Kapitel 16: Dokumenten– und Multimedia–Retrieval

16.1 Grundprinzipien des Information Retrieval

DBS verwalten große Sammlungen strukturierter Daten.
Information–Retrieval–Systeme (IRS) verwalten große Sammlungen *unstrukturierter Daten*, insbesondere *Textdokumente*, und unterstützen vor allem die Suche nach "relevanten Daten" zu einem spezifizierten Thema.

→ Integrierte DB/IR–Systeme ("Intelligente DBS") sind auf dem Vormarsch und werden insbesondere für Multimedia–Anwendungen von zentraler Bedeutung sein (z.B.: elektronisches Museum, elektronisches Lexikon, "News– und Mail–Filter" u.ä.).

Beispieldokument ("Abstract" eines wissenschaftlichen Zeitschriftenartikels):

Einsatz objektorientierter Datenbanksysteme für multimediale Anwendungen

Dieser Artikel gibt einen Überblick über den Einsatz von objektorientierten Datenbanksystemen für multimediale Anwendungen. Als Beispiel beschreiben wir einen multimedialen Informationsdienst, dessen Inhalte in einem Datenbanksystem verwaltet werden. Dadurch wird ein gleichzeitiger Zugriff der Redakteure und Leser des Dienstes erlaubt. Die Eigenschaften solcher multimedialer Mehr–Benutzer–Systeme werden vorgestellt und im Zusammenhang mit Datenbanksystemen betrachtet. Die unterschiedlichen Möglichkeiten zur Verwaltung multimedialer Daten in heutigen Datenbanksystemen zeigen, welche Eigenschaften jeweils unterstützt werden. Die Modellierung multimedialer Informationen wird am Beispiel des Abstrakten Datentyps "Stummfilm" dargestellt. Hieraus werden Funktionen abgeleitet, die nicht in den Anwendungen realisiert werden sollten, sondern die besser von einem multimedialen Datenbanksystem angeboten werden.

Schlüsselwörter: Multimedia, Datenbanken, Modellierung

Dienste eines IRS

Effiziente Volltextsuche

Suche nach Wörtern oder Phrasen unter Verwendung von Booleschen Kombinationen und von Kontextbedingungen (z.B. Nachbarschaft von Wörtern oder Begrenzung des Abstands im Text).

Beispiel:
((objektorientiertes ADJACENT Datenbanksystem) OR
(postrelationales ADJACENT Datenbanksystem))
AND (Multimedia OR multimedial)

Deskribierung / Indexierung von Dokumenten

(Halb−) Automatische Generierung und Verwaltung von *Deskriptoren* (über explizit angegebene Schlüsselwörter hinaus), die den Inhalt eines Dokuments konzise charakterisieren ("Inhaltserschließung").
Dazu stützt man sich primär auf die folgenden drei Dienste:

Reduktion von (morphologischen Varianten von) Wörtern und Phrasen auf Stammformen mit Hilfe linguistischer Methoden

Beispiele:

1) objektorientierter Datenbanksysteme	→ objektorientiertes Datenbanksystem
	→ Objektorientierung
	→ Datenbanksystem, Datenbank
2) multimedial	→ Multimedia
3) das System verwaltet große Datenmengen	→ Datenverwaltung

Einordnung der Deskriptoren in einen Thesaurus ("Begriffsgerüst")

Zweck: Erkennung von Synonymen, Unter−/Oberbegriffen und sonstigen Verwandtschaftsbeziehungen zwischen Deskriptoren (bis hin zur Wissensbank für das zugrundeliegende "Universe of Discourse")

Beispiele:

1) Film	→ Video
	→ Multimedia
2) objektorientiertes Datenbanksystem	→ postrelationales Datenbanksystem
3) Informationsdienst	→ Informationssystem
4) Datenbanksystem	→ Informationssystem
5) objektorientierte Modellierung	→ Objektmodell
6) Mehr−Benutzer−System	→ Synchronisation

Zuordnung von Gewichten zu den Deskriptoren eines Dokuments

Zweck: Gewichtung der Deskriptoren entsprechend ihrer Relevanz zur Charakterisierung eines Dokuments

Recherchen mit Deskriptoren (unter Verwendung des Thesaurus)

Beispiel:

Benutzeranfrage:
"alles über objektorientierte Datenbanken"

vom System generierte Anfrage:
(objektorientiert OR postrelational OR intelligent) AND
(Datenbank OR Informationssystem)

danach evtl. Einschränkung des Anfrageresultats durch Verfeinerung der Deskriptoren (evtl. (halb−) automatische Anfragemodifikation aufgrund von *Relevanz−Feedback* des Benutzers)

Boolesches Retrieval

Das Resultat einer Anfrage ist eine (ungeordnete) Menge von Dokumenten.

Nicht−Boolesches Retrieval (Ranking)

Das Resultat einer Anfrage ist eine nach Relevanz absteigend geordnete Liste von Dokumenten (Rangbildung; engl.: ranking).
→ Ähnlichkeitssuche, Probabilistisches Retrieval, "vages Suchen"

Bewertung von IRS

Bewertungsmaße:

Fähigkeit, zu einer Anfrage *nur* relevante Dokumente zu liefern:

$$\textit{Präzision} \text{ einer Anfrage (engl.: precision)} = \frac{\textit{Anzahl relevanter Resultatdokumente}}{\textit{Anzahl der Resultatdokumente}}$$

