Andreas Henrich

Information Retrieval 1

Grundlagen, Modelle und Anwendungen

Version: 1.2 (Rev: 5727, Stand: 7. Januar 2008)

Otto-Friedrich-Universität Bamberg
Lehrstuhl für Medieninformatik, 2001 – 2008

Vorwort zur Online-Ausgabe

Der vorliegende Lehrtext ist in seiner ursprünglichen Form um das Jahr 2000 entstanden und seitdem kontinuierlich weiterentwickelt worden. Er war und ist Grundlage diverser Lehrveranstaltungen des Lehrstuhls für Medieninformatik (http://www.uni-bamberg.de/minf) an der Otto-Friedrich-Universität Bamberg.

Für die Unterstützung bei der Erstellung und Weiterentwicklung des Lehrtexts gilt mein Dank insbesondere Dr. Karlheinz Morgenroth, Dr. Günter Robbert, Daniel Blank, Martin Eisenhardt, Dr. Wolfgang Müller, Sven-Uwe Wolf und Sebastian Gaßner. In einem von der Virtuellen Hochschule Bayern (http://www.vhb.org) geförderten Projekt wurde zu dem vorliegenden Text eine Lernumgebung mit Selbsttestaufgabben, Applets etc. entwickelt.

Wir haben mit diesem Buch gute Erfahrungen gemacht und möchten auch anderen Interessierten die Möglichkeit geben, die in diesem Lehrtext präsentierten Materialien zu nutzen – sei es zur eigenen Lektüre oder als Grundlage oder Ergänzung für eine Lehrveranstaltung bzw. einen Kurs.

Um dem Gedanken des freien Austauschs von Wissen zu entsprechen, haben wir uns nach reiflicher Überlegung dazu entschlossen, dieses Buch unter einer *Creative Commons*-Lizenz zu publizieren, speziell unter der Variante *by-nc-nd* (http://creativecommons.org).

Sie dürfen dieses Buch also in unveränderter Form elektronisch oder papiergebunden weitergeben, und Sie dürfen es für jeden nicht-kommerziellen Zweck verwenden. Änderungen an dem Buch behalten wir uns vor und möchten auch als Autoren des Werkes genannt werden.

Natürlich kann kein Buch dieses Umfangs den Anspruch auf Fehler- und Widerspruchsfreiheit für sich erheben; so gehen wir auch davon aus, dass im vorliegenden Text noch tatsächliche fachliche Fehler, sprachliche Ungenauigkeiten, Tippfehler und andere Imperfektionen enthalten sind.

Wir haben in der Vergangenheit, während der Genese dieses Buches, sehr gute Erfahrungen damit gemacht, die Leser und Nutzer des Buches um Kommentare, Anmerkungen und Fehlermeldungen zu bitten. Auf diese Weise konnten und können wir das Buch immer weiter entwickeln und nach und nach die noch verbliebenen Fehler ausmerzen. Auch Hinweise auf eventuell noch nicht abgedeckte, für das Verständnis der dargestellten Inhalte aber wichtige Themen sind uns willkommen. Dabei sei allerdings

darauf hingewiesen, dass die Lehrveranstaltungen zum Information Retrieval in Bamberg aus einen Kurs Information Retrieval 1 (Grundlagen, Modelle und Anwendungen) – basierend auf diesem Text – und einem Kurs Information Retrieval 2 (ausgewählte weiterführende Themen) – zu dem es keinen in dieser Form ausgearbeiteten Lehrtext gibt – bestehen. Das vorliegende Buch hat daher eher einführenden Charakter.

Für Anmerkungen, Kommentare, Hinweise, kurz: »bug reports« aller Art besuchen Sie bitte http://www.uni-bamberg.de/minf/IR1-Buch.

Wir möchten den Lesern nun bei der Lektüre des Buches viel Spaß und möglichst großen Wissensgewinn wünschen!

Andreas Henrich Bamberg, 7. Januar 2008

Inhaltsverzeichnis

1 Motivation und Einführung **11**

1.1 Der Begriff Information Retrieval (IR) 18
1.1.1 Eine erste Begriffserklärung 18
1.1.2 Weitere Begriffserklärungen 22
1.1.3 Beispielszenarium für den Einsatz eines IR-Systems 26
1.1.4 Die historische Sicht und neuere Entwicklungen . . 27
1.2 Aufgabenstellungen für IR-Systeme 31
1.3 Abgrenzung von IR und Fakten Retrieval 33
1.4 Das grundsätzliche Vorgehen beim Information Retrieval . 40
1.4.1 Boolesches Retrieval 41
1.4.2 Das Vektorraummodell 45
1.4.3 Probabilistisches Information Retrieval 52
1.5 Formulierung eines Informationswunsches 53
1.6 Darstellung und Beurteilung von Suchergebnissen 54
1.7 Zusammenfassung und Aufbau des Kurses 57

2 Evaluierung und Beurteilung von IR-Systemen **59**

2.1 Begriffsbildung 61
2.1.1 Effizienz vs. Effektivität 61
2.1.2 Relevanz 62
2.2 Recall und Precision 63
2.2.1 Vergleich mehrerer IR-Systeme 67
2.2.2 Bestimmung des Recall 68
2.2.3 Mittelwertbildung über mehrere Anfragen 71
2.2.4 Recall/Precision-Werte bei Systemen mit Ranking 74
2.2.5 Vergleich mehrerer Systeme, die ein Ranking liefern 77
2.3 Experimente und Testkollektionen 78
2.3.1 Die TREC-Kollektion 79
2.3.2 Die CACM- und CISI-Kollektionen 86
2.4 Kriterienkataloge zur Evaluierung von IR-Systemen . . . 88
2.5 Zusammenfassung 90

3 Berücksichtigung der Vagheit in Sprache 93
3.1 Ein einführendes Beispiel . . . 94
3.2 Stoppworteliminierung . . . 95
3.3 Stamm- und/oder Grundformreduktion . . . 98
3.3.1 Grund- und Stammformreduktion . . . 101
3.3.2 Verfahren zur Grund- und Stammformreduktion . 102
3.3.3 Verfahren auf Basis einfacher Trunkierung . . . 103
3.3.4 Lovins-Algorithmus zur Grundformreduktion . . . 105
3.3.5 Verfahren, die auf Wörterbüchern basieren . . . 109
3.4 Mehrwortgruppenidentifikation . . . 110
3.4.1 Ermittlung von Mehrwortgruppen . . . 111
3.4.2 Darmstädter Indexierungsansatz . . . 112
3.4.3 Ein Nachsatz zur Begriffswelt . . . 114
3.5 Terminologische Kontrolle . . . 115
3.6 Ein sprachunabhängiger Ansatz . . . 120
3.7 Auszeichnung mittels Metadaten . . . 122
3.7.1 Dublin Core . . . 122
3.7.2 Semantic Web . . . 125
3.7.3 RDF . . . 127
3.8 Zusammenfassung . . . 135

4 Einfache IR-Modelle und ihre Implementierung 137
4.1 Pattern-Matching . . . 138
4.2 Wortsuche . . . 139
4.2.1 Invertierte Listen . . . 140
4.2.2 Signaturen . . . 149
4.2.3 Überlagerungsfähige Signaturen . . . 152
4.2.4 Speicherungsstrukturen für Signaturdateien . . . 161
4.3 Boolesches Retrieval . . . 170
4.4 Coordination-Level-Match . . . 176
4.5 Fuzzy-Set Modell . . . 178
4.6 Zusammenfassung . . . 183

5 Das Vektorraummodell 185
5.1 Die Grundidee des Vektorraummodells . . . 185
5.2 Die Basisformeln zum Vektorraummodell . . . 187
5.3 Eine Variante der tf·idf-Formel . . . 195
5.4 Relevance Feedback . . . 199
5.5 Implementierung des Vektorraummodells . . . 204
5.5.1 Basisalgorithmus mit invertierten Listen . . . 204
5.5.2 Steigerung der Performance . . . 212

5.6 Zusammenfassung 214

6 Alternativen zur globalen Suche 217
6.1 Klassifikationen 218
6.1.1 Die Internationale Patentklassifikation 220
6.1.2 Die Dezimalklassifikation 223
6.1.3 Kataloge im Internet 227
6.2 Cluster-Ansätze 232
6.2.1 Ähnlichkeitsmaße und die Ähnlichkeitsmatrix . . . 233
6.2.2 Nichthierarchische Ansätze zur Clusterbildung . . 236
6.2.3 Hierarchische Ansätze zur Clusterbildung 237
6.2.4 Anwendungsgebiete für Cluster-Verfahren 243
6.3 Browsing 247
6.3.1 Begriff und Arten des Browsing 247
6.3.2 Visualisierung von Dokumentenkollektionen 249
6.3.3 Beispiele für die Visualisierung 250
6.4 Zusammenfassung 259

7 Weitere IR-Modelle 263
7.1 Das Binary Independence Retrieval-Modell (BIR-Modell) 263
7.2 Okapi BM 25 280
7.3 Latent Semantic Indexing 282
7.3.1 Einführung 282
7.3.2 Ein Beispiel zur Motivation 283
7.3.3 Die Idee des Latent Semantic Indexing 285
7.3.4 Anwendung der SVD 288
7.4 Zusammenfassung 290

8 Multimedia Information Retrieval 291
8.1 Einleitung 291
8.2 Allgemeine Überlegungen 292
8.3 Bild-Retrieval 295
8.3.1 Farbe 296
8.3.2 Textur 303
8.3.3 Segmentierung 307
8.3.4 SIFT – Scale Invariant Feature Transform 316
8.4 Audio Retrieval 325
8.4.1 Audio Features 327
8.4.2 Techniken für das Audio Retrieval 330
8.4.3 Techniken zur Grundklassifikation 331
8.5 Video Retrieval 332

8.5.1 Videosegmentierung mittels Shot-Detection 333
8.5.2 Kamerabewegung (Motion Detection) 334
8.5.3 Key Frame Extraction 334
8.6 Retrieval für Multimedia-Dokumente 335
8.7 Zusammenfassung 347

9 Information Retrieval und das World Wide Web 349
9.1 Einführung 350
9.2 Aufbau einer Suchmaschine 350
9.3 Begriffsdefinitionen 352
9.3.1 Internet, Intranet und Extranet 352
9.3.2 Hypertext 354
9.3.3 Statische und dynamische Dokumente 356
9.3.4 Deep Web und Shallow Web 358
9.3.5 Breiten- und Tiefensuche 359
9.3.6 Crawler, Spider, Robot 360
9.4 Techniken des Crawling 362
9.4.1 Funktionsweise eines Crawlers 362
9.4.2 Der Robots Exclusion Standard 363
9.4.3 Indexierung von HTML-Dokumenten 365
9.4.4 Indexierung anderer Dokumenttypen 369
9.4.5 Prioritätsgeführtes Crawling 370
9.4.6 Probleme beim Crawlen 371
9.5 Typische Funktionen einer Suchmaschine 374
9.5.1 Einfache Suche mit Suchbegriffen 374
9.5.2 Suche nach Phrasen 376
9.5.3 Einschränkung des Suchergebnisses 377
9.5.4 Web Directories 379
9.6 Erweiterte Funktionen 380
9.6.1 Suche im Internet außerhalb des WWW 381
9.6.2 Suche nach Bildern 383
9.6.3 Metasuchmaschinen 385
9.7 Spezielle Verfahren im Web Information Retrieval 386
9.7.1 PageRank™-Algorithmus 387
9.7.2 Sponsored Links 393
9.7.3 Bewertungssysteme 395
9.8 Probleme bei Suchmaschinen 395
9.8.1 Staatliche und nicht-staatliche Zensur 396
9.8.2 Veraltete Indexe 397
9.8.3 Weblogs und Co. 397
9.9 Zusammenfassung 398

10 Neuere Entwicklungen und innovative Anwendungen 403

Literaturverzeichnis 405

Index 414

1 Motivation und Einführung

Information Retrieval

Gegenstand des Information Retrieval ist die **Suche nach Dokumenten**. Traditionell handelt es sich dabei um Textdokumente. In neuerer Zeit kommt aber verstärkt die Suche nach multimedialen Dokumenten (Bilder, Videos, Audios, Hypertext-Dokumente) oder z.B. die Suche nach Experten und Arbeitsgruppen mit einem bestimmten Kompetenzprofil hinzu. Dabei hat das Gebiet des Information Retrieval auch durch das Aufkommen des Internet bzw. des World Wide Web an Bedeutung und Aktualität gewonnen. Die enorme **Menge der verfügbaren Dokumente** macht eine effiziente (also letztlich schnelle) und zielgerichtete (also genaue) Suche nach nützlichen Dokumenten unabdingbar.

