Einführung in Filtermethoden und Information Retrieval

Andre Masloch

Quellen

- C.Faloutsos und D.W.Oard. A Survey of Information Retrieval and Filtering Methods.

- R.S.Boyer und J.S.Moore. A fast string searching algorithm. *CACM*, 20(10):762-772, Oktober 1977
- G.Salton und A.Wong. Generation and search of clustered files. *ACM TODS*, 3(4):321-346, Dezember 1978
- C.J.Van-Rijsbergen. An algorithm for information structuring and retrieval. *Computer Journal*, 14(4):407-412, 1971
- S.Deerwester, S.T.Dumais, G.W.Furnas, T.K.Landauer und R.Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391-407, 1990

Übersicht

- Traditionelle Methoden
- neuere Methoden
- kurze Bewertung der Methoden

Traditionelle Methoden

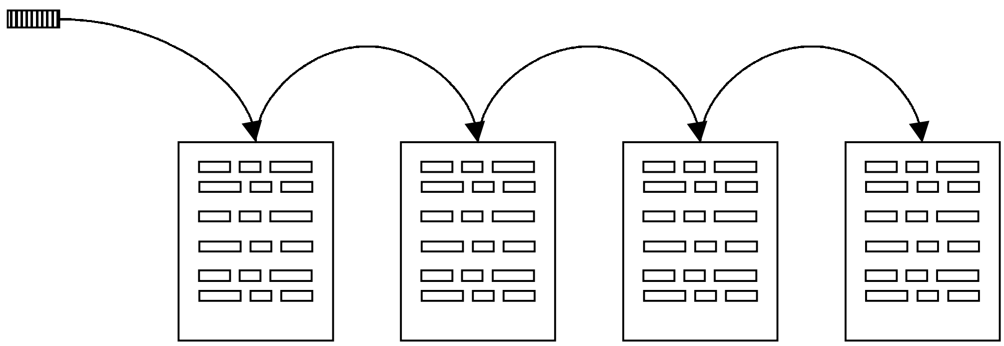
- Volltextsuche
  - „Brute Force“-Ansatz
  - Boyer-Moore-Algorithmus
- Signaturdateien
- Inversion
- Vektorenmodell, Clustering
  - Clustererstellung
  - Suchen in Clustern

neuere Methoden

- Latent Semantic Indexing
  - Technik
  - Dokumente vergleichen
  - Terme vergleichen
- Natural Language Processing
- (Neurale Netze)

Volltextsuche

- Durchsuchen aller Texte nach dem gesuchten Wort (Muster)



- **Vorteile :**
  - kein Speicher-Overhead
  - geringe Kosten für Einfügen und Ändern
- **Nachteil :**
  - schlechte Leistung (Antwortzeit) besonders bei großen Datenmengen

„Brute Force“-Ansatz

- serieller Vergleich der Zeichen des Suchmusters mit denen des Dokumentes
- stimmt das (aktuelle) Zeichen des Dokumentes nicht mit dem (aktuellen) des Suchtextes überein, geht man zum nächsten Zeichen im Dokument und fängt wieder am Anfang des Suchmusters an