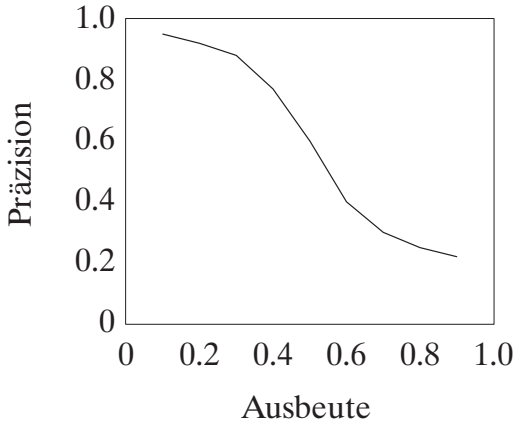
Fähigkeit, zu einer Anfrage *alle* relevanten Dokumente zu liefern:

$$\textit{Ausbeute} \text{ einer Anfrage (engl.: recall)} = \frac{\textit{Anzahl relevanter Resultatdokumente}}{\textit{Anzahl aller relevanten Dokumente}}$$

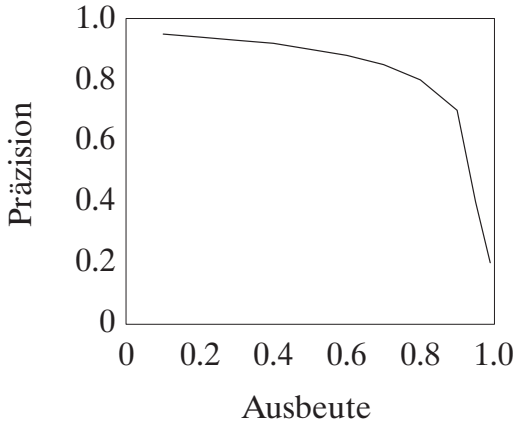
Präzision und Ausbeute eines IRS =
mittlere Präzision und Ausbeute für einen repräsentativen Anfragemix

Zur "Messung" Präzision und Ausbeute wird eine Relevanzbewertung (aller Dokumente) durch den Benutzer benötigt.
→ empirische Untersuchungen mit Testkollektionen

Typisches Verhalten eines IRS



Fast ideales Verhalten eines IRS



Bei nicht−Booleschem Retrieval mit Ranking beziehen sich Präzision und Ausbeute auf die inspizierten Dokumente, also die ersten r Dokumente der Rangliste. Damit sind dann:

$$\textit{Präzision} = \frac{\textit{Anzahl relevanter Dokumente unter den ersten r Dokumenten}}{r}$$

$$\textit{Ausbeute} = \frac{\textit{Anzahl relevanter Dokumente unter den ersten r Dokumenten}}{\textit{Anzahl aller relevanten Dokumente}}$$

16.2 Vektorraummodell für das Information Retrieval mit Ranking

Dokumente und Queries werden als Vektoren in einem von den Deskriptoren (Features, Termen) aufgespannten Vektorraum interpretiert. Zwischen Dokumenten bzw. Dokumenten und einer Query kann dann eine Ähnlichkeitsfunktion auf den Vektoren definiert werden. Anfrageliste sind nach absteigender Ähnlichkeit sortierte Ranglisten.

Gegeben ist also:

- eine Menge von *Features (Deskriptoren, Terme)*
  $f_j \in F$ mit $|F| = m$
- eine (Multi−)Menge von *Dokumenten(vektoren)*
  $d_i \in D \subseteq [0,1]^{|F|}$ bzw. $[-1,1]^{|F|}$ mit $|D| = n$
  (Gewichtung d_{ij} von f_j in d_i)
- eine *Anfrage* $q \in [-1,1]^{|F|}$

Zu einer Query q wird gesucht:

- eine (absteigend) sortierte (Rang−)Liste
  von Dokumenten d_i, die q ähnlich sind

Standardansatz:

- Sortierung nach einem Distanz− bzw.
  Ähnlichkeitsmaß wie zum Beispiel dem *Cosinusmaß*:

$$similarity(d_i, q) = \frac{\sum_{j=1}^{m} d_{ij}\, q_j}{\sum_{j=1}^{m} d_{ij}^2 \sum_{j=1}^{m} q_j^2}$$

Der Wert der Ähnlichkeitsfunktion wird häufig auch als Retrieval Status Value (RSV) oder Score von d_i für q_k bezeichnet.

Beispiel:

F = {Politik, Lafontaine, Sport, Graf}
d1 = <1, 0.9, 0, 0.1>, d2= <1, 0, 0.5, 1>, d3=<0, 0, 1, 1>
q = <1, 0, 0, 1>

similarity (q, d1) ≈0.576,
similarity (q, d2) ≈0.948,
similarity (q, d3) ≈0.5

Also ist die Ergebnisrangliste für q:
1.: d2 2.: d1 3.: d3

Festlegung der Gewichtung von Features in Dokumenten

Heuristisch−pragmatische Ansätze
basierend auf Worthäufigkeiten wie z.B. die *ff*idf−Gewichtung*:

$d_{ij} = ff(f_j, d_i) * idf(f_j)$

mit der *Featurehäufigkeit* (feature frequency)

$ff(f_j, d_i)$ = Häufigkeit von f_j in d_i

und der *inversen Dokumenthäufigkeit* (inverse document frequency)

$$idf(f_j) = \log \frac{n}{Anzahl\ der\ Dokumente\ mit\ f_j}$$

oder inklusive Normierung auf Gewichte zwischen 0 und 1:

$$d_{ij} = (ff(f_j, d_i) * idf(f_j)) / \sqrt{\sum_k ff(f_k, d_i) * idf(f_k)}$$

oder als Variation z.B.:

$$w_{ij} = \frac{Häufigkeit\ von\ f_j\ in\ d_i}{\max_k \{Häufigkeit\ von\ f_k\ in\ d_i\}} \log \frac{n}{Anzahl\ der\ Dokumente\ mit\ f_j}$$

und der Normierung $d_{ij} = w_{ij} / \sqrt{\sum_k w_{ik}^2}$

Solche Gewichtungen können z.T. durch Modelle des Probabilistischen Retrieval formal gerechtfertigt werden.

