumenten unterscheiden sich in ihren Termen
- Binary Independence Retrieval (BIR) Model:
  - Die Wahrscheinlichkeiten des Auftretens eines Terms sind paarweise unabhängig.
  - Termgewichte sind binär $\in \{0,1\}$.
- Für Terme, die nicht in q vorkommen, sind die Wahrscheinlichkeiten des Auftretens in relevanten und in irrelevanten Dokumenten identisch.

Probabilistisches Retrieval: Ranking proportional zur Relevanz-Quote

$$sim(d,q) = O(R \mid d) = \frac{P[R \mid d]}{P[\neg R \mid d]}$$ (Quote („Odds“) für Relevanz)

$$= \frac{P[d \mid R] \times P[R]}{P[d \mid \neg R] \times P[\neg R]}$$ (Satz von Bayes)

$$\sim \frac{P[d \mid R]}{P[d \mid \neg R]} \quad = \prod_i \frac{P[X_i \mid R]}{P[X_i \mid \neg R]}$$ (independence bzw. linked dependence)

$$sim(d,q)' = \log \prod_{i \in q} \frac{P[Xi \mid R]}{P[Xi \mid \neg R]}$$ (Xi = 1, falls d den i-ten Term enthält, 0 sonst)

$$= \sum_i \log P[Xi \mid R] - \log P[Xi \mid \neg R]$$

Probabilistisches Retrieval: Ranking proportional zur Relevanz-Quote (Forts.)

$$= \sum_i \log\left(pi^{Xi}(1-pi)^{1-Xi}\right) - \log\left(qi^{Xi}(1-qi)^{1-Xi}\right)$$

(binary features)

mit pi=P[Xi=1|R] und qi=P[Xi=1|¬R]

$$= \sum_i \log\left(\frac{pi^{Xi}(1-pi)}{(1-pi)^{Xi}}\right) - \log\left(\frac{qi^{Xi}(1-qi)}{(1-qi)^{Xi}}\right)$$

$$= \sum_i Xi \log\frac{pi}{1-pi} + \sum_i Xi \log\frac{1-qi}{qi} + \sum_i \log\frac{1-pi}{1-qi}$$

$$\sim \sum_i Xi \log\frac{pi}{1-pi} + \sum_i Xi \log\frac{1-qi}{qi} = sim(d,q)''$$

Probabilistisches Retrieval: Formel von Robertson und Sparck Jones

Schätze pi und qi auf der Basis einer Trainingsstichprobe (Anfrage q auf kleinem Trainingskorpus) bzw. der intellektuellen Bewertung eines ersten (groben) Resultats (Relevanz-Feedback):

Sei V die Kardinalität einer Stichprobe relevanter Dokumente
Vi die Anzahl der Dok. in V, die den i-ten Term enthalten
ni eine Schätzung für die Anz. der Dok, die den i-ten Term enthalten

$\Rightarrow$ Schätzung: $pi = \frac{Vi}{V} \qquad qi = \frac{ni - Vi}{N - V}$

bzw.: $pi = \frac{Vi + 0.5}{V + 1} \qquad qi = \frac{ni - Vi + 0.5}{N - V + 1}$

$\Rightarrow sim(d,q)'' = \sum_i Xi \log \frac{Vi + 0.5}{V - Vi + 0.5} + \sum_i Xi \log \frac{N - ni - V + Vi + 0.5}{ni - Vi + 0.5}$

$\Rightarrow$ Gewicht von Term i in d: $\log \frac{(Vi + 0.5)\ (N - ni - V + Vi + 0.5)}{(V - Vi + 0.5)(ni - Vi + 0.5)}$

Probabilistisches Retrieval: Bezug zur tf-idf-Formel

Annahmen (ohne Trainingsstichprobe oder Relevanz-Feedback):

- pi ist identisch für alle i
- Die meisten Dokumenten sind irrelevant.
- Jeder einzelne Term kommt selten vor.