- worst-case-Laufzeit : ***O(n*m)***
  m: Suchmusterlänge,
  n: Dokumentlänge

Boyer-Moore-Algorithmus

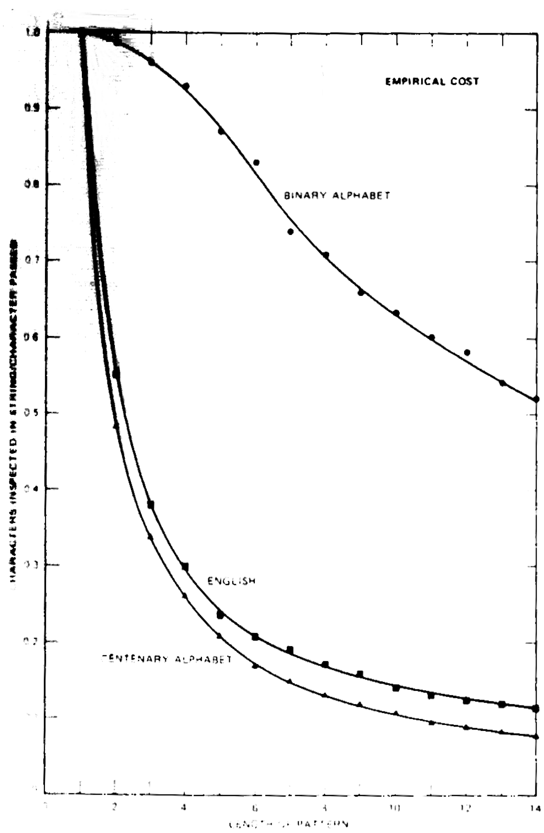
- Umgekehrte Suche des Suchmusters (von rechts nach links)
- bei Nichtübereinstimmung des aktuell betrachteten Zeichens kann das Suchmuster bis zu *m* Positionen im Dokument vorrücken

- Die Vorverarbeitung des Suchmusters benötigt ***O(m)***

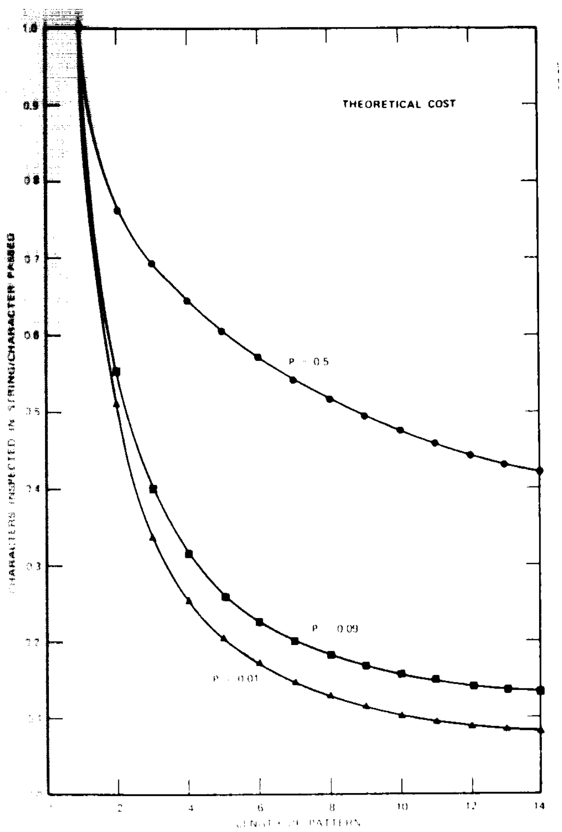
- worst-case-Laufzeit : ***O(n+m)***
- Average-case (*m=5*) : ***i/4***
  i: Startposition der Fundstelle

Boyer-Moore-Alg. Statistik

gemessen



Theorie



Boyer-Moore-Alg. Vorverarbeitung I

Benötigt werden zwei Arrays :

delta_1 :

Eine Stelle für jedes Zeichen im Alphabet

delta_2 :

Eine Stelle für jede Stelle des Suchmusters (Also *m* Stellen)

Erzeugen des Arrays $\text{delta}_1(z)$:

Für jeden Buchstaben *z* im Alphabet wird ein Integer wie folgt in $\text{delta}_1(z)$ bestimmt :

Taucht *z* nicht im Suchmuster auf, so ist $\text{delta}_1(z) = m$, ansonsten ist $\text{delta}_1(z) = m - j$, wobei *j* die letzte Stelle ist, an der *z* im Suchmuster auftaucht.

Boyer-Moore-Alg. Vorverarbeitung II

Erzeugen des Arrays $\mathrm{delta}_2(j)$:

Für jede Position *j* des Suchmusters wird ein Integer wie folgt in $\mathrm{delta}_2(j)$ bestimmt :

Suche die größte Position rpr(*j*), so daß das Teilmuster [*j+1*,...,*m*] wieder im Suchmuster auftaucht, aber nicht das Teilmuster [*j*,...,*m*].

Dann ist $\mathrm{delta}_2(j) = m + 1 - \mathrm{rpr}(j)$
die Anzahl Stellen, um die das Suchmuster verschoben werden kann.