Festlegung der Gewichtung von Features in Queries

Im einfachsten Fall werden alle (konjunktiv verknüpften) Deskriptoren einer Query mit 1 gewichtet und negierte Deskriptoren ggf. mit −1 (oder Dokumente mit negierten Deskriptoren werden aus dem Gesamtresultat herausgefiltert). Alternativ kann die Reihenfolge, in der die Deskriptoren in der Query angegeben sind, als Gewichtung interpretiert werden. Weitergehende, aber ähnlich heuristisch−pragmatische Ansätze sind denkbar.

Relevanz−Feedback

Nachdem der Benutzer die ersten r Dokumente der Rangliste einer Query inspiziert hat, kann er durch Markieren der relevanten und irrelevanten Dokumente dem IRS Feedback geben, aufgrund dessen in einer weiteren Verfeinerungs− bzw. Präzisierungs−Query explizite Gewichte in der Query gesetzt werden. Im einfachsten Fall kann dies so realisiert werden, daß der Benutzer den besten Treffer auswählt und eine Query startet, die genau die Deskriptoren dieses Dokuments mit seinen entsprechenden Gewichten enthält.

Alternativ könnten − in einer längeren Recherche−Sitzung auch benutzer− bzw. sitzungsspezifische Dokumentgewichte geführt und je nach Feedback angepasst werden, was aber wesentlich aufwendiger zu implementieren wäre.

Latent−semantische Indexierung (Latent Semantic Indexing, LSI)

Das einfache Vektorraum−Modell ist in (mindestens) zweierlei Hinsicht zu kritisieren:

1) Da es u.U. starke Korrelationen im Vorkommen verschiedener Features in den Dokumenten gibt, ist die Dimensionalität des Feature−Vektorraums eigentlich unnötig hoch.
   Beispiel:
   Wenn "Web" und "Internet" nahezu immer zusammen vorkommen,
   braucht man nur eines dieser beiden Features zu indexieren.

2) Dasselbe Feature kann je nach vorkommender Kombination mit anderen Features eine andere Semantik haben, so daß solche Korrelation explizit berücksichtigt sein sollten.
   Beispiel:
   Wenn "Java" und "Library" zusammen auftreten, ist die Bedeutung von "Java" vermutlich die Programmiersprache Java; das Dokument behandelt, also im weiteren Sinne das Thema Internet. Wenn "Java", "Kona Blend" und "Mokka" zusammen auftreten, handelt das Dokument mit hoher Wahrscheinlichkeit von Kaffee. Wenn schließlich "Java", "Sumatra" und "Borneo" zusammen auftreten, betrifft das Dokument vermutlich das Land Indonesien.

Die Idee von LSI ist, solche Korrelationen auszunutzen, um von den Featurekombinationen auf Themen (Topics) bzw. Konzepte (Concepts) zu schließen, die den Inhalt von Dokumenten charakterisieren. Die Anzahl verschiedener Themen ist typischerweise deutlich kleiner als die der Features. Wenn man das Vektorraum−Modell auf Themen anwendet statt auf Features, erzeugen die Themen einen Vektorraum mit deutlich niedrigerer Dimensionalität. In den Themen kommt also die "latente Semantik" der Dokumente klarer zum Ausdruck.

Gegeben seien:
− eine Featuremenge F mit $|F| = m$
− eine Dokumentmenge D mit $|D|=n$ und $D \subseteq [0,1]^m$
− eine Query $q \in [-1,1]^m$

D kann also als nxm−Matrix A mit Werten aus dem Intervall [0,1] interpretiert werden und q als 1xm−(Zeilen)Vektor.

Gesucht wird:
− eine "möglichst gute" Abbildung von D und q in einen Themen−Vektorraum mit Dimensionalität $k \ll m$

Lösung:
Die Lösung besteht aus einer auf der sogenannten Singulärwertdekomposition von A beruhenden Transformation von A und q in einen automatisch "erzeugten" Vektorraum niedrigerer Dimensionalität. Dabei werden die folgenden mathematischen Eigenschaften der Singulärwertdekomposition ausgenutzt.

Satz:
Jede reellwertige nxm−Matrix A mit Rang r kann zerlegt werden in die Form

$$A = U \times \Delta \times V^T$$

mit einer nxr−Matrix U mit orthonormalen Spaltenvektoren, einer rxr−Diagonalmatrix Δ und einer mxr−Matrix V mit orthonormalen Spaltenvektoren.
Diese Zerlegung heißt *Singulärwertdekomposition* (engl.: Singular Value Decomposition, kurz: SVD) und ist unter der Annahme, daß die Elemente von D geordnet sind, eindeutig bestimmt.

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 Eigenwerten von $A^T \times A$,
− die Spaltenvektoren von U sind die Eigenvektoren von $A \times A^T$,
− die Zeilenvektoren von V sind die Eigenvektoren von $A^T \times A$.

Dabei kann $A \times A^T$ als Dokument−Dokument−Ähnlichkeitsmatrix interpretiert werden und $A^T \times A$ als Feature−Feature−Ähnlichkeitsmatrix.
U ist als Dokument−Thema−Ähnlichkeitsmatrix und V als Feature−Thema−Ähnlichkeitsmatrix zu interpretieren.