Die konkrete Ausprägung der Suche kann dabei variieren:

- *In welchem Datenbestand wird gesucht?*

  Hier unterscheidet sich eine Suche im Internet wesentlich von einer Suche in einem selbst verwalteten Datenbestand. Unterschiede liegen im Umfang des Datenbestandes und auch im Grad der Heterogenität. Während im Internet Dokumente verschiedenster Autoren in unterschiedlichster Qualität und in verschiedensten Formaten vorliegen, hat man in einem selbst verwalteten Intranet eine wesentlich bessere Kontrolle über Qualität und Formate.

- *Welche Typen von Dokumenten werden gesucht?*

  Häufig wird die Suche Textdokumente adressieren, aber auch die können in verschiedenen Formaten vorliegen (z.B. als HTML-Datei, als PDF-Datei, als einfache Text-Datei oder im Format eines bestimmten Textverarbeitungs-Programms). Daneben kann aber z.B. auch nach Bildern, Videos oder Audio-Dateien gesucht werden. In Abhängigkeit vom gesuchten Medientyp ergeben sich offensichtlich sehr unterschiedliche Anforderungen an die Formulierung des Informationswunsches, die technische Umsetzung der Suche und auch an die Darstellung des Ergebnisses.

- *Wie genau kann der Informationswunsch beschrieben werden?*

  In bestimmten Situationen weiß der Anfragende sicher sehr genau, was er sucht. In anderen Situationen wird der Informationswunsch eher vage sein. Dabei ist auch zu berücksichtigen, dass die gesuchten Dokumente selbst in typischen Anwendungen des Information Retrieval keine Datensätze mit eindeutigen Schlüsselattributen sind. Daher muss sich die Suche z.B. auf die in einem Dokument enthaltenen Begriffe beziehen. Dabei entstehen zahlreiche Probleme z.B. durch Synonyme (*Pferd* und *Gaul* meinen (fast) das Gleiche) oder durch Homonyme (*Bank* steht einmal für eine Sitzgelegenheit und einmal für ein Geldinstitut).

Die sich aus diesen Aspekten ergebenden Problemstellungen (und einige mehr) sind Gegenstand des Information Retrieval.

Beispiele für Information Retrieval Anwendungen

An dieser Stelle sollen zur ersten Einordnung fünf Beispiele für Systeme angeführt werden, die letztlich auf Methoden des Information Retrieval beruhen.

- Suche nach **Textinhalten** im World Wide Web. Diese Suchdienste erlauben sowohl die Suche nach einzelnen Begriffen als auch nach Phrasen und Sätzen in textuellen Dokumenten im World Wide Web. Beispiele dafür sind *Google*[1] (Abb. 1.1), *MSN Search*[2] oder *Yahoo Search*[3].

- Suche nach **Bildern** im World Wide Web oder in speziellen Angeboten. Zunächst bieten alle großen Internetsuchmaschinen, wie Google, MSN oder Yahoo auch eine Suche nach Bildern bzw. Fotos im World Wide Web an. Dabei werten diese Dienstanbieter fast ausschließlich den Text in der Umgebung eines Bildes sowie den Dateinamen aus. Spezielle Angebote wie *Flickr*[4] (Abb. 1.2) erlauben eine manuelle Beschreibung von Bildern mit Texten bzw. Begriffen, nach denen dann gesucht werden kann.

[1] *http://www.google.de/*, letzter Abruf: 19.10.06
[2] *http://www.live.com/*, letzter Abruf: 19.10.06
[3] *http://search.yahoo.com/*, letzter Abruf: 19.10.06.
[4] *http://www.flickr.com/*, letzter Abruf: 19.10.06

Abbildung 1.1 — *http://www.google.de/*, ein Beispiel für die Eingabemaske einer Suchmaschine im Internet

- Suche **innerhalb eines Webauftrittes** (Abb. 1.3). Webauftritte werden heute typischerweise mittels eines Content-Management-Systems verwaltet. Derartige Systeme legen die einzelnen Seiteninhalte in einer Datenbank ab und können auf dieser Basis Suchanfragen auf den Inhalten durchführen. Dabei entfällt das für allgemeine Suchmaschinen im World Wide Web notwendige Einlesen und Extrahieren der Textinhalte aus Webseiten.

- Suche in **weiteren Medientypen**. Sowohl die großen Suchmaschinenbetreiber wie auch zahlreiche kleine Anbieter unterstützen neben Text mittlerweile die Suche in Video- oder Radioausstrahlungen. Bisher werden dazu vorwiegend die gesprochenen Texte erfasst. Daneben gibt es Spezialsuchmaschinen, wie bspw. *Koders*[5] (Abb. 1.4), die in diesem Fall in Quelltexten von Open-Source-Projekten suchen.

- **Spezielle Dienste**. Während ein typischer Suchdienst auf eine manuelle Anfrage eines Anwenders hin eine Recherche nach ähnlichen Dokumenten im aktuellen Datenbestand beginnt, gibt es auch

[5] *http://www.koders.com/*, letzter Abruf: 19.10.06

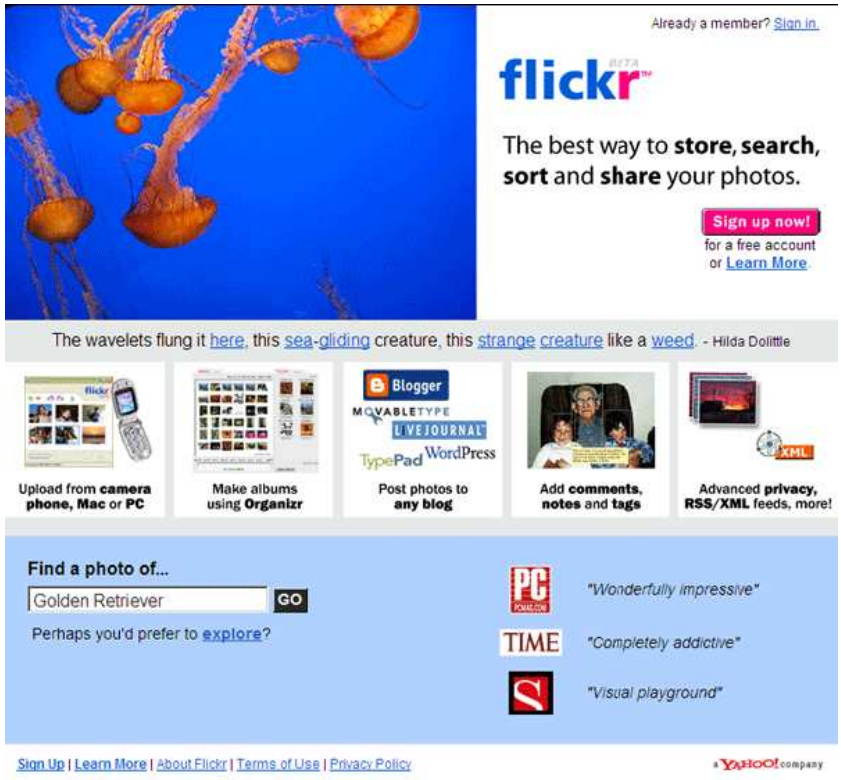


Abbildung 1.2 — Suche nach Bildern über zugeordnete textuelle Beschreibungen in *Flickr*

Dienste, wie bspw. *Google Web Alerts*[6] (Abb. 1.5), die einmal mit einem oder mehreren Suchbegriffen versehen, im World Wide Web von der Suchmaschine über die Zeit hinweg neu gefundene und gegenüber der Suchanfrage als relevant erachtete Dokumente z.B. per Email einem Anwender melden.

Einordnung und historische Wurzeln

Konkret geht es im klassischen Information Retrieval (IR) darum, aus einer **Kollektion von Dokumenten** zu einem gegebenen **Informa-**

[6] *http://www.google.com/alerts/*, letzter Abruf: 19.10.06

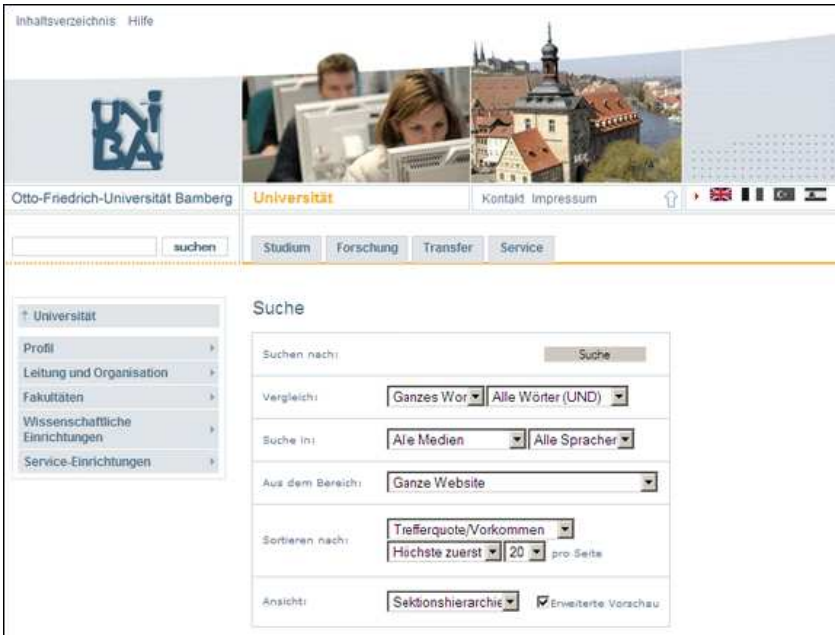


Abbildung 1.3 — Ein Beispiel für eine Eingabemaske zur Suche innerhalb der in einem Content-Management-System verwalteten Webseiten einer Organisation

tionswunsch die relevanten Dokumente zu ermitteln. Hierzu werden Techniken eingesetzt, die weit über eine einfache zeichenkettenbasierte Textsuche hinausgehen. So versucht man, von der konkreten Wortwahl in einem Dokument zu abstrahieren und statt dessen die Semantik des Dokumentes zu adressieren. Man spricht in diesem Zusammenhang auch von einer *inhaltsbasierten Suche*.

inhaltsbasierte Suche

Die Suche nach den zu einem Informationswunsch relevanten Dokumenten berücksichtigt dabei die **Vagheit und Unvollständigkeit**, die sowohl bei der Formulierung des Informationswunsches als auch bei der – ggf. automatischen – Interpretation des Inhalts der betrachteten Dokumente besteht.