Boyer-Moore-Alg.
Beispiel $delta_2$ -Array

j:	1	2	3	4	5	6	7	8	9
Such-muster	A	B	C	X	X	X	A	B	C
$Delta_2(j)$	14	13	12	11	10	9	11	10	1

für $j = 1$: $delta_2(j) = 9 + 1 - (-4)$

-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9
						A	B	C	X	X	X	A	B	C
A	B	C	X	X	X	A	B	C						

Boyer-Moore-Algorithmus

```
       i := m ;
       large:=n+m+1;
       d0(pat[m]):=large;
       if i>n then return false;
fast:  i:=i+d0(text[i]);
       if i≤n then goto fast;
undo:  if i≤large then return false;
       i:=(i-large)-1;
       j:=m-1;
slow:  if j=0 then return i+1
       if text[i]=pat[j] then
          i:=i-1; j:=j-1;
          goto slow;
       endif;
       i:=i+max(d1(text[i]),d2(j));
       goto fast;
```

Boyer-Moore-Alg. Beispiel

Suchmuster : `AT−THAT`

Dokument :

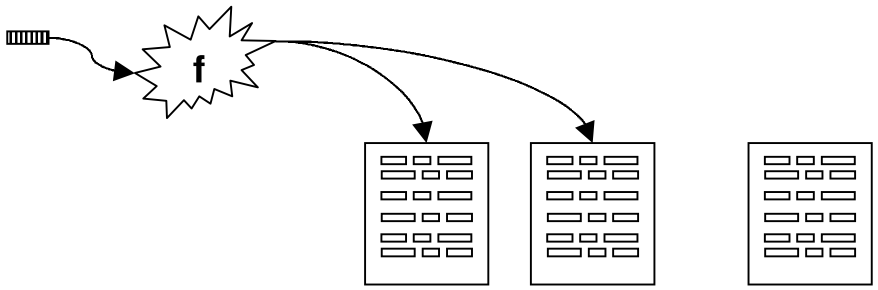
```
... WHICH−FINALLY−HALTS.−−AT−THAT−POINT ...

    AT−THAT
... WHICH−FINALLY−HALTS.−−AT−THAT−POINT ...
          ↑
          AT−THAT
... WHICH−FINALLY−HALTS.−−AT−THAT−POINT ...
                ↑
             AT−THAT
... WHICH−FINALLY−HALTS.−−AT−THAT−POINT ...
                   ↑
                  AT−THAT
... WHICH−FINALLY−HALTS.−−AT−THAT−POINT ...
                        ↑
                      AT−THAT
... WHICH−FINALLY−HALTS.−−AT−THAT−POINT ...
                         ↑
```

Volltextsuche - Andere Varianten

- Knuth, Morris, Pratt
  Ähnlich zu Boyer-Moore
  Laufzeit : ***O(m+n)***
  Vorverarbeitung benötigt ***O(m)***
- Aho, Corasick
  Endlicher Automat, parallele Suche von mehreren Suchtexten möglich
  Laufzeit : ***O(n)***
  Kosten zur Herstellung des Automaten sind linear zu der Summe der Buchstaben der Suchtexte.
- Wu, Manber
  Tolerierung von Tippfehlern, schnell und flexibel

Signaturdateien

- Benutzen von Hashing und überlagernder Codierung
- Speicherung der Signaturen in einer Signaturdatei



- **Vorteile :**
  - einfache Implementierung, leicht Parallelisierbar
  - schnelles Einfügen ist möglich
  - Tolerierung von Tippfehlern
  - Teilwortsuche ist möglich

- **Nachteile :**
  - Antwortzeit bei großer Signaturdatei
  - Änderungen sind schwieriger