Daraus folgt:

- $\sum_i Xi \log \frac{pi}{1-pi} = c \sum_i Xi$ mit Konstante c
- $qi = P[Xi = 1 | \neg R] \approx \frac{df_i}{N}$
- $\frac{1-qi}{qi} = \frac{N - df_i}{df_i} \approx \frac{N}{df_i}$

$$\Rightarrow \quad sim(d,q)'' = \sum_i Xi \log \frac{pi}{1-pi} + \sum_i Xi \log \frac{1-qi}{qi}$$

$$\approx c \sum_i Xi + \sum_i Xi \, idf_i$$

Skalarprodukt des Produkts von tf- und gedämpften idf-Werten über die Anfrageterme

Beispiel für Probabilistisches Retrieval

Dokumente mit Relevanz-Feedback:

q: t1 t2 t3 t4 t5 t6

	t1	t2	t3	t4	t5	t6	R
d1	1	0	1	1	0	0	1
d2	1	1	0	1	1	0	1
d3	0	0	0	1	1	0	0
d4	0	0	1	0	0	0	0
ni	2	1	2	3	2	0	
Vi	2	1	1	2	1	0	
pi	5/6	1/2	1/2	5/6	1/2	1/6	
qi	1/6	1/6	1/2	1/2	1/2	1/6	

V=2, N=4

Bewertung (Retrieval Status Value) eines neuen Dokuments:

d5∩q: 1 1 0 0 0 1 $\rightarrow$sim(d5, q) = log5 + log1 + log0.2 + log5 + log5 + log5

$$sim(d,q)'' = \sum_i Xi \log \frac{pi}{1-pi} + \sum_i Xi \log \frac{1-qi}{qi}$$

Ansätze zur Erweiterung des Probabilistischen Retrievals

1) Berücksichtigung von Termhäufigkeiten in Dok. (nichtbinäre Xi)
   → benötigt Annahmen über Termverteilung wie z.B.

   a) Gleichverteilung:
   P[zufällig im Korpus gezogener Term = t_i] = 1/m =: p_i

   $$\Rightarrow P[tf_{ij} = k \mid d_j \text{ hat } l \text{ Wörter}] = \binom{l}{k} p_i^{\,k} (1 - p_i)^{l-k}$$ *Binomial-verteilung*

   b) Zipf-Verteilung:
   P[zufällig im Korpus gezogener Term = t_i] = (1/i) / H_m =: p_i

   mit $H_m = \sum_{j=1}^{m} \frac{1}{j}$

   c) 2-(Klassen-)Poisson-Verteilung:

   $$P[tf_{ij} = k] = p_{i1} e^{-l_{i1}} \frac{l_{i1}^{\,k}}{k!} + p_{i2} e^{-l_{i2}} \frac{l_{i2}^{\,k}}{k!}$$ mit Konstanten π_{i1} und π_{i2}

Ansätze zur Erweiterung des Probabilistischen Retrievals

2) Berücksichtigung von Termkorrelationen in Dok. (mit binären Xi)
   → Problem der Schätzung m-dimensionaler W.verteilungen
   $P[X1=... \wedge X2= ... \wedge ... \wedge Xm=...] =: f_X(X1, ..., Xm)$

Ein Ansatz **– Tree Dependence Model**:

a) Betrachte nur 2-dimensionale Wahrscheinlichkeiten (für Termpaare)
   $f_{ij}(Xi, Xj)=P[Xi=..\wedge Xj=..]= \sum_{X_1} .. \sum_{X_{i-1}} \sum_{X_{i+1}} .. \sum_{X_{j-1}} \sum_{X_{j+1}} .. \sum_{X_m} P[X_1 = ... \wedge .. \wedge X_m = ...]$

b) Für jedes Termpaar
   schätze den Fehler der Unabhängigkeitsannahme bzw. die Korrelation

c) Konstruiere einen Baum mit Termen als Knoten und den
   m-1 höchsten Korrelationen als gewichteten Kanten

Bewertung zweidimensionaler Termkorrelationen (2b)

Variante 1:
Fehler der Approximation von f durch g
(insbesondere g mit paarweiser Termunabhängigkeit):

$$e(f,g) := \sum_{\vec{X}\in\{0,1\}^m} f(\vec{X}) \log \frac{f(\vec{X})}{g(\vec{X})} = \sum_{\vec{X}\in\{0,1\}^m} f(\vec{X}) \log \frac{f(\vec{X})}{\prod_{i=1}^{m} g_i(X_i)}$$