Die Motivation für die Suche nach den zu einem konkreten Informationswunsch relevanten Dokumenten stammt **historisch** aus dem **Bibliothekskontext**. Hier werden von einem konkreten Benutzer die Bücher und Artikel der Bibliothek gesucht, die ihm bei einer konkreten Aufga-

Abbildung 1.4 — *Koders.com*: Suche in Quelltexten von Open-Source-Projekten

benstellung behilflich sein könnten. In der heutigen Zeit lässt sich dieses Szenario analog auf die Suche im Internet übertragen.

Neben dieser klassischen Aufgabenstellung des Information Retrieval treten dabei zunehmend auch Varianten auf. So kann man aus einem **Strom von Dokumenten**, wie z.B. aus einem abonnierten Nachrichtenticker oder aus einem Strom von eingehenden Emails die für einen bestimmten Benutzer relevanten ermitteln. Während bei der klassischen Aufgabenstellung des IR von einer relativ statischen Dokumentenkollektion ausgegangen wird, auf der verschiedenste Informationsanfragen bearbeitet werden müssen, ist nun der Dokumentenbestand stark dynamisch – es kommen immer neue Nachrichten oder Emails an – während die Anfragen, die in diesem Kontext typischerweise das Interessenspektrum des Benutzers charakterisieren, zumindest mittelfristig stabil sind.

Eine andere Aufgabenstellung ist die **Suche nach Experten** für ein bestimmtes Thema **oder nach Arbeitsgruppen**, die sich mit einem bestimmten Thema beschäftigen. Dabei wird nicht mehr nach Dokumenten gesucht, sondern nach Experten oder Arbeitsgruppen, die durch ein entsprechendes Profil charakterisiert sind. Schließlich ist auch die

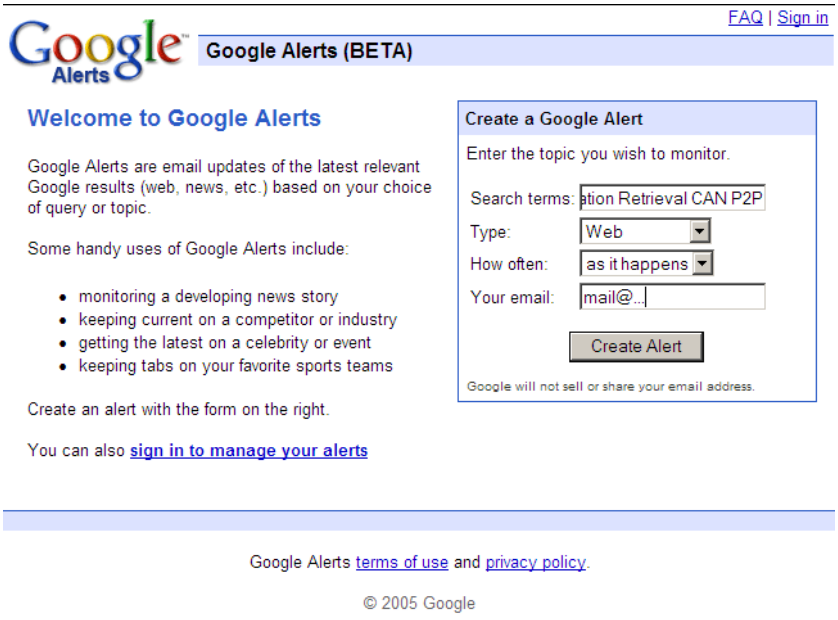


Abbildung 1.5 — Eingabemaske zur Einrichtung eines *Alerts bei Google*

bereits angesprochene **Suche nach multimedialen Dokumenten** ein interessantes Anwendungsgebiet für Techniken des Information Retrieval.

Dieser Kurs beleuchtet die unterschiedlichen Aspekte des Information Retrieval und der Implementierung von Information Retrieval Systemen (IR-Systemen). Wichtige Aspekte hierbei sind:

Information Retrieval System

- die Formulierung des Informationswunsches,
- die Interpretation des Inhalts textueller und auch multimedialer Dokumente,
- die hinter IR-Systemen stehenden Modelle,
- Fragen der Implementierung von IR-Systemen (wie die verwendeten Algorithmen und Datenstrukturen) sowie
- Fragen zur Beurteilung von IR-Systemen.

Der Bereich des Information Retrieval umfasst aber auch Klassifikationsverfahren und Ansätze zur Wissensrepräsentation, die in diesem Kurs ebenso behandelt werden wie Suchmaschinen im Internet oder Bilddatenbanken als spezielle IR-Systeme.

Das erste Kapitel soll zum einen die Zielsetzung des Information Retrieval aufzeigen. Zum anderen werden die Unterschiede zwischen Information Retrieval und klassischen Datenbanken erläutert. Neben einem ersten Überblick über verschiedene Verfahren des Information Retrieval werden zudem die Möglichkeiten zur Beurteilung von Suchergebnissen einleitend dargelegt.

1.1 Der Begriff Information Retrieval (IR)

Eine sehr nahe liegende Vorgehensweise bei der Annäherung an ein Gebiet ist, sich zunächst näher mit dem Begriff auseinander zu setzen. Wir wollen deshalb auch an dieser Stelle den Begriff des **Information Retrieval** genauer betrachten.

1.1.1 Eine erste Begriffserklärung

Information Retrieval

Der Begriff Information Retrieval (IR) setzt sich aus zwei Teilbegriffen zusammen. Ein Ansatz zur Erklärung des Begriffes Information Retrieval ist daher, die Begriffe *Information* und *Retrieval* einzeln zu deuten.

Information

Information

Der Begriff der Information wird in [SGR97] als »zweckgerichtetes Wissen zur Vorbereitung und Durchführung von Handlungen« definiert. Der Begriff der *Information* wird dabei im Dreiklang von *Daten*, *Wissen* und *Information* gesehen:

Daten

- Der Begriff der **Daten** ist auf der syntaktischen Ebene angesiedelt. In [SGR97] wird definiert: »Eine Darstellung maschinell verarbeitbarer Information nennt man in der Informatik Daten.« Hier geht es also um die Codierung auf einem Rechner. Formate für ganze Zahlen und Gleitkommazahlen sind hier ebenso ein Thema wie die syntaktische Darstellung strukturierter Daten z.B. mit XML. In

XML

diesem Sinne wäre also eine *Datenbasis* eine schlichte Sammlung von technisch verarbeitbaren Werten ohne jegliche Semantik.

Wissen

- **Wissen** bezeichnet darauf aufbauend mit semantischen Bedeutungen hinterlegte Daten. Allgemeiner wird Wissen auch als die Gesamtheit der Kenntnisse auf einem bestimmten Gebiet gesehen. Am Beispiel einer Bildersammlung würden die in ihren Farbwerten abgespeicherten Bilder die Daten darstellen. Das in diesen Bildern dargestellte repräsentiert jedoch das eigentliche Wissen.

Information

- Der Begriff der **Information** berücksichtigt zusätzlich den pragmatischen Aspekt. So definiert Kuhlen [Kuh90] Information aus Sicht des Information Retrieval folgendermaßen: *»Information ist die Teilmenge von Wissen, die von jemandem in einer konkreten Situation zur Lösung von Problemen benötigt wird.«* Häufig ist die Information zur Durchführung einer bestimmten Handlung nicht vorhanden. Eine Person wird dann auf entsprechende Quellen zugreifen, um aus dem dort gespeicherten Wissen die benötigte Information zu extrahieren. Mit der durch diesen Prozess neu erarbeiteten Information kann die Person ihre Handlung informationell absichern.

Information bezieht sich somit letztlich auf eine Teilmenge des Wissens, die in der konkreten Situation für eine bestimmte Person oder Personengruppe nützlich sein kann. Ohne klare Daten- und auch Wissensrepräsentation wäre aber die Extraktion von Information aus einem Wissensfundus gar nicht möglich. Wir werden uns deshalb in diesem Kurs auch mit Formen der Daten- und vor allem der Wissensrepräsentation beschäftigen.

Wissensrepräsentation

Das oben beschriebene Begriffsverständnis entspricht auch dem der deutschen **Informationswissenschaft**, die sich vor einigen Jahren auf eine einheitliche Terminologie geeinigt hat, die in Abbildung 1.6 graphisch dargestellt ist.

Der Schritt von Daten zum Wissen – also der Schritt von der syntaktischen zur semantischen Ebene – muss dabei im Allgemeinen mit der **Repräsentation** dieser Semantik einhergehen.

Repräsentation

Dies kann man plastisch am Beispiel von Bildern verdeutlichen. Die Bilder selbst sind zunächst in einem bestimmten technischen Format (z.B. JPEG, GIF oder TIFF) abgelegt. Damit sind Anwendungen in der Lage diese Bilder zu speichern, anzuzeigen und pixelweise zu bearbeiten. Die

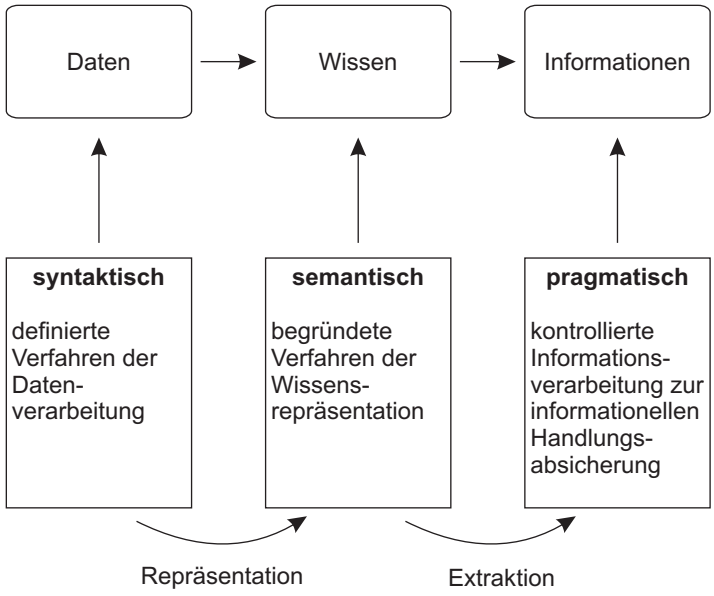


Abbildung 1.6 — Abgrenzung der Begriffe Daten, Wissen und Information

Semantik der Bilder wird von diesen Formaten aber nicht abgedeckt. Eine Anfrage nach Bildern, in denen z.B. Albert Einstein zu sehen ist, kann auf der Ebene dieser Bildformate nicht vorgenommen werden. Dazu muss der Inhalt der Bilder – das in ihnen vorhandene Wissen – explizit gemacht werden. Mit anderen Worten: dieses Wissen muss *repräsentiert* werden. Dies kann z.B. dadurch geschehen, dass wir jedes Bild in einer Datenbank mit Informationen über die darin abgebildeten Personen, Gegenstände, Tiere, ... versehen. Wir werden später sehen, dass dieser Aspekt der Wissensrepräsentation bei Textdokumenten zwar auch wünschenswert aber nicht ganz so zwingend wie z.B. bei Bildern ist. Der Grund hierfür ist, dass durch die einzelnen verwendeten Wörter die Semantik eines Textes wesentlich konkreter gegeben ist als die Semantik eines Bildes durch die einzelnen Farbpixel.

Extraktion

Der Schritt von der Semantik zur Information lässt sich dann als **Extraktion** charakterisieren. Aus einer großen Menge von gespeichertem Wissen wird die Information extrahiert, die einer Person in einer bestimmten Problemstellung hilfreich sein kann. Damit wird auch unmittelbar der Bezug zum *Retrieval* deutlich.

