Grundlegende Methoden in Information Retrieval

Cristina Vertan, Walther v. Hahn
{vertan,vhahn}@informatik.uni-hamburg.de

Rahmenbedingungen für ein IR-Szenario

- Man sucht Information in unstrukturierten Dokumenten
- Die Anzahl der Dokumente ist groß.
- Die Anfrage ist komplex (nicht nur elementare boolesche Operatoren zwischen Wörtern)
- Öfter wird ein Relevanz-Score für die Dokumente in der Ergebnisliste gewünscht.

Lineare Suche (grepping) ist ineffizient,

Lösung: Dokument-Indexierung

Boolesche Indexierung - Prinzip-

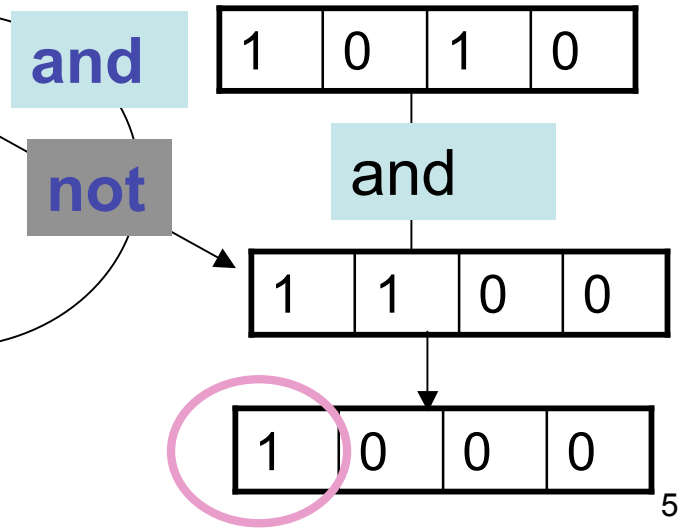
- Man extrahiert „Terms“= Schlüsselwörter/Mehrwortausdrücke aus den Dokumenten und konstruiert eine Term-Dokument Matrix wobei:
  - Jede Zeile einem Term entspricht
  - Jede Spalte einem Dokument
  - Eine Zelle (i,j) kann die Werte 1/0 enthalten: 1 wenn der Term i in Dokument j existiert und 0 wenn nicht.
- Die Anfrage wird auch durch einen booleschen Ausdruck dargestellt

Boolesche Indexierung -Beispiel

- Man sucht alle Dokumente, die sich mit XML, SGML aber nicht mit HTML sich beschäftigen

Anfrage formalisiert durch: XML **and** SGML and **not** HTML

	D1	D2	D3	D4
XML	1	0	1	0
HTML	0	0	1	1
XPATH	1		0	0
Markup-Sprachen	1	1	1	1
SGML	1	1	1	0