Variante 2:
Korrelationskoeffizient von Termpaaren

$$r(Xi,Xj) := \frac{Cov(Xi,Xj)}{\sqrt{Var(Xi)}\sqrt{Var(Xj)}}$$

mit Covarianz bzw. Varianz:

$$Cov(Xi,Xj) := E[(Xi - E[Xi])(Xj - E[Xj])]$$

$$Var(Xi) := E[(Xi - E[Xi])^2] = Cov(Xi,Xi) = E[X^2] - E[X]^2$$

und Erwartungswert E[...]: $E[Xi] := \sum_{k=0}^{1} k\, f_{Xi}(k) = \sum_{k=0}^{1} k\, P[Xi = k]$

Beispiel für Approximationsfehler e

m=2:

Gegeben seien Dokumente

d1=(1,1), d2(0,0), d3=(1,1), d4=(0,1)

Schätzung der 2-dimensionalen W.verteilung f:

$f(1,1) = P[X1=1 \wedge X2=1] = 2/4$

$f(0,0) = 1/4$, $f(0,1) = 1/4$, $f(1,0) = 0$

Schätzung der 1-dimensionalen Randverteilung g1 und g2:

$g1(1) = P[X1=1] = 2/4$, $g1(0) = 2/4$

$g2(1) = P[X2=1] = 3/4$, $g2(0) = 1/4$

Schätzung für die 2-dim. W.verteilung g mit unabhängigen Xi:

$g(1,1) = g1(1)*g2(1) = 3/8$,

$g(0,0) = 1/8$, $g(0,1) = 3/8$, $g(1,0) = 1/8$

Approximationsfehler ε:

$\varepsilon = 2/4 \log 4/3 + 1/4 \log 2 + 1/4 \log 2/3 + 0$

Bewertung zweidimensionaler Termkorrelationen (2c)

Gegeben:

Vollständiger Graph (V, E) mit m Knoten Xi ∈ V und

m^2 – mit ε bzw. ρ gewichteten – ungerichteten Kanten ∈ E

Gesucht:

Spannbaum (V, E‘) mit maximaler Gewichtssumme

Algorithmus:

Sortiere die m^2 Kanten von E nach absteigendem Gewicht

E‘ := ∅

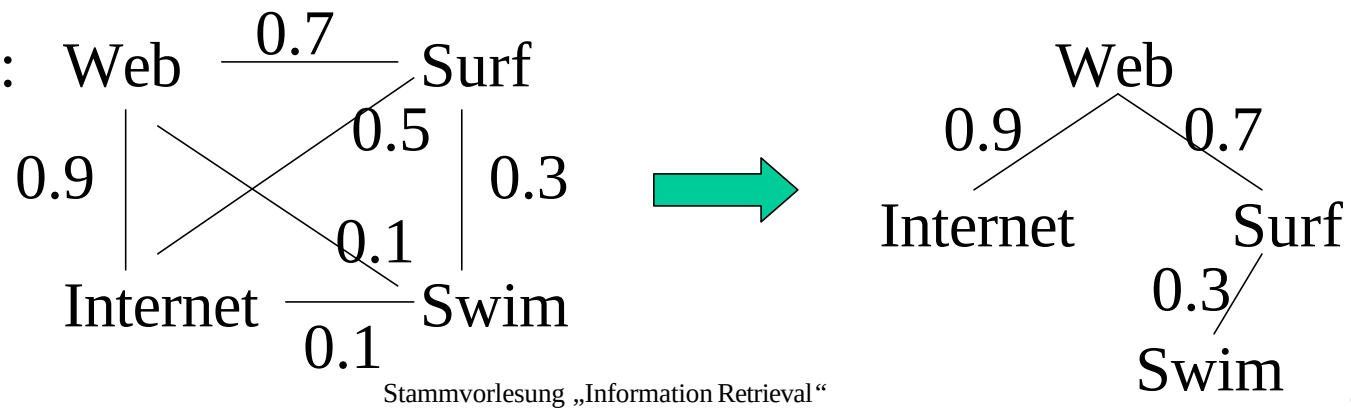
Wiederhole bis |E‘| = m-1

E‘ := E‘ ∪ {(i,j) ∈ E | (i,j) hat max. Gewicht in E}

sofern E‘ azyklisch bleibt;