Indexierung von D und Ausführung von Queries q:

Als Index zu verwalten ist die Dokument−Thema−Ähnlichkeitsmatrix U sowie − quasi als Domänenwissen − die Feature−Thema−Ähnlichkeitsmatrix V. Letztere kann man weitgehend statisch berechnen, sobald man eine repräsentative, große Dokumentenkollektion hat. Ein neu eingefügtes Dokument d (d.h. ein 1xm−Zeilenvektor) muß dann zur Indexierung in den Themen−Vektorraum abgebildet werden, und zwar durch die Transformation d' = d x V, und der resultierende 1xr−Zeilenvektor d' wird zum Index hinzugefügt (d.h. konzeptionell wird U um eine Zeile erweitert).

Eine Query q, die als mx1−Spaltenvektor aufgefaßt werden kann, wird dann zunächst in eine Query q' in den Themen−Vektorraum transformiert, und zwar durch $q' = V^T \times q$. Dann wird q' aufgrund des Index U evaluiert (mit der verwendeten Ähnlichkeitsfunktion des r−dimensionalen Themen−Vektorraums), und die dadurch erzeugte Rangliste von Dokumenten ist das Anfrageresultat. Im einfachsten Fall würde z.B. das Cosinus−Maß oder die Euklidische Distanz zwischen q' und den Zeilen von U (also den Dokumenten) verwendet.

Beispiel:

Die Dokument−Feature−Ähnlichkeitsmatrix A sei wie folgt gegeben (der Einfachheit halber seien hier auch Gewichte größer als 1 möglich):

Features / Dokumente	Lafontaine	Schröder	Euro	Uefa	Beckenbauer
d1	1	1	1	0	0
d2	2	2	2	0	0
d3	1	1	1	0	0
d4	5	5	5	0	0
d5	0	0	0	2	2
d6	0	0	0	3	3
d7	0	0	0	1	1

Die SVD ermittelt daraus:

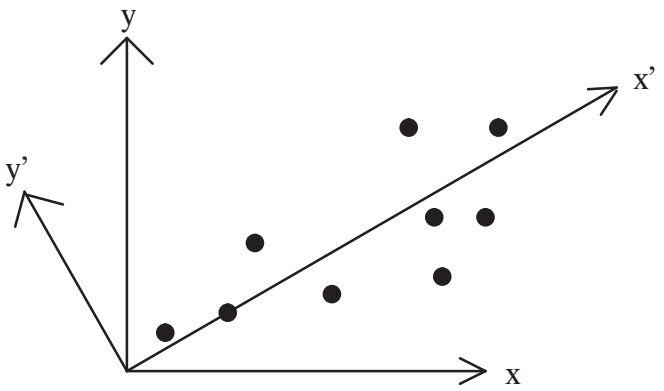
$$U = \begin{pmatrix} 0.18 & 0.00 \\ 0.36 & 0.00 \\ 0.18 & 0.00 \\ 0.90 & 0.00 \\ 0.00 & 0.53 \\ 0.00 & 0.80 \\ 0.00 & 0.27 \end{pmatrix} \qquad \Delta = \begin{pmatrix} 9.64 & 0.00 \\ 0.00 & 5.29 \end{pmatrix} \qquad V^T = \begin{pmatrix} 0.58 & 0.58 & 0.58 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.71 & 0.71 \end{pmatrix}$$

Intuitiv enthält die Dokumentenkollektion also zwei Themen, nämlich Politik und Sport. Ein weiteres Dokument d8 = (1 1 0 0 0) mit dem einzigen Feature "Lafontaine" würde dann auf den Themenvektor d8 x V= (0.54 0).
Eine Query $q = (0\,0\,1\,0\,0)^T$, die nach dem Feature "Euro" sucht, würde auf den Themenvektor $V^T \times q = (0.58 \;\; 0)^T$ abgebildet. Unabhängig vom wirklich verwendeten Ähnlichkeitsmaß im Themenvektorraum wäre also d8 vermutlich ein sehr guter Treffer für q.

Wenn der Rang r von A nicht hinreichend klein ist, also die Anzahl der "Themen" zu groß ist, kann man die Dimensionalität des Themen−Vektorraums künstlich auf k < r beschränken, indem man nur die k größten Singulärwerte von A und die zugehörigen Eigenvektoren betrachtet. Dadurch berücksichtigt man die "ausgeprägtesten" Themen und "stärksten" Zusammenhänge zwischen Features und Themen, was durch den folgenden Satz formal untermauert wird.

Satz:
Sei A eine nxm−Matrix mit Rang r und sei $A_k = U_k \times \Delta_k \times V_k^T$, wobei die kxk−Diagonalmatrix Δ_k die k größten Singulärwerte von A enthält und die nxk−Matrix U_k und die mxk−Matrix V_k aus den zugehörigen Eigenvektoren der Singulärwertdekomposition von A bestehen.
Unter allen nxm−Matrizen C mit einem Rang, der nicht größer als k ist, ist A_k diejenige Matrix, die den Wert $\|A-C\|_F^2 = \sum_{i=1}^{n}\sum_{j=1}^{m}(A_{ij}-C_{ij})^2$ minimiert (die Frobenius−Norm).

Zur Illustration:
Die SVD kann als affine Transformation der Dokumentvektoren aufgefaßt werden. Bei der Beschränkung auf die k größten Singulärwerte erfolgt quasi eine Projektion auf die k aussagekräftigsten Dimensionen. Wenn wir in dem folgenden Bild mit m=2 den Wert k=1 wählen, ist offensichtlich die Projektion auf x' besser als die auf y'. Tatsächlich erreicht die Wahl der k größten Singulärwerte genau diesen Effekt.