Retrieval

Zum Begriff *Retrieval* bietet z.B. Langenscheidts Wörterbuch [LAN87] folgende Bedeutungen an, die im Folgenden jeweils in den Kontext des Information Retrieval gesetzt werden: *Retrieval*

- *wiederfinden, wiederbekommen*
- *(sich etwas) zurückholen*
- *herausholen, herausfischen*
- *wiedergewinnen, wiedererlangen*
- *etwas der Vergangenheit entreißen*
- *apportieren*

Dieses Begriffsverständnis umfasst dabei letztlich drei Aspekte:

- Zunächst geht es darum etwas, was bereits existiert (wieder) zu finden. Dies ist exakt die Situation beim Übergang vom Wissen zur Information. Das Wissen ist vorhanden; es geht aber darum aus der Fülle des Wissens die in der konkreten Situation wichtigen Teile herauszufischen.
- Dies führt unmittelbar zum zweiten Aspekt des Begriffs, der darauf abzielt, dass aus einer großen Grundgesamtheit etwas *»herausgeholt«* oder *»herausgefischt«* werden muss. So hat man es im Information Retrieval meist mit großen Dokumentenbeständen wie Bibliotheken oder dem Internet zu tun, in denen Information gesucht wird.
- Schließlich deutet der Begriff *»apportieren«* auf IR-Systeme hin, die als »dienstbare Geister« die Informationsversorgung ihrer Benutzer gewährleisten sollen.

Damit kann in einer ersten Zusammenfassung festgehalten werden, dass der Begriff Information Retrieval die **(Wieder-)Gewinnung von Informationen, die von jemandem in einer konkreten Situation zur Lösung von Problemen benötigt werden**, meint. Ausgangsbasis ist im günstigsten Fall eine Wissensbasis, die bereits die Semantik der

Dokumente adressiert oder – im ungünstigeren Fall – eine Datenbasis, die Dokumente in einer wohldefinierten Syntax aber ohne Wissen über deren Semantik verwaltet.

Die **Aufgaben eines IR-Systems** sind in diesem Zusammenhang zum einen die *geeignete Repräsentation* von Daten in Wissen. Das heißt, Daten müssen in geeigneter Form so aufbereitet werden, dass das in diesen vorhandene Wissen zugreifbar wird. Zum anderen soll ein IR-System aus den um semantische Aspekte erweiterten Daten *Informationen extrahieren* können, die einem Anfragenden möglichst gut bei einer Problemlösung helfen können.

1.1.2 Weitere Begriffserklärungen

Um das Begriffsverständnis zum *Information Retrieval* auf eine breitere Basis zu stellen erscheint es sinnvoll einige weitere Begriffsbestimmungen hierzu aus der Literatur zu betrachten.

Die Fachgruppe Information Retrieval der Gesellschaft für Informatik e.V. (GI)[7] definiert den Begriff *Information Retrieval* wie folgt:

Prozess des Wissenstransfers

> »Im Information Retrieval (IR) werden Informationssysteme in Bezug auf ihre Rolle im *Prozess des Wissenstransfers* vom menschlichen Wissensproduzenten zum Informations-Nachfragenden betrachtet. Die Fachgruppe »Information Retrieval« in der Gesellschaft für Informatik beschäftigt sich dabei schwerpunktmäßig mit jenen Fragestellungen, die im Zusammenhang mit vagen Anfragen und unsicherem Wissen entstehen. *Vage Anfragen* sind dadurch gekennzeichnet, dass die Antwort a priori nicht eindeutig definiert ist. Hierzu zählen neben Fragen mit unscharfen Kriterien insbesondere auch solche, die nur im Dialog iterativ durch Reformulierung (in Abhängigkeit von den bisherigen Systemantworten) beantwortet werden können; häufig müssen zudem mehrere Datenbasen zur Beantwortung einer einzelnen Anfrage durchsucht werden. *Die Darstellungsform des in einem IR-System gespeicherten Wissens* ist im Prinzip nicht beschränkt (z.B. Texte, multimediale Dokumente, Fakten, Regeln, semantische Netze). Die *Unsicherheit* (oder die Unvollständigkeit)

[7] *http://www.uni-hildesheim.de/fgir/*, letzter Abruf: 20.10.06

> dieses Wissens resultiert meist aus der *begrenzten Repräsentation* von dessen Semantik (z.B. bei Texten oder multimedialen Dokumenten); darüber hinaus werden auch solche Anwendungen betrachtet, bei denen die *gespeicherten Daten selbst unsicher* oder unvollständig sind (wie z.B. bei vielen technisch-wissenschaftlichen Datensammlungen). Aus dieser Problematik ergibt sich die *Notwendigkeit zur Bewertung der Qualität* der Antworten eines Informationssystems, wobei in einem weiteren Sinne die *Effektivität des Systems* in bezug auf die Unterstützung des Benutzers bei der Lösung seines Anwendungsproblems beurteilt werden sollte.«

Diese Definition arbeitet neben der bereits im vorhergehenden Abschnitt besprochenen grundsätzlichen Einordnung des Information Retrieval zahlreiche Detailaspekte heraus:

Zunächst ist vom »Prozess des Wissenstransfers vom menschlichen Wissensproduzenten zum Informations-Nachfragenden« die Rede.

- Dieser Prozess beginnt damit, dass eine Person ein Dokument erstellt. Dabei kann es sich um ein Textdokument, um ein Bild, um einen Film oder etwas anderes handeln. Der Ersteller oder Autor des Dokumentes möchte dabei in dem Dokument einen Teil seines Wissens niederlegen. Offensichtlich wird das Dokument dieses Wissen aber nur unvollständig und ggf. auch missverständlich weitergeben.
- Schließlich wird das Dokument in einer Datenbank oder einem Dokumenten-Management-System in einer bestimmten Syntax abgelegt.
- Der Informations-Nachfragende wird sich nun an dieses System wenden um einige für ihn in seiner konkreten Situation hilfreiche Dokumente zu erhalten.

Jeder Schritt in diesem Prozess des Wissenstransfers bedingt dabei für sich genommen eine zusätzliche **Unsicherheit und Vagheit**. Es ist daher wichtig Information Retrieval nicht nur als den Prozess der Wissensextraktion aus einer Dokumentenkollektion sondern als Prozess des Wissenstransfers zu verstehen, wie er in Abbildung 1.7 vereinfachend zusammengefasst ist.

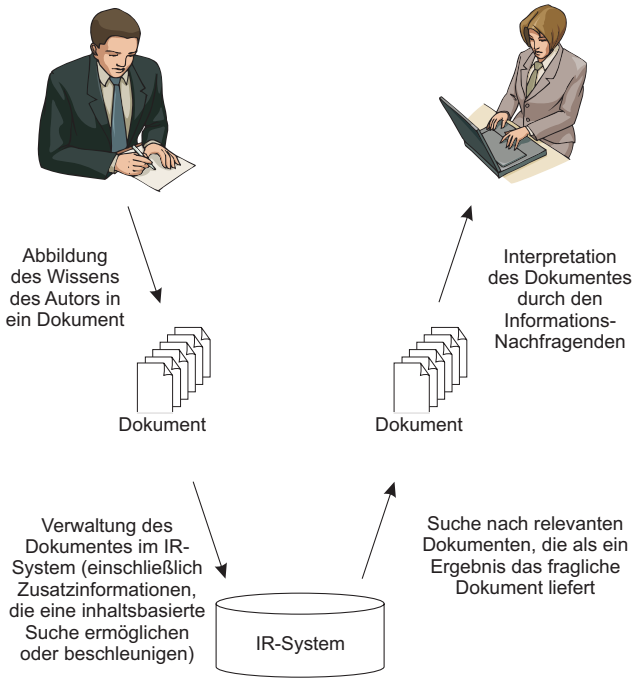


Abbildung 1.7 — Prozess des Wissenstransfers vom Autor eines Dokumentes bis zum Informations-Nachfragenden

Die Anfrage als vage Repräsentation des Informationsbedürfnisses

Repräsentation des Informationsbedürfnisses

Ein Teil der Vagheit in diesem Prozess entsteht dabei durch vage, ungenaue und unvollständige Anfragen. Die Anfrage, die an ein IR-System gestellt wird, ist immer nur ein Versuch das tatsächliche Informationsbedürfnis des Informations-Nachfragenden zu beschreiben. Die Zielsetzung eines IR-Systems darf nun nicht sein, diese gestellte Anfrage isoliert »richtig« zu beantworten. Vielmehr muss das System versuchen das hinter der Anfrage stehende Informationsbedürfnis zu befriedigen. Dazu kann es sinnvoll sein, neben der gestellten Anfrage noch weitere Quellen zur genaueren Klärung des Informationsbedürfnisses zu betrachten.

Relevance Feedback

Zum einen kann man dazu das so genannte **Relevance Feedback** einsetzen. Dabei gibt der Benutzer für die Ergebnisdokumente in einer ersten Antwort des Systems an, ob diese für ihn hilfreich sind oder nicht.

Auf dieser Basis kann dann ein verfeinertes Ergebnis ermittelt werden.

Ein anderer Ansatz mehr über das tatsächliche Informationsbedürfnis des Informations-Nachfragenden zu erfahren, kann die zusätzliche **Betrachtung des Kontextes** sein, in dem sich der Informations-Nachfragende befindet. Welche Anfragen hat er früher gestellt? Welche Dokumente betrachtet er gerade? Mit welchen Aufgaben ist er befasst? Welche Stelle/Rolle nimmt er ggf. im Unternehmen ein? Wissen über diesen Kontext kann dem IR-System helfen, die gestellte Anfrage genauer zu interpretieren.

Darstellung des Wissens in einem IR-System

Darstellung des Wissens

Neben der Anfrage liegt eine weitere Quelle der »Vagheit« in der Darstellungsform des in einem IR-System gespeicherten Wissens. Die *Unsicherheit* (oder die Unvollständigkeit) dieses Wissens resultiert dabei aus der *begrenzten Repräsentation* von dessen Semantik. Ziel des Information Retrieval müsste also eigentlich sein, dieses Wissen im IR-System möglichst vollständig zu verwalten.

Am **Beispiel von Bildern** kann man aber sehr schnell erkennen, dass Vollständigkeit hier nicht zu erreichen ist. Wie der Volksmund schon weiß, sagt ein Bild mehr als 1000 Worte und zur vollständigen Beschreibung des Inhalts eines Bildes würde man eine entsprechend umfassende Repräsentation brauchen.

Man denke zum Beispiel an ein Bild, das einen Popstar vor einer Menschenmenge zeigt. Hier ist es naheliegend, den Namen des Popstars mit dem Bild zu verknüpfen. Die Namen der Personen im Publikum wird man allerdings vermutlich nicht für so wichtig halten – und meist auch gar nicht kennen. Nun könnte aber jemand auf die Idee kommen, 40 Jahre später in einer Bilddatenbank nach Bildern zu suchen, die den aktuellen Bundeskanzler beim Besuch eines Popkonzertes zeigen.

Das Problem ist dabei offensichtlich, dass zum Zeitpunkt des Entstehens eines Dokumentes nicht klar ist, aus welchen Blickwinkeln Informations-Nachfragende später auf dieses Dokument schauen könnten. Trotzdem muss beim Design eines IR-Systems die Frage nach der »Darstellungsform des in einem IR-System gespeicherten Wissens« eine zentrale Rolle spielen.

Beurteilung der Qualität von IR-Systemen

Qualität von IR-Systemen

Während sich klassische Datenbanken bei der Beantwortung von Standardanfragen – also z.B. bei der Suche nach den Bestellinformationen eines bestimmten Kunden – nicht in der Qualität des Ergebnisses unterscheiden (hier sollten verschiedene Datenbanksysteme bei der gleichen gegebenen Datenbasis immer das gleiche Ergebnis liefern) verhält sich dies bei IR-Systemen anders. Ein gutes IR-System unterscheidet sich von einem schlechten System eben dadurch, dass es

- einerseits **mehr relevante Dokumente** zum Informationsbedürfnis liefert und
- andererseits den Anfragenden mit **weniger irrelevanten Dokumenten** im Ergebnis belastet.