E := E – {(i,j) ∈ E | (i,j) hat max. Gewicht in E}

Beispiel:



Schätzung mehrdimensionaler Wahrscheinlichkeiten mit dem Termpaarbaum

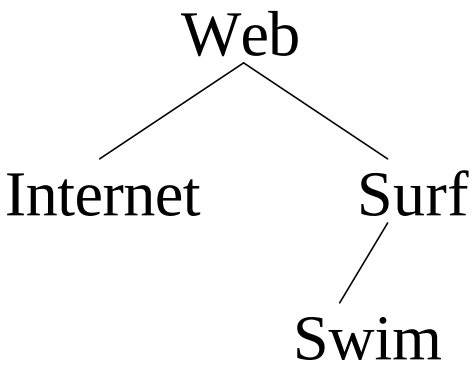
Gegeben sei der Termpaarbaum (V = {X1, ..., Xm}, E‘).
O.B.d.A. sei X1 die Wurzel, und sei angenommen, daß
Xi und Xj für (i,j) $\notin$ E‘ unabhängig sind. Dann gilt:

$$P[X1 = .. \wedge .. \wedge Xm = ..] = P[X1 = ..]\, P[X2 = .. \wedge .. \wedge Xm = .. | X1 = ..]$$

$$= P[X1] \bullet \prod_{(1,i) \in E'} P[Xi \mid X1] \bullet P[\vec{X}_j \; mit \; (1, j) \notin E' | X1, \vec{X}_i \; mit \; (1, i) \in E']$$

$$= P[X1] \bullet \prod_{(i,j) \in E'} P[Xj \mid Xi] \quad = P[X1] \bullet \prod_{(i,j) \in E'} \frac{P[Xi, Xj]}{P[Xi]}$$

Beispiel:



P[Web, Internet, Surf, Swim] =

$$P[Web] \frac{P[Web, Internet]}{P[Web]} \frac{P[Web, Surf]}{P[Web]} \frac{P[Surf, Swim]}{P[Surf]}$$

Latent Semantic Indexing (LSI): Grundsätze

Ziel:

Transformation der Dokumentvektoren vom hochdimensionalen Termvektorraum in einen **Themenvektorraum** niedrigerer Dimensionalität unter

- Ausnutzung von Korrelationen zwischen Termen
  (z.B. „Web“ und „Internet“ häufig zusammen)
- implizite Differenzierung von Polysemen, die sich in ihren Korrelationen mit anderen Termen unterscheiden
  (z.B. „Java“ mit „Library“ vs. „Java“ mit „Kona Blend“ vs. „Java“ mit „Borneo“)

mathematisch:

gegeben: m Terme, n Dokumente (i.d.R. n > m) und eine m×n-Term-Dokument-Ähnlichkeitsmatrix A,

gesucht: möglichst gute – ähnlichkeitsbewahrende – Abbildung der Spaltenvektoren von A in einen k-dimensionalen Vektorraum (k << m) für gegebenes k

Exkurs: Singulärwertdekomposition (SVD)

Satz:
Jede reellwertige m×n-Matrix A mit Rang r kann zerlegt werden
in die Form $\mathbf{A = U \cdot D \cdot V^T}$
mit einer **m´r-Matrix U** mit orthonormalen Spaltenvektoren,
einer **r´r-Diagonalmatrix** D und
einer **n´r-Matrix V** mit orthonormalen Spaltenvektoren.
Diese Zerlegung heißt Singulärwertdekomposition und ist eindeutig,
wenn die Elemente von Δ der Größe nach geordnet werden.

Satz:
In der Singulärwertdekomposition $A = U \times \Delta \times V^T$ der Matrix A
sind U, Δ und V wie folgt bestimmt:
- Δ besteht aus den Singulärwerten von A,
  d.h. den positiven Wurzeln der Eigenwerte von $A^T \times A$,
- die Spaltenvektoren von U sind die Eigenvektoren von $A \times A^T$,
- die Spaltenvektoren von V sind die Eigenvektoren von $A^T \times A$.