Bei der Indexierung von Dokumenten und der Ausführung von Queries im − nunmehr k−dimensionalen Themenreaum − spielen dann einfach U_k und V_k die Rolle von U und V.

16.3 Such−Engines im WWW

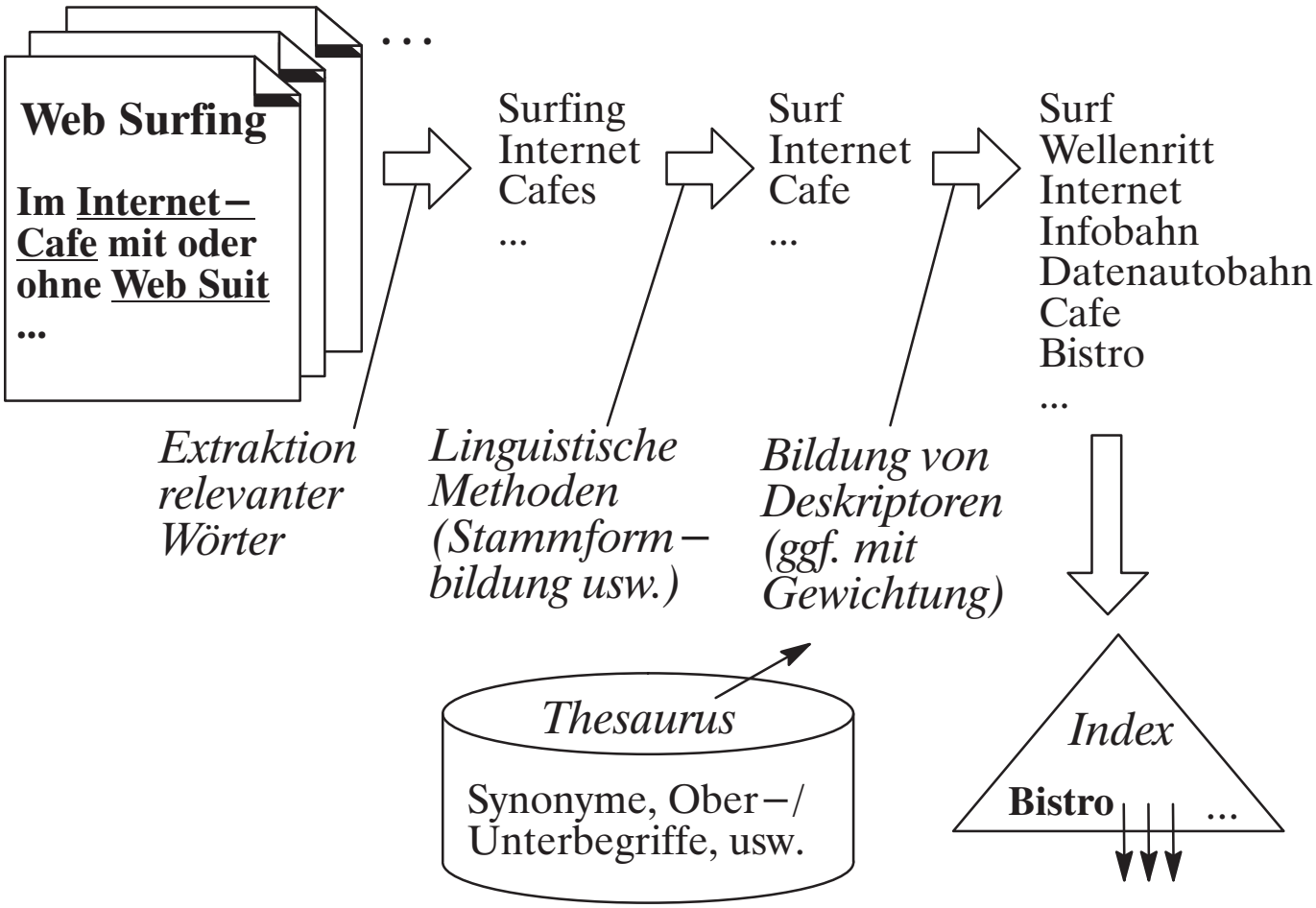
Such−Engines wie AltaVista, Lycos, Infoseek, Excite, HotBot usw. basieren auf dem Vektorraum−Modell des Information Retrieval. Features sind aus HTML−Seiten extrahierte und ggf. auf Stammformen reduzierte Wörter, ggf. in Verbindung mit bestimmten HTML−Tags (z.B. Title oder Author) oder URLs (der Seite selbst oder der in ihr enthaltenen Links). Die Gewichtungen der Features in Dokumenten orientieren sich wie beim klassischen Information Retrieval an Worthäufigkeiten sowie ggf. zusätzlich an der "Popularität" von Dokumenten im Web, konkret z.B. der Anzahl der eingehenden Links oder der Anzahl der Hits bei allen Web−Queries eines Zeitraums.

Indizes sind in der Regel als B*−Bäume (oder eine andere Art von Suchbäumen) implementiert, deren Blätter Features mit jeweils einer Liste von URLs und Gewichten enthalten. Der Indizes der Such−Engines werden typischerweise durch periodisches oder kontinuierliches "Traversieren" des Webs (von sog. Gatherers, Crawlers oder Spiders) aktualisiert.

Alternativ zu dieser Art von IR−basierten Such−Engines gibt es auch Such−Engines wie Yahoo u.ä., die auf einer expliziten, intellektuell erstellten Ontologie von spezifischen Domänen (z.B. Reisen) beruhen, in der Begriffshierarchien und andere semantische Beziehungen repräsentiert werden. Die Indexierung von HTML−Seiten wird dabei typischerweise bestenfalls semi−automatisch sein.