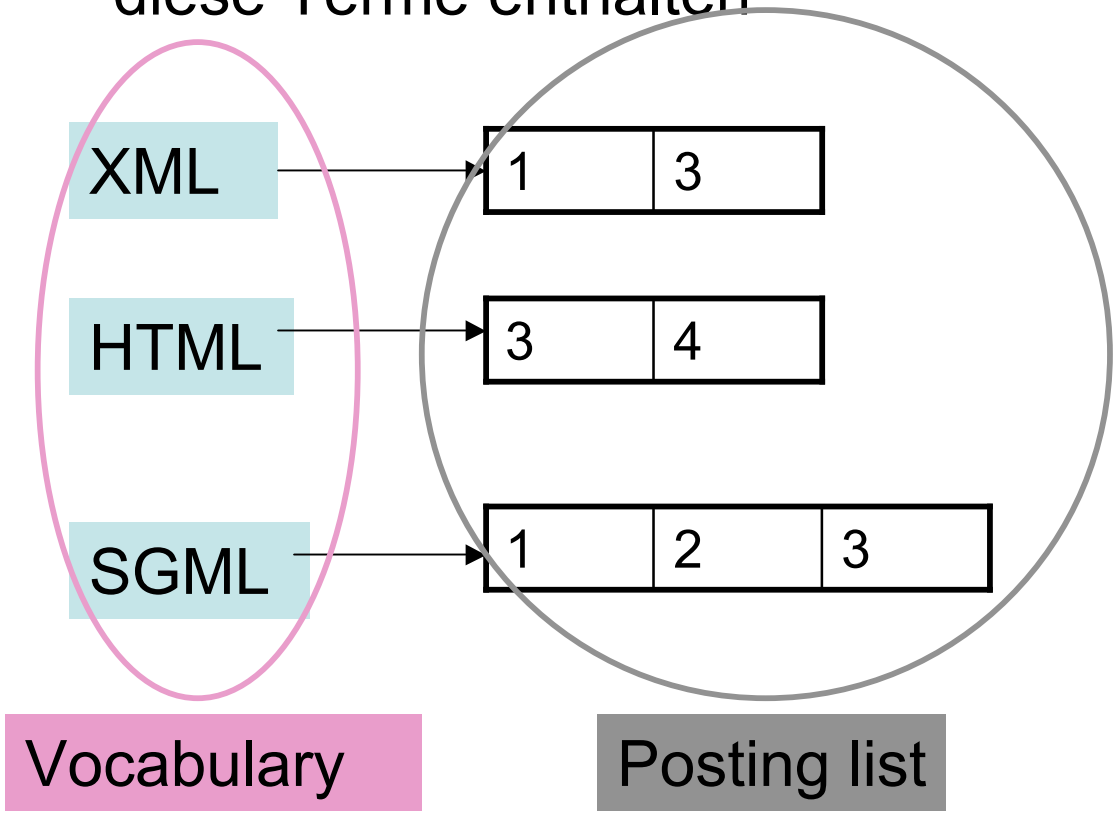
Grenze der boolesche Indexierung

- In einem realistischen Szenario (z.B. Web) sucht man durch zirka 1 Billion Dokumente und jedes Dokument enthält zirka 1000 Terme, d.h., man kann annehmen, dass zirka 500 000 distinkte Terme in allen Dokumenten existieren.
- Die Matrize hat dann 500 000×1Billion Zellen und viele davon sind 0, d.h, viel Platz wird unnötig verschenkt.
- Man bekommt keine Information über die Relevanz der Ergebnisse
- Man kann nur AND/OR/NOT -Anfragen stellen

„inverted“ Index

„Inverted“-Index - Prinzip -1-

- Man speichert nur die Terme und die Dokumente die diese Terme enthalten



Zusätzlich speichert man die Term/Dokument-Frequänz

„Inverted“-Index -Algorithmus -2-

- Man sammelt die Dokumente, die indiziert werden.
- Die Texte werden tokenisiert (d.h. jedes einzelne Wort /Term) wird isoliert
- Sinnvoll ist manchmal eine linguistische Vorbearbeitung (z.B. Lemmatisierung)
- Aufbau des Inverted-Index . Eine Liste mit Termen, deren Frequenz, und die entsprechende Posting -Liste.
- Boolesche Anfragen werden durch die Kombination der entsprechenden Posting-Lists formuliert
- Kombination heißt hier: Mengeoperationen über Listen
- Wichtig ist hier, dass man einen effizienten Algorithmus für Listennavigation implementiert

Grenzen des „Inverted“-Index Algorithmus

- Nun hat man zwar das Speicherplatz -Problem gelöst, aber man kann doch nur boolesche Anfragen bearbeiten;
- Synonyme beispielsweise werden als unterschied-liche Terme betrachtet

Mehr Bearbeitungsaufwand bei der Erstellung des Lexikons und der posting-Listen

Normalisierung

- Man kann Synonymen-Listen definieren, d.h. Listen mit Temen die für bestimmte Domänen änliche Bedeutungen haben.
- D.h. wenn man HTML und XHTML als Synonyme definiert, dann wird ein Dokument, das XHTML enthält, in der posting-Listen für HTML indexiert.

N-Gramms Indexe

- Die Hypothese ist hier, dass viele Wort-Sequenzen, die in der Anfrage vorkommen, auch konsekutiv in Text vorkommen.
- Das ist relevant besonders für Sequenzen wie <Nomen1>...<Nomen3>, <DET><NOMEN>, <ADJ>. <NOMEN>
- 2 Vorgehensweise:
  - Entweder man extrahiert 2- und 3-gramme
  - Oder man macht zuerst ein PoS-Tagging für das Dokument und extrahiert ausgewählte syntaktische Sequenzen
- Die Anfrage wird dann in boolesche Kombinationen von Wortsequenzen übersetzt.

Grenzen von Vorbearbeitung der Posting-Listen

- Man braucht linguistische Werkzeuge, die nicht für jede Sprache verfügbar sind
- Die Übersetzung der Anfrage in boolesche Ausdrucke von Term-Sequenzen erfolgt nicht immer,
- Alle Terme in der Anfrage werden als gleichgewichtig betrachtet.