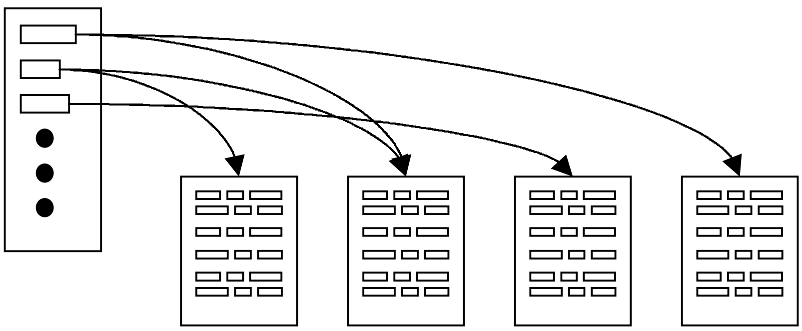
Signaturdateien - Variationen

- Hashing von
  - Dokumenten
  - Worten
  - Textsegmenten
- Benutzen von STOP-Listen
- Reduzierung der Worte auf Stammformen

- keine Überlagerung verwenden
- Multi-Level-Signaturen
- spezielle Anordnungen in der Signaturdatei („bit-slicing“)

Inversion

- Repräsentation des Dokumentes durch eine Liste von Schlüsselworten, die im Dokument vorkommen
- Einfache Suche durch „Umkehrung“ der Listen
  - Jedes Schlüsselwort hat eine Zeigerliste auf die passenden Dokumente
  - Die (sortierte) Schlüsselwortliste wird in einer separaten Indexdatei gespeichert z.B. als Liste, B-Baum



Inversion

- **Vorteile :**
  - einfache Implementation
  - schnell (auch bei großen Datenmengen)
  - einfache Unterstützung von Synonymen

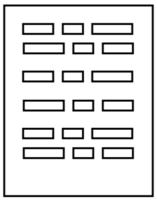
- **Nachteile :**
  - Speicherplatzverbrauch bis zu 300% vom Originalverbrauch
  - Änderungs- und Reorganisierungskosten der Indizes bei dynamischer Umgebung
  - Zusammenführungskosten bei langen Listen
  - Verzerrung der Listen beeinflußt die Suchgeschwindigkeit

Inversion - neuere Entwicklungen

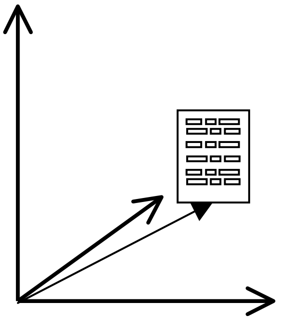
- Vermeidung der „Verzerrung“
  - durch hybride Methoden und
  - durch Algorithmen, die die Listen adaptiv wachsen lassen
- Techniken, um Einfügungen bei großen Indizes schnell auszuführen
  - Ausnutzen der Verzerrung durch unterschiedliche Behandlung von kurzen und langen Listen
  - Komprimierung der Indizes
  - grobe Indizes für ungenaues Suchen

Vektorenmodell, Clustering

- Zusammenfassen von ähnlichen Dokumenten in Gruppen (sog. Cluster)
- Jedes Dokument wird durch einen ***n***-dimensionalen Vektor dargestellt
- Die Stellen des Vektors stehen jeweils für ein Schlüsselwort
- Verwenden eines binären Gewichtes oder einer anderen Gewichtsfunktion für die Stellen des Vektors



$$\rightarrow \begin{pmatrix} 2.4 \\ 0.6 \\ 1.3 \end{pmatrix}$$



Vektorenmodell - Gewichtsfunktionen

- „occurrence frequency“
  FREQ(*i*,*k*)
- „Term specificity“
  log *N* - log*(DOCFREQ(k)) + 1*
- „Inverse document frequency“
  FREQ(i,k) / DOCFREQ(k)
- „Optimal weight“
  *FREQ(i,k) * TERMREL(k)*