Es gibt auch eine ganze Reihe von Broker−Engines wie z.B. MetaCrawler oder ProFusion, die eine Query auf mehreren Such−Engines gleichzeitig ausführen oder die für die jeweilige Query beste Such−Engine auswählen. Das dabei auftretende Problem, die Ranglisten verschiedener Such−Engines systematisch zu mischen, so daß eine gesamthaft nach Ähnlichkeit bzw. Relevanz absteigend sortierte Liste erzeugt wird, ist Gegenstand der aktuellen Forschung.

Illustration der Indexierung von HTML−Seiten durch Web−Search−Engines



16.4 Suche in multimedialen Daten

Eine etablierte Vorgehensweise bei der Suche in multimedialen Datenbeständen – Bilder, Audiosequenzen, Videos oder kombinierten Dokumenten beispielsweise einer digitalen Bibliothek – ist die Vorgabe eines Query–Objekts als Referenz, z.B. einer Fotografie einer Wüstenlandschaft, und das daraufhin initiierte Retrieval ähnlicher Datenobjekte, also z.B. aller Fotografien von Wüstenlandschaften. Diese Vorgehensweise kann man als direkte Verallgemeinerung des Vektorraummodells für Information Retrieval auffassen.

16.4.1 Ähnlichkeitsmaße

Jedem multimedialen Datenobjekt werden bestimmte Features aufgrund bestimmter automatisierter Analysen zugeordnet, z.B. eine Farbverteilung, Parameter, die die Textur (Körnigkeit, Orientierung, usw.) von Bildern bzw. Bildteilen beschreiben, oder die Amplituden bestimmter Frequenzbereiche in Audio–Sequenzen. Eine sinnvolle Feature–Bestimmung ist hochgradig abhängig von der jeweiligen Anwendung.

Datenobjekte und Queries sind dann als Vektoren eines hochdimensionalen Feature–Raums beschreibbar. Mit der Festlegung eines geeigneten Distanzmaßes bzw. Ähnlichkeitsmaßes zwischen Vektoren können nun Ähnlichkeitsqueries mit einem Ranking der Resultate ausgeführt werden; ggf. kann eine zu spezifizierende obere Grenze für die erlaubte Distanz der Resultate zur Query zum Abschneiden der Rangliste dienen.

Beispiel 1: Audio–Signale

Ein (Audio–) Signal kann im einfachsten Fall als mit einer bestimmten Frequenz diskret abgetastete Zeitreihe der Amplitude aufgefaßt werden. Seien x, y solche Signale, also mx1–Spaltenvektoren. Eine mögliche Distanzfunktion ist dann die Euklidische Distanz bzw. deren Quadrat

$$\text{distance}\,(x, y) = (x-y)(x-y)^T = \sum_{i=1}^{m} (x_i - y_i)^2$$

Beispiel 2: Farbbilder

Bei Farbbildern kann man die Häufigkeit der verschiedenen Farben unter den Pixeln des Bildes, ggf. in verschiedenen Bildteilen (z.B. Mitte, oberer Rand, unterer Rand, usw.), zur Grundlage eines Feature–Raums bzw. eines Distanzmaßes machen. Sei m die Anzahl der möglichen Farben. In einer mxm–Matrix A sei die Ähnlichkeit verschiedener Farben untereinander mit Werten zwischen 0 und 1 ausgedrückt (z.B. sind Orange und Gelb ähnlich). Objekte und Queries sind dann mx1–Spaltenvektoren mit Werten zwischen 0 und 1 (also relativen Häufigkeiten der verschiedenen Farben). Die Distanz zwischen zwei Objekten x und y bzw. einem Objekt und einer Query kann dann sinnvoll durch die folgende quadratische Form beschrieben werden:

$$\text{distance}\,(x, y) = (x-y)^T\, A\, (x-y) = \sum_{i=1}^{m} \sum_{j=1}^{m} A_{ij}\, (x_i - y_i)\, (x_j - y_j)$$

Nicht immer können derart quantitative Features explizit bestimmt werden. Unter Umständen ist die Distanz bzw. Ähnlichkeit zwischen zwei Objekten nur durch einen mehr oder weniger komplexen Algorithmus beschreibbar. Wünschenswert ist dann, daß die Objekte zumindest als Elemente eines metrischen Raums aufgefaßt werden können, daß also für je drei Objekte x, y und z die Dreiecksungleichung

$$\text{distance}(x,z) \leq \text{distance}(x,y) + \text{distance}(y,z)$$

gilt.
Dies würde z.B. bei der Levenshtein−Distanz für Strings gelten, die die minimale Anzahl von Editieroperationen (Zeichen austauschen, einfügen oder löschen) angibt, um einen String in einen anderen zu überführen. Bei realen Anwendungen wie z.B. dem algorithmischen Bestimmen der Ähnlichkeit von Fingerabdrücken kann die Gültigkeit der Dreiecksungleichung nicht a priori angenommen werden.

16.4.2 Effiziente Ähnlichkeitssuche

Ein direkter Algorithmus zur Ähnlichkeitssuche würde jedes Objekt der gegebenen Datenkollektion mit der Query vergleichen, die Ähnlichkeit berechnen und am Ende die Objekte nach absteigender Ähnlichkeit sortieren. Dabei treten zwei Engpässe auf:

- die potentiell sehr teure Auswertung einer u.U.komplexen Ähnlichkeitsfunktion und
- der Scan über alle Objekte.

Zur Eliminierung bzw. Entschärfung dieser Engpässe werden effizient berechenbare Schranken für das Ähnlichkeitsmaß als Filter und mehrdimensionale Indexstrukturen zur Reduktion der relevanten Objektmenge verwendet.