„Inverse Dokument „-Frequenz -Algorithmus

Inverse Dokument-Frequänz -Prinzip

- Die Frequenz des Terms t in Dokument d, notiert mit tf_{td} wird durch die Logarithmus-Funktion gewichtet:
- D.h.
  - $wf_{td} = 1 + \log tf_{td}$, wenn $tf_{td} > 0$ und
  - $wf_{td} = 0$ wenn $tf_{td}=0$
- Mit dieser Gewichtung werden alle Termes in einer Anfrage als gleich wichtig bhandelt.
- Deswegen berechnet man auch die Inverse-Frequenz $(idf_t)= \log (N / df_t)$
  - Wobei df_t= die Anzahl von Dokumenten, die den Term t enthalten
- Die Wichtigkeit des Terms für ein Dokument ist eine Mischung der 2 Gewicht, nämlich:

$$\textbf{(1) Tf-idf}_{td} = \textbf{tft}_{d} \times \textbf{idf}_{t}$$

- Der sog. Konfidenzwert einer Anfrage ist dann die Summe des Koeffizienten (1) die für für jeden Term der Anfrage berechnet werden

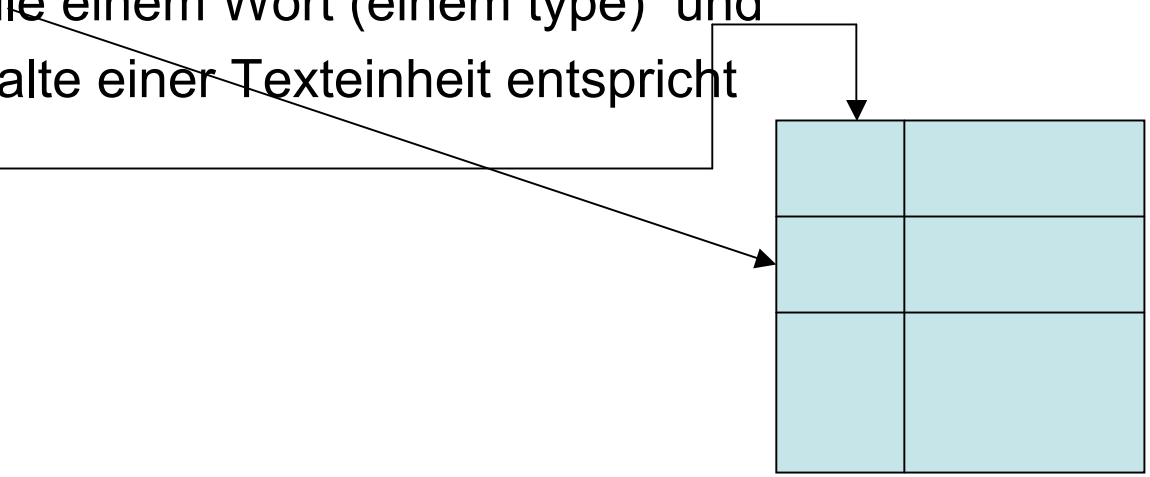
Latent Semantic Analysis (LSA)

- LSA ist eine vollautomatische statistische Methode, um aus sehr großen Textmengen die Wahrscheinlichkeit von lexikalisch-semantischen Beziehungen (Ähnlichkeiten) zu erheben und in großen Matrizen (ca 100 x 500) darzustellen.
- LSA arbeitet ohne:
  - Lexikon
  - Wissensbasis
  - Semantische Netze
  - Syntaktische Parser
  - Morphologie

- Home-Page: http://lsa.colorado.edu/

Latent Semantic Analysis
-Eingabe-

- Die LSA-Eingabe ist allein der Rohtext:
  - in Wörter segmentiert (ein Wort = ein einziger String)
  - in bedeutungsvolle Passagen getrennt (Sätze, Paragraphen)
- Der Text wird in eine Matrix eingelesen, in der:
  - jede Zeile einem Wort (einem type) und
  - jede Spalte einer Texteinheit entspricht



Latent Semantic Analysis
-Texteingabe-Beispiel -

Benutzt werden Wörter, die in mindestens 2 Titeln erscheinen (außer extrem häufige Funktionswörter)

Text (Titel technischer Berichte):

c1: *Human* machine *interface* for *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*