Harter [Har96] formuliert dies wie folgt: *»For a given information problem, the purpose of the system is to capture wanted items and to filter out unwanted items.«*

Damit stellt sich unmittelbar die Frage, wie die Qualität des Ergebnisses eines IR-Systems gemessen werden kann. Auch diese Fragestellung zählt zu den zentralen Forschungsthemen im Information Retrieval.

Nachdem wir nun zahlreiche wichtige Teilaspekte des Begriffs Information Retrieval angerissen haben, wollen wir im Weiteren auf die typischen Einsatzszenarien von IR-Systemen eingehen. Zur Abrundung unserer Begriffsbestimmung soll hier aber noch die recht griffige Definition von Frakes und Baeza-Yates [FBY92] angegeben werden:

> »An IR system matches user queries – formal statements of information need – to documents stored in a database. A document is a data object, usually textual, though it may also contain other types of data such as photographs, graphs, and so on.«

1.1.3 Beispielszenarium für den Einsatz eines IR-Systems

Eine typische Ausgangssituation für den Einsatz eines Information Retrieval Systems kann folgendermaßen aussehen:

- Jemand steckt in einer konkreten Situation bzw. muss ein bestimmtes Problem lösen.
- Im Hinblick auf diese Situation bzw. auf dieses Problem sucht er für ihn interessante Dokumente bzw. Objekte.
- Bei dieser Suche soll ihn das Information Retrieval System unterstützen.

Um in diesem Szenario hilfreich zu sein, muss das IR-System den semantischen Inhalt der Dokumente – bzw. allgemeiner der Objekte – adressieren. Man spricht deshalb auch von der *inhaltsbasierten Suche nach Dokumenten bzw. Objekten.*

Das Ziel muss somit eine Suche sein, die nicht auf den Vergleich von Zeichenketten (das so genannte *Pattern Matching*) beschränkt ist, sondern den semantischen Inhalt der Dokumente adressiert. Die Art der Dokumente kann dabei einfache Textdokumente und strukturierte Textdokumente, aber auch Bilder oder Audio- und Videodokumente umfassen.

Pattern Matching

Bevor wir auf die Möglichkeiten zur Realisierung einer solchen inhaltsbasierten Suche eingehen, wollen wir uns im Folgenden zunächst noch mit der zeitlichen Entwicklung des Information Retrieval und vor allem mit der Abgrenzung zwischen Information Retrieval und Datenbanken beschäftigen.

1.1.4 Die historische Sicht und neuere Entwicklungen

Information Retrieval ist eine alte Disziplin, die vor allem im Zusammenhang mit dem Bibliothekswesen entstanden ist. Die Zielsetzung Bücher in Bibliotheken effizient auffindbar zu machen, hat dabei zwei Teilaspekte:

- Zum einen geht es um die **Anordnung der Bücher in den Regalen**. Thematisch ähnliche Bücher sollten nah beieinander stehen, damit ein Bibliotheksnutzer die für ihn relevanten Bücher nicht völlig verstreut in der ganzen Bibliothek zusammensuchen muss. Dies klingt auf den ersten Blick recht harmlos. Man muss aber bedenken, dass viele Bücher mehrere Themen in Kombination behandeln und ihre eindeutige Zuordnung zu genau einem Thema damit schwer fällt. Die Anordnung der Bücher in einer Bibliothek kann aber in den Regalen nur ein- bis maximal zweidimensional erfolgen.

- Neben einer sinnvollen physikalischen Positionierung der Bücher trägt das Vorhandensein entsprechender Kataloge wesentlich zur Nutzbarkeit einer Bibliothek bei. Dabei sind **Autoren, Titel- und auch Schlagwortkataloge** üblich. Diese Kataloge helfen dem Informations-Nachfragenden sich, bevor er physisch zu den Büchern geht, über mögliche Standorte relevanter Bücher zu informieren.

Klassifikationsschemata

Im Hinblick auf eine bibliotheksübergreifend einheitliche Systematik für das Aufstellen der Bücher wurden **Klassifikationsschemata** entwickelt, die im allgemeinen hierarchisch organisiert sind. Jedes Buch wird dann in dieses ggf. mehrstufige Schema eindeutig eingeordnet und entsprechend dieser Einordnung mit anderen gleich eingeordneten Büchern physisch gemeinsam abgestellt. Wir werden entsprechende Klassifikationsschemata im weiteren Verlauf der Vorlesung noch ausführlich betrachten. Aus der heutigen Perspektive ist dabei interessant zu beobachten, dass es sich bei den so genannten Katalogen im Internet (Yahoo, *Web.de*[8], ...) im Grunde auch um solche Klassifikationsschemata handelt.

Verschlagwortung

Thesaurus

Bei den Katalogen ist die Erstellung eines Titel- und eines Autorenkataloges noch relativ einfach. Die Erstellung eines Schlagwortkataloges setzt aber eine sinnvolle Zuordnung von Schlagwörtern zu Büchern voraus. Dies impliziert zwei Probleme. Zum einen entsteht hier ggf. ein recht hoher manueller Aufwand, wenn diese **Verschlagwortung manuell** durch Experten erfolgen soll (was andererseits eine hohe Qualität der Verschlagwortung gewährleistet). Zum anderen muss aber eine einheitliche Verwendung der Begrifflichkeiten gewährleistet werden. Zu diesem Zweck wurden **Thesauri** entwickelt, die die Begriffsverwendung bei der Verschlagwortung möglichst eindeutig regeln sollen. Liegen diese Thesauri auch dem Anfragenden bei der Suche im Katalog vor, so kann ein Schlagwortkatalog ein sehr effizientes Hilfsmittel bei der Suche sein.

Thesauri finden sich in allen modernen IR-Systemen in der einen oder anderen Form wieder, weil ein eindeutiges Begriffsverständnis für eine erfolgreiche Suche von entscheidender Bedeutung ist, und auch Schlagwortkataloge und Klassifikationsschemata haben kaum etwas von ihrer Aktualität eingebüßt. Beide sind auch im Zusammenhang moderner IR-Systeme sinnvoll. Trotzdem erscheint es zunächst hilfreich einen kurzen Blick auf die neuen Herausforderungen an IR-Systeme zu werfen, die sich insbesondere mit den Schlagworten *Internet* oder *WWW* einerseits und *Multimedia* andererseits verbinden.

[8] *http://www.web.de/*, letzter Abruf: 19.10.06

Neuere Entwicklungen

Durch das Internet ist die oben beschriebene klassische Aufgabenstellung des Information Retrieval allgegenwärtig: Aus einer unübersehbaren Flut von Dokumenten muss in einer konkreten Situation die möglicherweise nützliche Information herausgefischt werden. Dabei ergeben sich im Hinblick auf das **World Wide Web** (WWW) folgende neue Anforderungen für das Information Retrieval:

- In fast allen Fällen sind zu den angebotenen Informationen im World Wide Web **keine bibliographischen Informationen vorhanden**: der Autor ist häufig unbekannt, einen Verlag gibt es nicht und statt eines Erscheinungsjahres hat man allenfalls ein *creation date*. Obwohl entsprechende Meta-Tags die Angabe bibliographischer Informationen durchaus erlauben würden, sind diese oft leer oder von zweifelhaftem Inhalt – eine sorgfältige Erfassung durch einen geschulten Bibliothekar findet dabei natürlich nicht statt. Dadurch kann eine im Bibliotheksbereich sehr gut arbeitende Suche auf den bibliographischen Informationen hier nicht greifen. Eine Indexierung im WWW muss ihren Schwerpunkt auf die Inhalte der Dokumente legen.

- Die im World Wide Web abrufbare Menge an Dokumenten hat Ausmaße angenommen, die eine vollständige Erfassung unmöglich machen. Hinzu kommt das schnelle Wachstum. Nach Schätzungen umfasst das Web zur Zeit einige Milliarden Dokumente und jeden Tag kommen ca. 1.000.000 neue Dokumente hinzu. Damit ist insbesondere jeder Indexierungsansatz, der auf manuelle Arbeiten setzt, aussichtslos. Lediglich **vollautomatische Ansätze** können hier Erfolg haben.

- Schließlich gewinnen **strukturierte Dokumente**, deren Struktur auch technisch zugreifbar ist, an Bedeutung. Die Standardisierung von *XML*[9] ist hier als eine wesentliche Triebkraft zu nennen, aber auch HTML enthält ja in begrenztem Umfang Strukturierungsmöglichkeiten (siehe hierzu z.B. *SELFHTML*[10]). Dies erlaubt einerseits bei der Suche nicht mehr nur vollständige Dokumente zu adressieren – es können nun auch gezielt einzelne Abschnitte gefunden werden. Andererseits kann die Struktur auch zu präziseren

Strukturierte Dokumente

[9] *http://www.w3.org/XML/*, letzter Abruf: 19.10.06
[10] *http://de.selfhtml.org/*, letzter Abruf: 19.10.06

Anfragen genutzt werden. So könnte jemand nach Büchern suchen, die nach einem Abschnitt, der von Berlin und dem Mauerbau handelt, im Folgeabschnitt auf die Teilung Koreas eingehen. Hieraus ergeben sich neuartige Anforderungen an IR-Systeme.

Eine andere Entwicklung stellen neue Medien in Form von **digitalen Audio-, Bild- und Videoformaten** dar. Auch auf diesem Gebiet ist die Suche in den Inhalten der einzelnen Bild-, Ton- und Videodokumenten wünschenswert, wenn auch deutlich schwieriger verglichen mit der Suche in Textdokumenten. Grundsätzlich kann man hier in zwei Richtungen vorgehen.

- Man kann eine **manuelle Verschlagwortung** der Bilder, Audios oder Videos vornehmen und diese mit Metadaten über ihre Inhalte versehen. Die Suche auf diesen Metadaten ist dann wieder eine reine Textsuche. Allerdings ist der manuelle Aufwand für die Verschlagwortung sehr groß.

- Alternativ werden daher – um es am Beispiel der Bilder zu verdeutlichen – Verfahren der **Bildanalyse** angewendet, um die Objekte im Bild und ihre charakteristischen visuellen Eigenschaften zu ermitteln. Dies ist zumindest für Ähnlichkeitsanfragen recht erfolgversprechend, bei denen zu einem gegebenen Musterbild nach ähnlichen Bildern gesucht wird.

Nimmt man den Multimedia-Aspekt und den Aspekt der strukturierten Dokumente zusammen, so ergeben sich weitere Möglichkeiten. Erfolgversprechend ist in diesem Kontext die Kombination der Suchmöglichkeiten auf mehreren Medientypen. So lässt sich bei Bildern, die sich innerhalb eines Textes befinden, z.B. durch die Bildunterschrift auch auf den möglichen Inhalt des Bildes schließen.

Multimedia

Wir werden auf die Aspekte des **Multimedia-IR** gegen Ende des Kurses noch eingehen. Zunächst werden wir uns aber ausführlich mit Text-IR beschäftigen, weil – wie das Beispiel der Bildunterschrift verdeutlicht – Techniken des Text-IR auch im Hinblick auf andere Medientypen immer wieder eine leistungsfähige Basis bilden.