Exkurs: SVD als Regressionsverfahren

Satz:

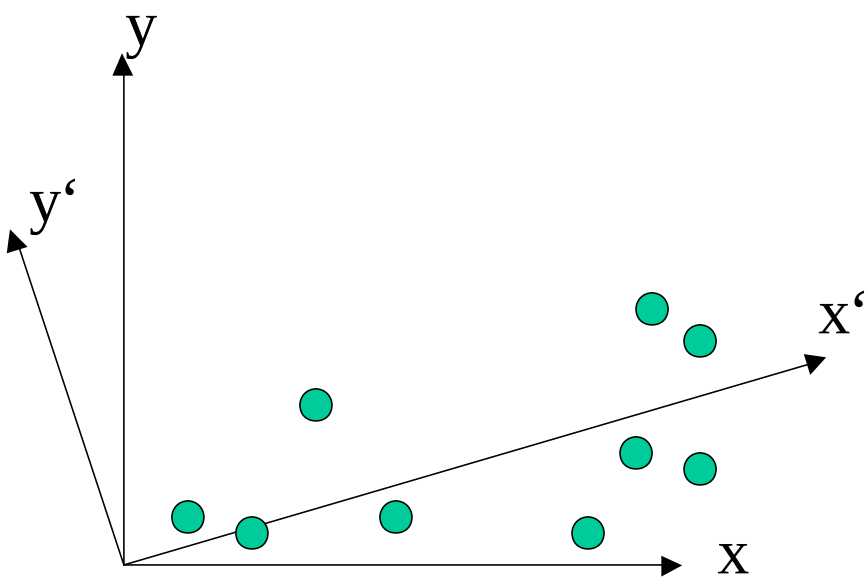
Sei A eine m×n-Matrix mit Rang r und sei $Ak = U_k \times \Delta_k \times V_k^T$, wobei die k×k-Diagonalmatrix Δ_k die k größten Singulärwerte von A enthält und die m×k-Matrix U_k sowie die n×k-Matrix V_k aus den zugehörigen Eigenvektoren der Singulärwertdekomposition von A bestehen.

Unter allen m×n-Matrizen C mit einem Rang, der nicht größer als k ist, ist A_k diejenige Matrix, die den Wert

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

minimiert (die Frobenius-Norm).

Beispiel:
m=2, n=8, k=1
Projektion auf x‘-Achse
minimiert „Fehler“ bzw.
maximiert Varianz
im k-dimensionalen Raum



Anwendung der SVD auf das Vektorraummodell

A ist die m×n-Term-Dokument-Ähnlichkeitsmatrix. Dann sind:

- U bzw. U_k die m×r- bzw. m×k-Term-Themen-Ähnlichkeitsmatrix,
- V bzw. V_k die n×r- bzw. n×k-Dokument-Themen-Ähnlichkeitsmatrix
- $A \times A^T$ bzw. $A_k \times A_k^T$ die m×m-Term-Term-Ähnlichkeitsmatrix,
- $A^T \times A$ bzw. $A_k^T \times A_k$ die n×n-Dokument-Dokument-Ähnlichkeitsmatrix

Indexierung und Anfrageauswertung:

- Die Matrix V_k^T entspricht einem „Themen-Index“ und ist in einer geeigneten Datenstruktur zu verwalten.
- Zusätzlich muß die Term-Themen-Abbildung U_k gespeichert werden.
- Eine Anfrage q (ein 1×m-Zeilenvektor) im Termvektorraum wird in die Anfrage $q' = q \times U_k$ (ein 1×k-Zeilenvektor) transformiert und dann im Themenvektorraum (also V_k) ausgewertet (z.B. mittels Skalarproduktmaß $q' \times V_k^T$ oder Cosinusmaß)
- Ein neues Dokument d (ein m×1-Spaltenvektor) wird in $d' = U_k^T \times d$ (ein k ×1-Spaltenvektor) transformiert und als neue Spalte an den „Index“ V_k^T angefügt („folding-in“)