  mit TERMREL(k) = $\dfrac{r_k / (R - r_k)}{s_k / (I - s_k)}$

N: Gesamtzahl der Dokumente,

k: Wort, *i*: Dokument,

R: relevante Dokumente,

r_k : relevante Dokumente mit Wort *k*

I: irrelevante Dokumente,

s_k : irrelevante Dokumente mit Wort *k*

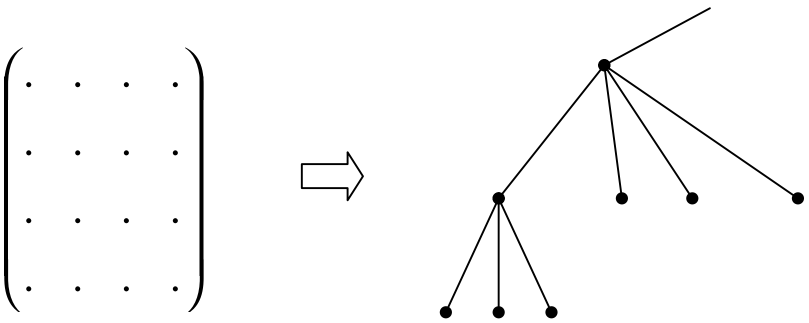
Clustererstellung

Wünschenswerte Eigenschaften

- Stabilität bei Wachstum
- kleine Fehler in der Beschreibung verursachen nur kleine Fehler in der Gruppierung
- Unabhängigkeit von der anfänglichen Anordnung / Reihenfolge der Dokumente
- Effizienz (schneller Algorithmus)

Clustererstellung - Ähnlichkeitsmatrix

- Erstellen einer Ähnlichkeitsmatrix anhand einer *Dokument-zu-Dokument*-Vergleichsfunktion.
- Einteilung der Dokumente in den selben Cluster z.B. durch
  - vorgegebenen *Schwellwert*
  - Graph und *„single-link“* -Kriterium (nächster Nachbar)



- Zeitaufwand: ***O(*** n^2***)***

Clustererstellung - iterative Methode

- Erzeugen einer Partitionierung
- Wiederholtes Neuanordnen und Abändern der Partitionierung, bis keine „Verbesserung“ mehr auftritt.

- Zeitaufwand : ***O(*** $n\log(n)$ ***), O(*** $\frac{n^2}{\log(n)}$ ***)***

Beispiel : Single-pass-Verfahren

jedes Dokument wird mit den bestehenden Clustern verglichen und dementsprechend den Clustern hinzugefügt oder es wird ein neuer Cluster erzeugt

Suchen in Clustern

- Vergleich des Dokument- oder Query-Vektors mit den Schwerpunkten der Cluster z.B. durch Cosinus-Maß
- Erweiterung :
  Verbesserung der Suche durch *relevance-feedback*

- *relevance feedback :*

  Der Benutzer sagt dem System, welche der gefundenen Dokumente treffend waren und das System formuliert anschließend den Anfragevektor neu (z.B. vektorielle Addition der Treffer und Subtraktion der Nicht-Treffer).

neuere Methoden

- Latent Semantic Indexing
  - Technik
  - Dokumente vergleichen
  - Terme vergleichen
- Natural Language Processing

Latent Semantic Indexing

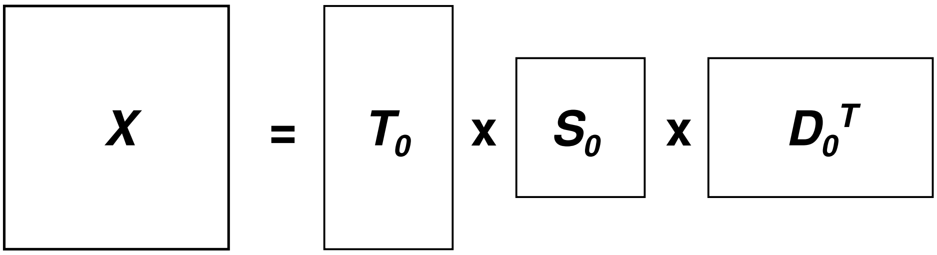
- Repräsentation der Dokumente als Vektoren von Term-Häufigkeiten
- Aufstellen einer
  Term-Dokument-Matrix $\boldsymbol{X}$ (t x d)
- Anschließend :
  Zerlegen der Matrix mittels SVD (Singular Value Decomposition), um syntaktische Variationen und Fehler in den Dokumenten zu unterdrücken, aber den Inhalt (Semantik) zu erhalten.