1.2 Aufgabenstellungen für IR-Systeme

Welche Aufgabenstellungen können nun typischerweise von IR-Systemen unterstützt werden? Anders könnte man auch fragen: Welche Möglichkeiten gibt es, um den Problemstellungen des Information Retrieval nachzugehen? Die typischen Ansätze lassen sich in vier Klassen kategorisieren:

- **Klassifikation** *Klassifikation*

  Die Wurzeln dieses Ansatzes liegen, wie bereits erwähnt, im Bibliothekswesen. Dabei werden die Dokumente im Allgemeinen manuell in ein Klassifikationsschema eingeordnet. Klassifikationsschemata sind meist in Form einer Baumstruktur aufgebaut, die von einer grob strukturierten Ebene ausgehend mehrstufig Untergliederungen definiert. Die Zuordnung der Dokumente zu den Knoten dieses Baumes ist dabei häufig eindeutig, d.h. ein Dokument wird genau einem Knoten zugeordnet. Wenn man mit der Klassifikation das Aufstellen von Büchern organisieren will, ist eine solche eindeutige Zuordnung offensichtlich zwingend. Im Falle von elektronischen Katalogen (wie Yahoo) kann allerdings durchaus eine Mehrfachzuordnung vorgenommen werden. Hier erscheint auch eine zumindest teilautomatische Zuordnung von Dokumenten sinnvoll.

- **Anfragebearbeitung** *Anfragebearbeitung*

  Während bei der Klassifikation die Vorbereitung des Dokumentenbestandes für die effiziente Durchsicht durch Benutzer im Vordergrund steht, geht es hier darum, einen konkreten Informationswunsch zu betrachten und zu bearbeiten. Der Informations-Nachfragende stellt eine *Anfrage* an das System, d.h. der Benutzer formuliert seinen Informationswunsch als Anfrage an ein System. Das System versucht nun zu dieser Anfrage ohne weitere manuelle Eingriffe relevante Dokumente zu finden. Zur Klasse der Systeme, die sehr klar dieser Herangehensweise folgen, gehören bspw. Suchmaschinen, die den Text umfangreicher Dokumentenkollektionen oder von Internetseiten durchsuchen.

  Nicht ganz von der Hand zu weisen ist dabei aber, dass Klassifikation und Anfragebearbeitung letztlich zwei Seiten der gleichen Medaille sind, denn eine effiziente Anfragebearbeitung ist ohne entsprechende Aufbereitung/Strukturierung der Dokumentenkollektion gar nicht denkbar.

Browsing

- **Browsing**

  Eine Alternative zur Anfragebearbeitung ist das Browsing. Ausgangspunkt ist hier eine Kollektion von Hypertextdokumenten, die untereinander verlinkt sind. Dies bedeutet, man kann sich das in einer Dokumentensammlung enthaltene Wissen interaktiv (primär durch Navigation) erarbeiten. Man startet also an einem thematisch eher allgemein ausgerichteten Knoten und bewegt sich von dort Schritt für Schritt weiter zu spezielleren Inhalten. Man muss sich das Browsing dabei keineswegs als Navigieren im Internet vorstellen. Vielmehr werden die Dokumente mit so genannten Clusterverfahren zu inhaltsähnlichen Clustern gruppiert, deren Dokumente untereinander nicht verlinkt sein müssen, aber inhaltsgleiche Informationen enthalten. Dabei projizieren diese Verfahren Dokumente in einen meist mehrdimensionalen Raum und bilden aus in diesem Raum benachbarten Dokumenten einzelne Cluster, deren Dokumente inhaltlich ähnlich sind. Der Benutzer kann dann in diesem Raum navigieren und er findet dort bei einem für ihn interessanten Dokument in der Nachbarschaft mit hoher Wahrscheinlichkeit weitere interessante Dokumente. Wir werden auch solche Verfahren gegen Ende des Kurses ansprechen.

Clusterverfahren

Information Filtering

- **Information Filtering**

  Die vierte Möglichkeit versucht mittels des Information Filtering aus einem andauernden Strom von Dokumenten, wie bspw. Nachrichten, automatisch die für einen Benutzer interessanten herauszufiltern. Dabei wird als Grundlage meist ein sog. Benutzerprofil eingesetzt, das die Prioritäten und Informationswünsche des Benutzers speichert. Durch den kontinuierlichen Vergleich von ankommenden Dokumenten mit dem Profil werden so die relevanten Dokumente ermittelt.

  Information Filtering und das Stellen einer Anfrage haben somit zahlreiche Gemeinsamkeiten, aber auch Unterschiede. So werden bei beiden die für einen Benutzer relevanten Dokumente auf der Basis eines von ihm geäußerten Informationswunsches ermittelt. Diese Äußerung findet beim Stellen einer Suchanfrage explizit statt, während Profile diesen Informationswunsch meist implizit aus einem Fragenkatalog zu Interessensgebieten oder einer Historie der bisher betrachteten Dokumente ermitteln.

Informationswunsch

  Unterschiede zwischen Information Filtering und der Bearbeitung einer Anfrage in einer Suchmaschine finden sich auch in der Reali-

sierung der Systeme wieder. Während beim Information Filtering laufend neue Dokumente betrachtet werden müssen, arbeitet eine Suchmaschine auf vielen meist statischen Dokumenten, die zuvor indexiert wurden. D.h. eine Suchmaschine kann sich, vergleichbar mit einem Datenbanksystem, intelligente Such- und Zugriffspfade aufbauen und damit die Anzahl der zu betrachtenden Dokumente bei einer Anfrage stark einschränken.

Wir werden auf alle vier Varianten im Laufe des Kurses eingehen. Den Schwerpunkt bilden aber – nicht zuletzt aufgrund ihrer praktischen Bedeutung – Verfahren der Anfragebearbeitung. Die Bearbeitung einer Anfrage an ein IR-System weist dabei natürlich auch gewisse Parallelitäten zu Datenbanksystemen auf. Es gibt aber, wie wir im folgenden Abschnitt sehen werden, auch zahlreiche Unterschiede.

1.3 Abgrenzung von Information Retrieval und Fakten Retrieval

Zum genauen Verständnis des Information Retrieval ist ein klares Bild über die Beziehungen zu Datenbanken wichtig. Während beide Bereiche nämlich lange Zeit relativ unabhängig voneinander betrachtet wurden, wird zunehmend deutlicher, dass bei allen Unterschieden zwischen IR-Systemen und Datenbankmanagement-Systemen (DBMS) viele Problemstellungen existieren, die sich mit einem »Mittelding« sehr gut abdecken ließen.

Datenbank

DBMS

Bevor wir auf diesen Aspekt zurückkommen erscheint es aber wichtig, die Unterschiede von **Information Retrieval** zu **Fakten/Daten Retrieval** zu verstehen. Unter Fakten Retrieval verstehen wir dabei bspw. den lesenden Zugriff auf klassische relationale Datenbanken.

Die Tabelle 1.1, die sich an [Rij79] orientiert, stellt stichwortartig die Unterschiede zwischen Fakten und Information Retrieval heraus.

Die einzelnen Punkte werden wir im Folgenden im Detail betrachten.

Matching

Beim Fakten Retrieval liegt ein **Exact Match** vor, d.h. die Zugehörigkeit

Exact Match

	Fakten Retrieval	Information Retrieval
Matching	Exact Match	Partial Match, Best Match
Inference	Deduction	Induction
Model	Deterministic	Probabilistic
Classification	Monothetic	Polythetic
Query Language	Artificial	Natural
Query specification	Complete	Incomplete
Items wanted	Matching	Relevant
Error response	Sensitive	Insensitive

Tabelle 1.1 — Abgrenzung von Information Retrieval und Fakten/Daten Retrieval

zum Ergebnis ist für jeden Datensatz eindeutig entscheidbar. Lautet die Anfrage z.B. **SELECT** Vorname, Nachname **FROM** Kunde **WHERE** Saldo < -1000, dann ist für jeden Kundendatensatz eindeutig zu bestimmen, ob er zum Ergebnis der Anfrage gehört (von der Problematik der Nullwerte wollen wir hier einmal absehen).

Partial Match
Best Match

Im Information Retrieval setzt man dagegen stärker auf ein **Partial Match** oder ein **Best Match**. Würde z.B. nach Dokumenten gesucht, die sich mit *Katzen, Futter, Kosten, Transport, Versand* beschäftigen, dann würde man wohl sicherlich Dokumente bevorzugen, die alle diese Begriffe enthalten. Aber was ist, wenn es keine solchen Dokumente gibt? Dann wäre der Informations-Nachfragende vielleicht auch an den Dokumenten interessiert, die nur einen Teil der Anfrage erfüllen (Partial Match). Dies kann nun z.B. in einem sehr einfachen Ansatz so umgesetzt werden, dass der Informations-Nachfragende ein Ranking der Dokumente erhält, in dem zunächst die Dokumente gelistet werden, die alle fünf gewünschten Begriffe enthalten, anschließend die mit vier Begriffen, die mit drei Begriffen, ... (Best Match).

Als wesentlicher Unterschied zu Datenbanken ist das Ergebnis hier keine Menge von Dokumenten, sondern eine Rangordnung, die versucht, die Dokumente, die mit recht hoher Wahrscheinlichkeit interessant für den Anfrager sein werden, oben zu listen.

Mittlerweile finden sich auch einfache Funktionalitäten in klassischen Datenbanksystemen, die ein Partial Match zulassen, bspw. die Sortierung nach einer bestimmten Reihenfolge (in SQL bspw. durch: **order by**) bei gleichzeitiger Größenbeschränkung der Ergebnismenge.

Ein Beispiel für den Bedarf nach Partial Match in Datenbanken: Auswahl in einer Datenbank

Das folgende Beispiel soll verdeutlichen, dass auch in relationalen Datenbanksystemen Anwendungen üblich sind, in denen eigentlich eine Best Match Suche wünschenswert ist.

Best Match Suche

Als Beispiel betrachten wir die Suche nach einem für unsere Bedürfnisse passenden Wagen in einer Datenbank mit Gebrauchtwagen.

Die Tabelle 1.2 stellt einen Ausschnitt aus der Datenbank mit Gebrauchtwagen dar. In der Tabelle *Gebrauchtwagen* repräsentiert jeder einzelne Datensatz einen Wagen mit seiner Typenbezeichnung, seinen technischen Daten, der Laufleistung sowie dem Preis. Jeder in dieser Tabelle erfasste Wagen wird durch den Primärschlüssel G_NR eindeutig identifiziert.

G_NR	MODELL	KLIMA	BAUJAHR	KW	KILOMETER	PREIS
132	601	ja	1994	45	19000	27000
472	601	ja	1993	80	10000	17000
173	601	ja	1995	45	23000	19000
...	...	...	...	...	...	...

Tabelle 1.2 — Ausschnitt aus der Datenbank mit Gebrauchtwagen

Gesucht wird nun ein gebrauchter Pkw Modell 601 mit folgenden Eigenschaften:

- Laufleistung weniger als 20.000 km
- Baujahr nach 1993
- Motorleistung mindestens 45 KW und
- möglichst günstiger Preis

SQL-Anfrage

Möchten wir in der Datenbank zu unseren gewünschten Eigenschaften passende Pkws finden, so können wir eine entsprechende SQL-Anfrage an die Datenbank stellen. Die folgende Anfrage erscheint dabei ebenso typisch wie nahe liegend:

```
SELECT G_NR
    FROM Gebrauchtwagen
    WHERE KILOMETER < 20000
        AND BAUJAHR > 1993
        AND KW >= 45
    ORDER BY PREIS ASC
```

Die »möglichst günstig« Anforderung wird durch das **ORDER BY** PREIS **ASC** realisiert, das alle gefundenen Datensätze nach dem Preis aufsteigend sortiert.