Spaltenanordnung

m4: *Graph minors*: A *survey*

m3: *Graph minors* IV: Widths of *trees* and well-quasi-ordering

m2: The intersection *graph* of paths in *trees*

m1: The generation of random, binary, ordered *trees*

c5: Relation of *user* perceived *response time* to error measurement

c4: System and *human system engineering testing of EPS*

c3: The *EPS user interface* management *system*

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

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

Latent Semantic Analysis
-Textmatrizen-Beispiel -

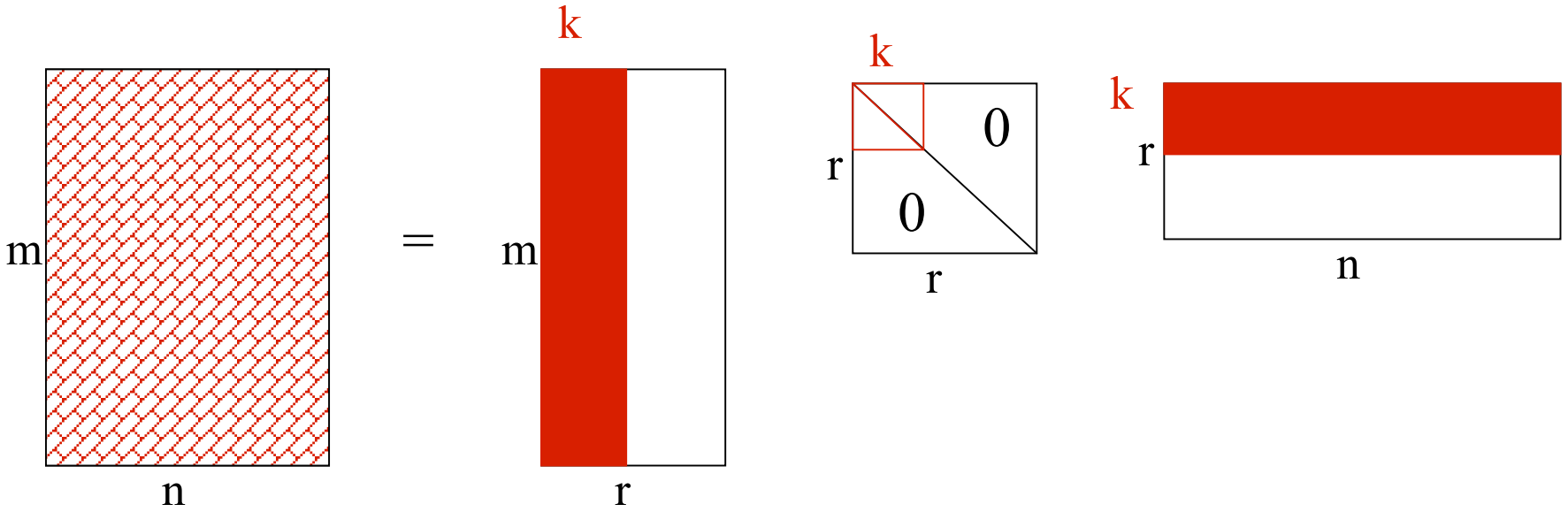
	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	1	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors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

Auftretenshäufigkeit im entspechenden Satz

In realistischen Anwendungen werden die Zellen gewichtet.

Latent Semantic Analysis
-Matrizentransformation -

SVD - Single Value Decomposition



A = B × I × C

Je größer die Matrizen umso größer der Berechnungsaufwand

A_k = B_k × I_k × C_k

Latent Semantic Analysis
- Matrizentransformations-Beispiel -

Matrizenrekonstruktion mit k=2

	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

"tree" erscheint nicht in m4, aber in Titeln mit "graph" und "minors"

korr(human,user) = 0.94

korr(human,minors) = -0.83

	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

korr(human,user) = 0.38

korr(human,minors) = 0.29

c1	c2	c3	c4	c5	m1	m2	m3	m4
1	0	0	1	0	0	0	0	0
1	0	1	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0
0	1	1	0	1	0	0	0	0
0	1	1	2	0	0	0	0	0
0	1	0	0	1	0	0	0	0
0	1	0	0	1	0	0	0	0
0	0	1	1	0	0	0	0	0
0	1	0	1	0	0	0	0	1
0	0	0	0	0	1	1	1	0
0	0	0	0	0	0	1	1	1
0	0	0	0	0	0	0	1	1

Original

LSA als Modell menschlichen konzeptuellen Wissens (LSA Evaluationsliste)

- predictor of query ⇔ document topic similarity judgements
- a simulation of agreed upon word ⇔ word relations and of human vocabulary test synonym judgements
- a simulation of human choices on subject-matter multiple choice tests
- a predictor of text coherence and resulting comprehension
- a simulation of word ⇔ word and passage ⇔ word relations found in lexical priming experiments
- subjective ratings of text properties (i.e. grades assigned to essays).
- a predictor of appropriate matches of instructional text to learners,
- to mimic synonym, antonym, singular-plural and compound word relations

„MÜ“-Strategien

- Lexikalischer Transfer
- On-Line Übersetzung
- Beispiel:
  - man baut eine Datenbank mit möglichen Anfragen-Beispielen
  - Die Anfrage wird durch Edit-Distance mit den Beispielen verglichen
  - Gefundene Segmenten werden zusammengesetzt.

Architektur des Systems