Beispiel 1 für Latent Semantic Indexing

m=5 (Lafontaine, Schröder, Euro, Uefa, Beckenbauer), n=7

$$A = \begin{pmatrix} 1&2&1&5&0&0&0 \\ 1&2&1&5&0&0&0 \\ 1&2&1&5&0&0&0 \\ 0&0&0&0&2&3&1 \\ 0&0&0&0&2&3&1 \end{pmatrix} = \begin{pmatrix} 0.58 & 0.00 \\ 0.58 & 0.00 \\ 0.58 & 0.00 \\ 0.00 & 0.71 \\ 0.00 & 0.71 \end{pmatrix} \times \begin{pmatrix} 9.64 & 0.00 \\ 0.00 & 5.29 \end{pmatrix} \times \begin{pmatrix} 0.18 & 0.36 & 0.18 & 0.90 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.53 & 0.80 & 0.27 \end{pmatrix}$$

U $\quad$ Δ $\quad$ V^T

Anfrage q = (0 0 1 0 0) wird in
q‘ = q × U = (0.58 0.00) transformiert und gegen V^T evaluiert.

Neues Dokument d8 = $(1\ 1\ 0\ 0\ 0)^T$ wird in
d8‘ = $U^T \times$ d8 = $(1.16\ \ 0.00)^T$ transformiert und an V^T angefügt.

Beispiel 2 für 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

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

Beispiel 2 für Latent Semantic Indexing (2)

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

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

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

Beispiel 2 für Latent Semantic Indexing (3)

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

Beispiel 2 für Latent Semantic Indexing (4)

Anfrage q: baking bread
q = (1 0 1 0 0 0)

Transformation in den Themenraum mit k=3
$q' = q \times U_k = (0.5340 \;\; -0.5134 \;\; 1.0616)$

Skalarprodukt-Ähnlichkeit im Themenraum mit k=3:
$sim(q, d1) = q' \times Vk_{*1}^T \approx 0.8$ $sim(q, d2) = q' \times Vk_{*2}^T \approx -0.2$
$sim(q, d3) = q' \times Vk_{*3}^T \approx -0.2$ usw.

Folding-in eines neuen Dokuments d6:
algorithmic recipes for the computation of pie
$d6 = (0 \;\; 0.7071 \;\; 0 \;\; 0 \;\; 0 \;\; 0.7071)^T$

Transformation in den Themenraum mit k=3
$d6' = U_k^T \times d6 \approx (0.5 \;\; -0.28 \;\; -0.15)$

d6‘ als neue Spalte an Vk^T anhängen

Mehrsprachiges Retrieval mit LSI

- Konstruiere LSI-Modell (U_k, Δ_k, V_k^T) anhand von Trainingsdokumenten, die mehrsprachig vorliegen:
  - Betrachte alle Sprachversionen eines Dokuments als ein einziges Dokument und
  - extrahiere zur Indexierung alle Terme oder Wörter unabhängig von der Sprache.
- Indexiere weitere Dokumente durch „folding-in“, also Abbildung in den Themen-Vektorraum und Anhängen an V_k^T.
- Anfragen können dann in beliebiger Sprache gestellt werden und liefern Antworten in allen Sprachen.

Beispiel:

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.

Terme sind dann z.B. bake, bread, recipe, backen, Brot, Rezept, usw.
Dokumente und Terme werden auf einen kompakten Themenraum abgebildet.

Zusammenfassung zu LSI

+ Elegantes, mathematisch wohlfundiertes Modell
+ „Automatisches Lernen“ von Termkorrelationen
  (inkl. morphologischer Wortvarianten, Mehrsprachigkeit)
+ Impliziter Thesaurus (durch Korrelation von Synonymen)
+ Implizite Diskriminierung der verschiedenen Bedeutungen
  von Polysemen (durch verschiedene Korrelationen)
+ Verbesserte Retrievalgüte auf „geschlossenen“ Korpora
  (z.B. TREC-Benchmark, Finanznachrichten, Patentkollektionen u.ä.)
  mit empirisch günstigstem k in der Größenordnung 100-200
– Schwierige Wahl von günstigem k
– Rechen- und Speicheraufwand für sehr große (z.T. aber dünn
  besetzte) Matrizen
– Keine überzeugenden Resultate für Web-Suchmaschinen