Als Ergebnis erhalten wir nun den Datensatz aus Tabelle 1.3:

G_NR	MODELL	KLIMA	BAUJAHR	KW	KILOMETER	PREIS
132	601	ja	1994	45	19000	27000

Tabelle 1.3 — Ergebnis der Anfrage

An dieser Stelle kommt die Frage auf, ob bei unseren Kriterien zur Auswahl eines Pkws nicht eine Substituierbarkeit einzelner Kriterien vorliegt. D.h. wäre man vielleicht nicht gewillt gewesen, den Wagen mit der Nr. 173 zu nehmen, der zwar 3.000 Kilometer mehr Laufleistung hat, aber dafür Baujahr 1995 ist? Vielleicht wäre auch Pkw Nr. 472 aufgrund seines starken Motors bei gleichzeitig geringer Laufleistung interessant.

Wie wir also sehen, haben wir durch die Abbildung unserer eher vagen Kriterien in eine künstliche, formal definierte Sprache die Kriterien derart eingeschränkt, dass uns durchaus interessante Suchergebnisse verborgen geblieben sind.

Inferenz

Inferenz

Unter Inferenz versteht man die Fähigkeit aus vorhandenem Wissen Schlussfolgerungen zu ziehen, die wiederum zu neuem Wissen führen. Dabei gibt es mehrere mögliche Arten der Schlussfolgerung, zu denen *Deduktion* und *Induktion* zählen.

Fakten Retrieval benutzt die **Deduktion**, d.h. die Herleitung des Besonderen aus dem Allgemeinen. In einer Anfrage werden die gesuchten Datensätze (das Besondere) aus der Datenbasis (dem Allgemeinen) extrahiert. *Deduktion*

Information Retrieval setzt die **Induktion** ein, d.h. die Herleitung von allgemeinen Regeln aus Einzelfällen. Gemeint ist hiermit insbesondere, dass aus einer konkreten Anfrage auf die Eigenschaften der Dokumente, die zum dahinter liegenden Informationswunsch des Anfragenden passen könnten, geschlossen wird. Noch deutlicher wird dies wenn – wie im IR durchaus üblich – der Informationswunsch durch ein Beispieldokument repräsentiert wird. Der Anfrager sucht dann eben ähnliche Dokumente. Hier wird vom speziellen Einzelfall eines relevanten Dokumentes auf die Eigenschaften aller relevanten Dokumente geschlossen. *Induktion*

Daneben enthalten viele IR-Systeme natürlich auch deduktive Aspekte, wenn sie mittels einer Suchanfrage bspw. aus einer Menge von Dokumenten die relevanten heraussuchen wie Datenbanken dies ja auch tun. Im Vordergrund steht bei der Charakterisierung des IR aber das induktive Moment, welches sich eben darauf bezieht, dass aus einer Anfrage / einem Beispieldokument auf die Charakteristika aller relevanten Dokumente geschlossen wird.

Modell

Das Modell beim Fakten Retrieval ist **deterministisch**. Datenbanken kennen bspw. keine Vagheit, von einer Anfrage wird deterministisch auf das Ergebnis geschlossen. *Modell deterministisch*

Information Retrieval benutzt (in der Regel) ein **probabilistisches** Modell, da es zahlreiche Quellen der Unschärfe gibt. Die erste Quelle der Unschärfe entsteht durch eine Formulierung der Anfrage bspw. in einer natürlichen Sprache. Eine weitere Quelle ergibt sich aus der Unschärfe der verwalteten Dokumente, die Spielräume für Interpretation lassen – bezeichnet bspw. das Wort *Bank* eine *Sitzgelegenheit* oder ein *Geldinstitut*? Eine dritte Quelle stellt das induktive Schließen dar, das vom Speziellen aus verallgemeinert, was zwangsweise Unschärfe mit sich bringt. *probabilistisch*

Klassifikation

Fakten Retrieval versteht sich als **monothetisch**. Wenn man die Ein- *monothetisch*

teilung der Objekte in Klassen betrachtet, dann sind die Attributwerte der Objekte, die immer alle vorhanden sind, notwendig und hinreichend zur Einteilung. D.h. es gibt neben dem einen Kriterium mit den beiden Einteilungen relevant oder nicht relevant kein zweites Kriterium.

polythetisch

Im Gegensatz dazu begreift sich Information Retrieval als **polythetisch**. Im Information Retrieval existiert im Allgemeinen kein festes Schema, d.h. zu den Dokumenten sind u.U. unterschiedliche Informationen vorhanden. Wir gehen davon aus, dass der Aspekt, den wir durch den Informationswunsch kennen, nur ein Kriterium ist, das letztlich die Relevanz der Dokumente beeinflusst. Daher ist auch keine eindeutige Zuordnung im Ergebnis möglich.

Query Language

Anfragesprache

Bei Datenbanksystemen kommen künstliche **Anfragesprachen** zum Einsatz. Relationale Datenbanksysteme setzen die Structured Query Language (SQL) zur Formulierung von Anfragen ein. Eine derartige Anfragesprache ist syntaktisch und semantisch formal definiert. Ein Datenbanksystem liefert nun zu einer gegebenen Anfrage ein deterministisches Ergebnis.

Boolescher Ausdruck

Ein Information Retrieval System könnte natürlich auch eine künstliche Anfragesprache aufbauen. In der Tat verwenden bspw. fast alle Internetsuchmaschinen Boolesche Ausdrücke zur Formulierung einer Suchanfrage. Die Eingabe eines natürlichsprachlichen Satzes als Suchterm führt dann meist zu einer Suchanfrage die je nach verwendeter Suchmaschine Dokumente findet, die bei einer Und-Verknüpfung der einzelnen Wörter des Suchterms alle Wörter oder bei einer Oder-Verknüpfung möglichst viele Wörter der Suchanfrage enthalten.

Anfrageformulierung
Informationswunsch

Ziel des Information Retrieval ist die Unterstützung der Anfrageformulierung in natürlicher Sprache, bei der der Benutzer seinen Informationswunsch möglichst natürlich formulieren kann. Daneben steht aber immer auch das Verständnis, dass die Anfrage nur eine (unvollständige) Repräsentation des Informatioswunsches ist.

Query specification

Ein Fakten Retrieval System liefert für eine Anfrage immer ein eindeutiges Ergebnis, das genau dieser Anfrage entspricht. D.h. die Anfrage ist

für ein Ergebnis stets vollständig, da zum Liefern des Ergebnisses keine weiteren Informationen benötigt werden.

Da natürliche Sprache zwangsweise unscharf und unvollständig ist, ist auch die Anfrage beim Information Retrieval damit unscharf und unvollständig. Es ist daher nicht direkt überprüfbar, ob das Ergebnis zu einer Suchanfrage »richtig« oder »falsch« ist.

Items wanted

Die als Ergebnis einer Anfrage resultierenden und auch gewünschten Elemente sind im Falle eines Fakten Retrieval Systems, bspw. einer Datenbank, diejenigen die der Anfrage exakt entsprechen. Im Falle eines Information Retrieval Systems können wir nicht von exakt passend oder unpassend sprechen, sondern müssen vielmehr den Begriff der Relevanz einführen. Gefundene Dokumente sind für einen Benutzer mehr oder weniger relevant.

Fehlerverhalten

Im Bereich des Fakten Retrievals würden Anwendungen sehr sensibel reagieren, wenn das System z.B. bei der Rechengenauigkeit Zugeständnisse macht, um Laufzeit zu sparen, oder wenn im Ergebnis bestimmte Datensätze fehlen. Die Anwendungen sind daher im Hinblick auf ein mögliches Fehlverhalten des Systems als sehr **sensitiv** einzustufen. Wenn eine Datenbank zu einer gegebenen Anfrage falsche Ergebnisse liefert, ist dies schlichtweg nicht tolerierbar.

Auf dem Gebiet des Information Retrieval muss dagegen zunächst der Begriff des Fehlers definiert werden. Wenn das System ohnehin mit Ungenauigkeiten in der Anfrage und in den Dokumenten konfrontiert ist, lohnt es sich dann, mit 10 Nachkommastellen zu rechnen? Da das Ergebnis einer Anfrage beim IR ohnehin nicht als *korrekt* oder *nicht korrekt* eingestuft werden kann, ist es u.U. vertretbar, leichte Fehler im Ergebnis zuzulassen, wenn dadurch z.B. die Laufzeit erheblich verbessert wird. Typische Anwendungen von IR-Systemen sind daher eher **unsensitiv** gegenüber kleineren Fehlern. Deshalb werden bei Algorithmen zum IR immer wieder auch Varianten diskutiert, die mathematisch nicht völlig korrekt arbeiten. Wir werden selbst in diesem Kurs solche Algorithmen kennenlernen.

1.4 Das grundsätzliche Vorgehen beim Information Retrieval

Nachdem wir uns ausführlich mit der Abgrenzung und Charakterisierung von IR und IR-Systemen beschäftigt haben, erscheint es an der Zeit sich der Arbeitsweise solcher Systeme zu nähern. In der Abbildung 1.8 ist das grundsätzliche Vorgehen beim Information Retrieval dargestellt.

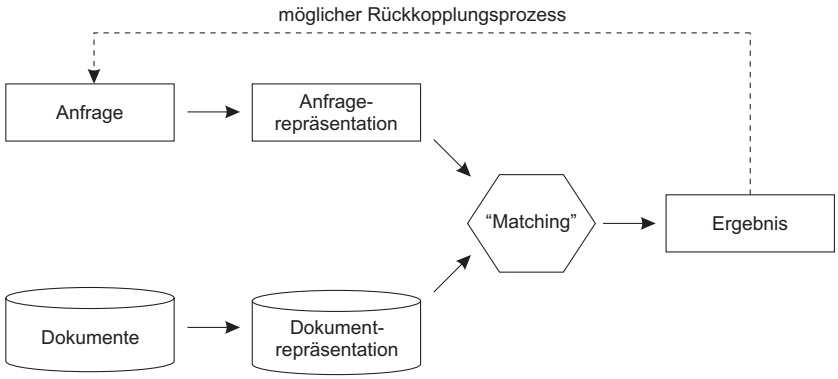


Abbildung 1.8 — Verallgemeinertes Modell der Anfragebearbeitung beim Information Retrieval

Mittels einer Anfrage – die einen zugrundeliegenden Informationswunsch repräsentiert – sollen aus einem Pool von Dokumenten relevante Dokumente in Form eines Ergebnisses gefunden werden. Dazu wird zunächst die Anfrage, die bspw. als natürlich sprachlicher Text von einem Benutzer eingegeben wird, in eine Anfragerepräsentation überführt, die maschinell leichter verarbeitet werden kann. Analog dazu können aus den Dokumenten entsprechende Dokumentenrepräsentationen abgeleitet werden. Durch ein Matching der Anfragerepräsentation mit den Dokumentenrepräsentationen gewinnen wir ein Ergebnis.

Dokumentenrepräsentation

Ein möglicher Rückkopplungsprozess kann im Information Retrieval bspw. durch das sog. Relevance Feedback erfolgen. **Relevance Feedback** bedeutet die manuelle Gewichtung der Einzelergebnisse durch den Benutzer nach deren Verwertbarkeit bzw. Relevanz für ihn. Durch diese Gewichtung bekommt ein Information Retrieval System zusätzliches Wissen über den Informationswunsch des Benutzers und kann dadurch die Anfrage oder die Anfragerepräsentation geeignet modifizieren um zu

Relevance Feedback

einem verbesserten Ergebnis zu gelangen.

Im Folgenden betrachten wir exemplarisch zwei konkrete Modelle des Information Retrieval um einen ersten Eindruck von den unterschiedlichen Möglichkeiten zur Ausprägung des oben skizzierten allgemeinen Modells zu erhalten. Wir werden diese Modelle im weiteren Verlauf des Kurses noch im Detail betrachten.