Ähnlichkeits−Filter

Für einen Feature−Raum oder Objekt−Raum F gilt es, eine effizient berechenbare Funktion $f: F \times F \rightarrow [0,1]$ zu finden, so daß für alle $x, y \in F$ und alle ε gilt:

$$\text{distance}(x,y) < \varepsilon \Rightarrow f(x,y) < \varepsilon ,$$

also: $f(x,y) \leq \text{distance}(x,y)$

Bei einer Ähnlichkeitssuche mit Query−Objekt q und Schwellwertdistanz ε können dann zunächst diejenigen Objekte y bestimmt werden, für die gilt $f(q,y) < \varepsilon$. Dies ist eine Obermenge der echten Treffermenge, eliminiert aber (hoffentlich) bereits den größten Teil der Nichttreffer. Auf die Objekte dieser herausgefilterten Obermenge muß dann die wirkliche Ähnlichkeitsfunktion “distance” angewandt werden.

Ähnlichkeitsfilter für Beispiel 1:

Ein möglicher Ähnlichkeitsfilter ergibt sich aus der Diskreten Fourier−Transformation (DFT) der Signalvektoren x und y, die durch die folgende Sequenz komplexer Zahlen definiert ist:

$$\hat{x}_i = \frac{1}{\sqrt{m}} \sum_{k=1}^{m} x_i \, e^{-j2\pi ki/m} = \frac{1}{\sqrt{m}} \sum_{k=1}^{m} x_i \, \cos(-2\pi ki/m) + j \, \sin(-2\pi ki/m)$$

mit $j = \sqrt{-1}$, i= 1, 2, ..., m

Aus $\hat{x}$ läßt sich x durch die inverse Transformation ohne Verlust rekonstruieren:

$$x_i = \frac{1}{\sqrt{m}} \sum_{k=1}^{m} \hat{x}_k \, e^{-j2\pi ki/m}$$

Eine wichtige Eigenschaft der DFT ist das folgende Theorem von Parseval:

$$\|x\|^2 = \sum_{i=1}^{m} |x_i|^2 = \sum_{k=1}^{m} |\hat{x}_k|^2 = \|\hat{x}\|^2$$

Damit kann man zeigen, daß eine auf der Basis der ersten h Glieder der DFT berechnete Distanz f eine (brauchbare) untere Schranke für die wirkliche Distanz liefert:

$$f(x,y) = \sum_{k=1}^{h} |\hat{x}_k - \hat{y}_k|^2 \leq \sum_{k=1}^{m} |\hat{x}_k - \hat{y}_k|^2 = \sum_{i=1}^{m} |x_i - y_i|^2 = \text{distance}(x,y)$$

Ähnlichkeitsfilter für Beispiel 2:

Anstelle der kompletten Farbverteilungsinformation kann man die mittleren Rot−, Grün− und Blauanteile eines Bildes für eine Ähnlichkeitsfunktion f verwenden. Sei n die Anzahl der Pixel eines Bildes x. Die Farbe jedes Pixels p ist aus Anteilen R(p), G(p), B(p) der Farben Rot, Grün und Blau synthetisiert. Die mittleren RGB−Anteile des Bildes x sind dann:

$$R(x) = \frac{1}{N} \sum_{p=1}^{N} R(p) \qquad G(x) = \frac{1}{N} \sum_{p=1}^{N} G(p) \qquad B(x) = \frac{1}{N} \sum_{p=1}^{N} B(p)$$

Daraus ergibt sich eine RGB−basierte Ähnlichkeitsfunktion z.B. als (quadrierte) Euklidische Distanz der mittleren RGB−Anteile von Bildern:

$f(x, y) = (R(x) - R(y))^2 + (G(x) - G(y))^2 + (B(x) + B(y))^2$

Entscheidend ist nun, daß man − mit einigen algebraischen Umformungen − zeigen kann, daß gilt:

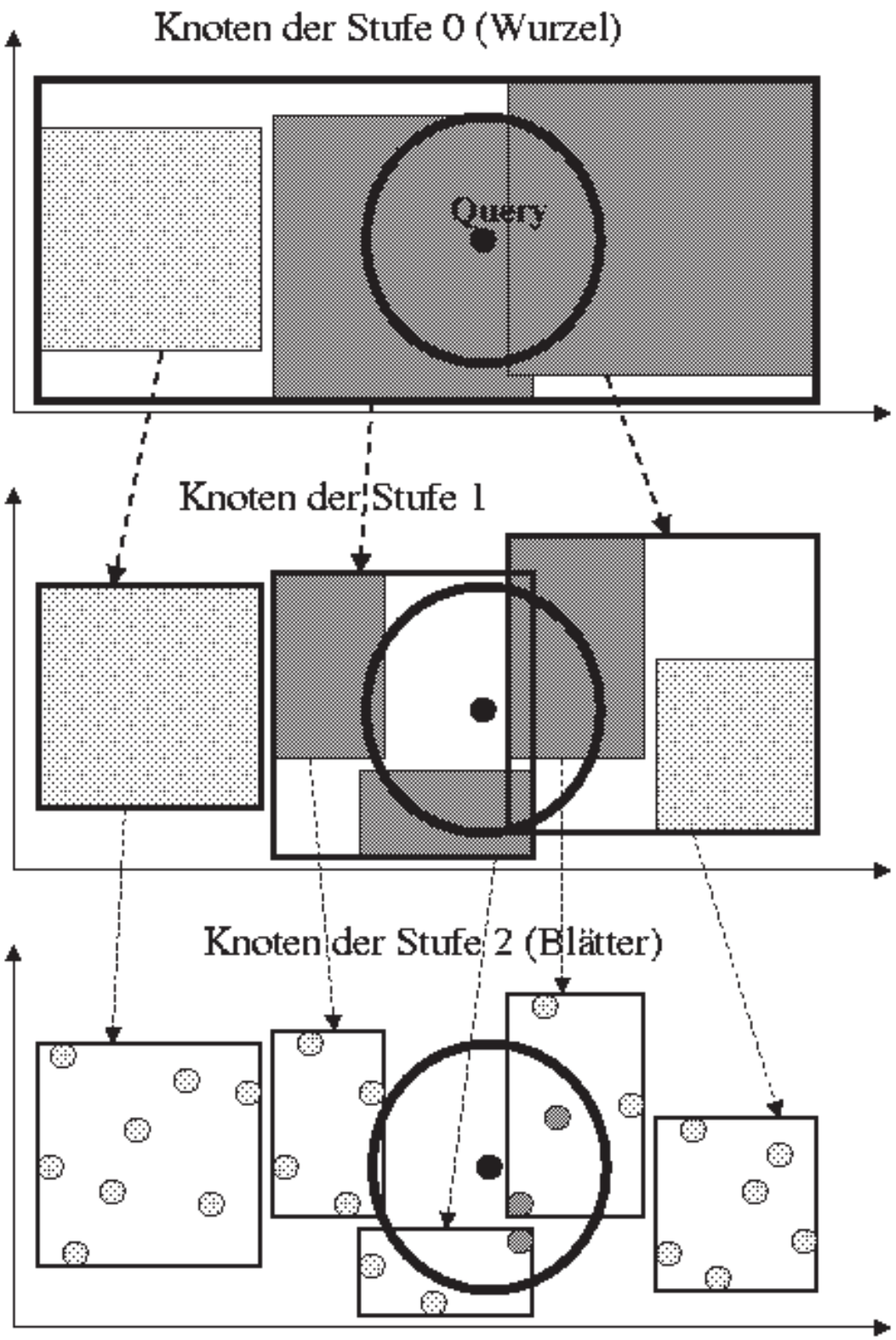
$f(x,y) \leq a * \text{distance}(x,y)$ mit einer von A abhängigen Konstanten a

Indexstrukturen für die Ähnlichkeitssuche

Wenn alle Objekte in einem mehrdimensionalen Index, z.B. einem R−Baum, als Featurevektoren repräsentiert sind (ggf. nach einer Transformation auf einen niederdimensionalen Raum, auf dem eine Filterung durchgeführt wird), kann man zunächst einmal eine komplette Traversierung des Index initiieren. Bevor man dabei in einen Teilbaum hinabsteigt, berechnet man aufgrund der Wegweiser−Information für den Teilbaum (z.B. des kleinsten umgebenden Rechtecks aller im Teilbaum liegenden Objekte) die kürzeste, überhaupt mögliche Distanz zwischen einem Objekt des Teilbaums und dem Query−Objekt. Nur wenn diese Distanz hinreichend klein ist, initiiert man die Traversierung des Teilbaums; ansonsten kann man diesen Teilbaum auslassen.

Diese Vorgehensweise funktioniert grundsätzlich mit verschiedenen Distanzmaßen. Bei Verwendung eines R−Baum−artigen Index ist man aber auf Vektorraum−Normen als Distanzmaße beschränkt. Für Ähnlichkeitsfunktionen, die auf keinem expliziten Feature−Raum basieren, aber immer noch die Dreiecksungleichung erfüllen, wurden in der Literatur sogenannte ”metrische Bäume” vorgeschlagen. Diese speichern für jeden Teilbaum zusammen mit der Wegweiser−Information im entsprechenden Vaterknoten ein Referenzobjekt des Teilbaums zusammen mit der maximalen Distanz aller im Teilbaum enthaltenen Objekte zum Referenzobjekt. Diese Information in Verbindung mit der Dreiecksungleichung kann bei der Indextraversierung ausgenutzt werden, um Teilbäume vom relevanten Suchraum auszuschließen.

Ähnlichkeitssuche mit Euklidischer Distanz auf einem R−Baum



Ergänzende Literatur zu Kapitel 16:

G. Salton, M.J. McGill, Information Retrieval – Grundlegendes für Informationswissenschaftler, McGraw–Hill, 1987

G. Salton, Automatic Text Processing, Addison–Wesley, 1988

W.B. Frakes, R. Baeza–Yates (Editors), Information Retrieval – Data Structures & Algorithms, Prentice Hall, 1992

N. Fuhr, Information Retrieval, Vorlesungsmanuskript, Universität Dortmund, http://ls6–www.cs.uni–dortmund.de/ir/teaching/courses/ir

K. Sparck–Jones, P. Willett (Editors), Readings in Information Retrieval, Morgan Kaufmann Publishers, 1997

V.N. Gudivada, V.V. Raghavan, W.I. Grosky, R. Kasanagottu, Information Retrieval on the World Wide Web, IEEE Internet Computing Vol.1 No.5, September 1997, pp. 58–68

P. Schäuble, Multimedia Information Retrieval, Kluwer Academic Publishers, 1997

J. Hellerstein (Editor), IEEE Data Engineering Bulletin Vol. 20 No. 4, December 1997, Special Issue on Data Reduction Techniques

C. Faloutsos, Searching Multimedia Databases by Content, Kluwer Academic Publishers, 1996

C. Faloutsos, Indexing of Multimedia Data, in: P. Apers, H. Blanken, M. Houtsma (Editors), Multimedia Databases in Perspective, Springer, 1997