- Suche durch Vektorvergleich (Cosinus-Maß)

t: Gesamtanzahl der Terme

d: Gesamtanzahl der Dokumente

LSI - Technik



- Aufspalten der Matrix ***X*** (*t* x *d*) mit SVD in
  T_0 (*t* x *m*),
  S_0 (*m* x *m*) und
  D_0 (m x d)
  mit $X = T_0 S_0 D_0^T$ (*m* = rang(***X***))
- Verändern der S_0-Matrix (nur die *k* größten Werte bleiben erhalten)
- Annähern von ***X*** durch neue Matrizen mit kleineren Dimensionen (ersetze *m* durch *k* (100 < *k* < 200))

LSI - Beispielmatrix

Titel :

c1 : *Human* machine *interface* for Lab ABC *computer* applications

c2 : A *survey* of *user* opinion of *computer system response time*

c3 : The *EPS user interface* managment *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, unordered *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*

Term	Dokumente								
	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

$T_0 =$

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

$S_0 =$

3.34								
	2.54							
		2.35						
			1.64					
				1.50				
					1.31			
						0.85		
							0.56	
								0.36

$D_0 =$

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

$X \approx$

T

0.22	−0.11
0.20	−0.07
0.24	0.04
0.40	0.06
0.64	−0.17
0.27	0.11
0.27	0.11
0.30	−0.14
0.21	0.27
0.01	0.49
0.04	0.62
0.03	0.45

S

3.34	
	2.54

D'

0.20	0.61	0.46	0.54	0.28	0.00	0.02	0.02	0.08
−0.06	0.17	−0.13	−0.23	0.11	0.19	0.44	0.62	0.53

$\hat{X} =$

0.16	0.40	0.38	0.47	0.18	−0.05	−0.12	−0.16	−0.09
0.14	0.37	0.33	0.40	0.16	−0.03	−0.07	−0.10	−0.04
0.15	0.51	0.36	0.41	0.24	0.02	0.06	0.09	0.12
0.26	0.84	0.61	0.70	0.39	0.03	0.08	0.12	0.19
0.45	1.23	1.05	1.27	0.56	−0.07	−0.15	−0.21	−0.05
0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
0.22	0.55	0.51	0.63	0.24	−0.07	−0.14	−0.20	−0.11
0.10	0.53	0.23	0.21	0.27	0.14	0.31	0.44	0.42
−0.06	0.23	−0.14	−0.27	0.14	0.24	0.55	0.77	0.66
−0.06	0.34	−0.15	−0.30	0.20	0.31	0.69	0.98	0.85
−0.04	0.25	−0.10	−0.21	0.15	0.22	0.50	0.71	0.62

Dokumente vergleichen

- Ein Vergleich zweier Dokumente entspricht dem Skalarprodukt zwischen zwei *Spalten* von ***X.***
- Betrachte die Matrix $X'X = DS^2D'$
  Der Vergleich entspricht dem Skalarprodukt der i-ten Zeile von ***DS*** mit
  der j-ten Zeile von ***DS***.

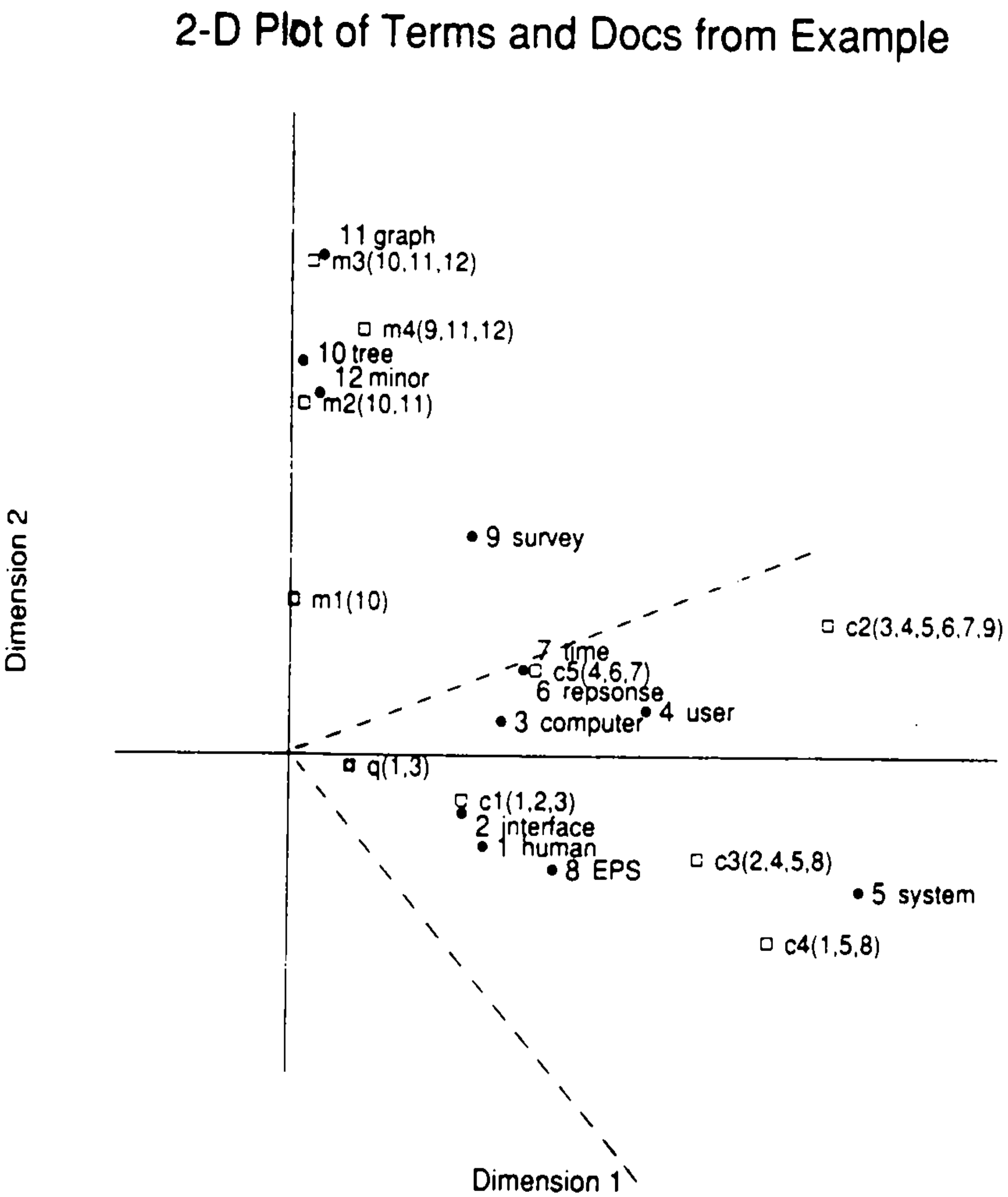
- Die Dokumente können grafisch also als Punkte im m-dimensionalen Raum angesehen werden. Die Koordinaten eines Punktes entsprechen einer Zeile der ***D***-Matrix skaliert durch ***S.***

Terme vergleichen

- Ein Vergleich zweier Terme entspricht dem Skalarprodukt zwischen zwei *Zeilen* von ***X.***
- Betrachte die Matrix $\boldsymbol{XX' = TS^2T'}$
  Der Vergleich entspricht dem Skalarprodukt der i-ten Zeile von ***TS*** mit
  der j-ten Zeile von ***TS***.

- Die Terme können grafisch also als Punkte im m-dimensionalen Raum angesehen werden. Die Koordinaten eines Punktes entsprechen einer Zeile der
  T-Matrix skaliert durch ***S.***

LSI - grafisches Bsp.



q: Pseudo-Dokument „Human computer interaction“

LSI

- **Nachteile :**
  - Die Berechnung der SVD ist ziemlich zeitaufwendig ($>(t+d)^2 * m^3$).
  - Neue Dokumente oder Terme können nur schwer aufgenommen werden.

Natural Language Processing

- Einbeziehen der semantischen Information im Dokument und der Anfrage durch
  - Syntaxanalyse (Parser)
  - semantische Analyse