1.4.1 Boolesches Retrieval

Boolesches Retrieval

Das Boolesche Retrieval geht von einem **Informationswunsch** des Anwenders aus. Diesen Informationswunsch muss der Anwender selbst in eine Boolesche Anfragerepräsentation umformen. Diese Anfragerepräsentation besteht üblicherweise aus Begriffen (auch *Terme* genannt), die durch Boolesche Operatoren wie AND und OR miteinander verknüpft sind. In der Abbildung 1.9 wird »Wirtschaft AND Industrialisierung« als einfaches Beispiel genutzt.

Term

Die **Dokumente** liegen zunächst als Fließtext – d.h. als Abfolge von Wörtern – vor. Jedes Dokument wird nun durch die Menge der in ihm vorkommenden Begriffe repräsentiert. Dabei können sehr häufig vorkommende Wörter, wie das Wort »und«, die für sich genommen meist keine eigene Semantik tragen, weggelassen werden. Diese Wörter werden als **Stoppwörter** bezeichnet.

Stoppwort

Innerhalb des **Matchings** werden nun die Dokumente gesucht, deren enthaltene Begriffe der Anfrage entsprechen. Lautet die Anfrage z.B. *(Haus OR Garten) AND Italien*, so werden nur die Dokumente in das Ergebnis aufgenommen, in deren Repräsentation der Begriff *Italien* und mindestens einer der beiden Begriffe *Haus* oder *Garten* auftreten. Das auf diese Weise gewonnene Ergebnis ist eine unstrukturierte Menge von Dokumenten, die der Booleschen Anfrage entsprechen. Je nach Formulierung der Anfrage kann dieses Ergebnis leicht leer oder aber sehr umfangreich sein. Die Aufgabe des Anfragers besteht dann darin, durch geeignete Erweiterungen oder Vereinfachungen der Anfrage eine geeignete Ergebnisgröße zu gewinnen.

Zusammenfassend lässt sich für das Boolesche Retrieval festhalten:

- Die Dokumente werden als Menge der in ihnen vorkommenden Terme repräsentiert.

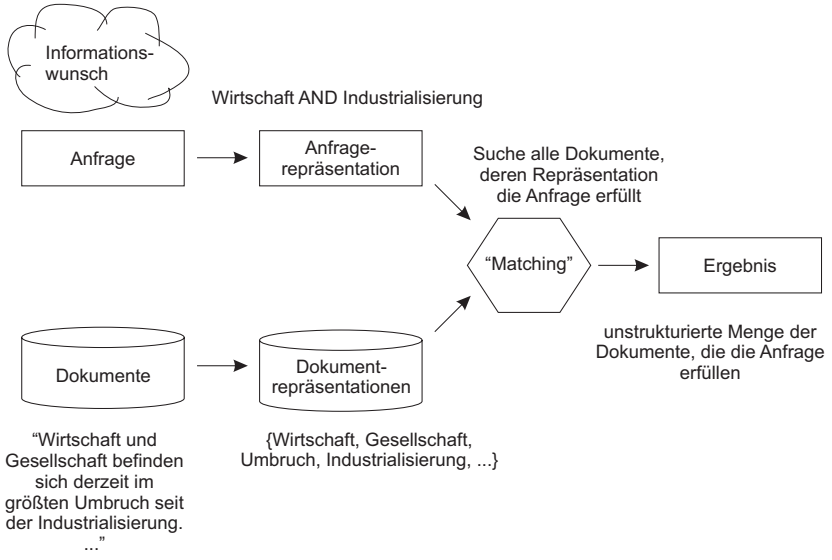


Abbildung 1.9 — Modell der Anfragebearbeitung beim Booleschen Retrieval

- Die Formulierung des Informationswunsches erfolgt durch einen Booleschen Ausdruck über die Terme, die in einem Dokument vorkommen sollen bzw. dürfen.

Beispiel: Suchanfrage mittels des Booleschen Retrievals

Lassen Sie uns als Beispieldokument den folgenden Ausschnitt aus einer Prüfungsordnung betrachten:

- **Dokument x:**
  »Die Regelstudienzeit beträgt einschließlich der Zeit für Abschlussprüfung und Anfertigung der Diplomarbeit acht Semester und drei Monate. Das Studium gliedert sich in ein viersemestriges Grundstudium und ein viersemestriges Hauptstudium. Der Höchstumfang der erforderlichen Lehrveranstaltungen im Pflicht- und Wahlpflichtbereich beträgt 136 Semesterwochenstunden. Das gemäß § 29 Abs. 1 Nr. 5 abzuleistende Praktikum wird auf die Regelstudienzeit nicht angerechnet.«

Wenn wir davon ausgehen, dass Dokumente durch die in ihnen vorkommenden Substantive im Nominativ Singular repräseniert werden, dann würde sich für dieses Dokument folgende Repräsentation ergeben:

- **Repräsentation für Dokument x:**
  »Abschlußprüfung«, »Anfertigung«, »Diplomarbeit«, »Grundstudium«, »Hauptstudium«, »Höchstumfang«, »Lehrveranstaltung«, »Monat«, »Pflicht«, »Praktikum«, »Regelstudienzeit«, »Semester«, »Semesterwochenstunde«, »Studium«, »Wahlpflichtbereich«, »Zeit«

Man kann sich natürlich fragen: warum Substantive im Nominativ Singular? An dieser Stelle mag dazu der Hinweis genügen, dass man auf irgendeine Weise das Problem angehen muss, dass Wörter in sehr verschiedenen Formen in Texten vorkommen. Übernimmt man die Form des Vorkommens im Text direkt in die Repräsentation, so ergibt sich das Problem, dass der Anfrager das gleiche Wort in einer anderen Form in seiner Anfrage haben kann. Man ist deshalb bestrebt, die Wortformen zu vereinheitlichen – was allerdings weder technisch trivial noch semantisch unproblematisch ist.

Nehmen wir nun folgende Anfrage an, so sollte zu der Anfrage unser obiges Beispieldokument in der Ergebnismenge liegen:

- **Beispielanfrage:**
  Studium AND Diplomarbeit AND NOT Note

Schwächen und mögliche Verbesserungen des Booleschen Retrievals

Das Boolesche Retrieval weist eine ganze Reihe von **Schwächen** auf. Dabei ist allerdings zu berücksichtigen, dass der Begriff *Boolesches Retrieval* sehr breit verwendet wird. Er bezeichnet einfache Systeme, die im Prinzip einer leicht modifizierten Zeichenkettensuche entsprechen, bis hin zu komplexen Internet-Suchmaschinen mit einer sehr umfangreichen und mächtigen Anfragesprache. Gemeinsam ist all diesen Systemen nur die Formulierung der Anfrage durch einen Booleschen Ausdruck. Die folgenden Schwachpunkte treten daher bei diesen Systemen natürlich unterschiedlich stark auf:

- In der Grundauslegung des Booleschen Retrievals gibt es – im Gegensatz zu unserem Beispiel auf Seite 42 – **keine Rückführung der Wörter auf ihre Grundform**. Die Begriffe werden in der Form des Vorkommens im Text in die Repräsentation übernommen. Dadurch findet eine Suche nach dem Wort »Haus« keine Dokumente, die nur das Wort »Häuser« enthalten.

- Es findet **keine Gewichtung der Wörter** statt, bspw. nach dem Ort des Vorkommens oder der Häufigkeit des Vorkommens.

- Es erfolgt **keine Zerlegung von Mehrwortgruppen**. D.h. zusammengesetzte Wörter werden durch Teilwörter nicht gefunden, bspw. werden keine Dokumente die nur das Wort »Bundeskanzlerwahl« enthalten, mittels einer Suche nach »Wahl« gefunden.

- Die **Formulierung der Suchanfrage ist relativ aufwendig**, da der Anwender den eigenen Informationswunsch in einem Booleschen Ausdruck formulieren muss.

- Über die **Größe eines Suchergebnisses** lässt sich keine klare Vorhersage treffen. Meist muss ein Benutzer schrittweise seine Anfrageformulierung aufgrund der Ergebnismenge selbst verfeinern.

- Das **Ergebnis** der Anfrage ist eine **unstrukturierte Menge**. Eine Sortierung der Elemente dieser Menge nach einem Kriterium wie der »Relevanzwahrscheinlichkeit« existiert nicht.

Den obigen Schwachstellen kann nun durch verschiedene **Möglichkeiten zur Verbesserung** des Booleschen Retrievals begegnet werden:

- Alle Wörter könnten auf ihre Grundform zurückgeführt werden.

- Der Vorkommensort eines Wortes könnte berücksichtigt werden. Wörter könnten je nach ihrem Vorkommen im Titel, in einer Überschrift oder in einer Zusammenfassung entsprechend gewichtet werden.

- Zusätzlich könnte die Unterscheidungskraft eines Wortes berücksichtigt werden. D.h. kommt ein Wort in vielen Dokumenten vor, so ist seine Unterscheidungskraft niedrig und vice versa.

- Weitere, die Relevanz eines Dokumentes beeinflussende Faktoren könnten berücksichtigt werden. Dies sind bspw. das Erstellungsdatum und der Autor eines Dokumentes.

- Ziel ist die Berechnung einer Kennzahl für jedes Dokument, die eine Abschätzung der Relevanzwahrscheinlichkeit ermöglicht. Auf Basis dieser Kennzahl kann eine Ergebnismenge von Dokumenten entsprechend sortiert werden, man spricht dann von der Aufstellung eines Rankings. *Ranking*

- Durch eine iterative Verbesserung des Ergebnisses, bspw. durch die Integration von Relevance Feedback können die gefundenen Ergebnisse verbessert werden.

Trotz der zahlreichen Schwächen des Booleschen Retrievals stellt es – wenn auch fast immer stark modifiziert – eines der heute am häufigsten eingesetzten Verfahren des Information Retrieval dar. Wir werden uns daher in diesem Kurs intensiv mit den Techniken des Booleschen Retrieval und ihrer Implementierung beschäftigen.

1.4.2 Das Vektorraummodell

Ein anderes Verfahren des Information Retrieval stellt das Vektorraummodell dar. Dieses Verfahren wurde von Salton [SM83] bereits 1960 vorgestellt. Bis heute schneiden bei Vergleichsmessungen Verfahren, die auf dem Vektorraummodell basieren, hervorragend ab. *Vektorraummodell*

Auch das Vektorraummodell geht von einem **Informationswunsch** des Anwenders aus (vgl. Abbildung 1.10). Diesen Informationswunsch kann der Anwender nun entweder als *Langtext in natürlicher Sprache* oder auch in Form eines zu diesem Informationswunsch relevanten *Beispieldokumentes* angeben.

Aus der textuellen Anfrage oder dem Beispieldokument wird nun eine **Anfragerepräsentation** in Form eines Anfragevektors berechnet. Dieser Vektor enthält pro Begriff eines Vokabulars eine Vektorkomponente.

Das Vokabular wiederum setzt sich aus einem Katalog von typischerweise einigen tausend oder gar einigen zehntausend auf ihre Stamm- oder Grundform reduzierten Wörtern zusammen. Stoppwörter sind in diesem Katalog meist eliminiert.

Die Dimension des Anfragevektors entspricht damit dem Umfang des Vokabulars. Jede einzelne Vektorkomponente innerhalb des Anfragevektors bestimmt nun, wie bedeutend ein Wort des Vokabulars für die Anfrage ist. Da jedoch sehr viele Vektorkomponenten 0 sind, lässt sich ein derartiger Vektor sehr kompakt repräsentieren, indem man nur diejenigen Komponenten speichert, die ungleich 0 sind.

Der Wert der einzelnen Vektorkomponenten lässt sich vereinfacht auf die Gleichung bringen: Anzahl der Vorkommen eines Begriffes im Text geteilt durch die Anzahl der Dokumente in denen der Begriff vorkommt. Dadurch wird die Unterscheidungskraft der einzelnen Begriffe in die Gewichtung mit einbezogen.

Beschreibungsvektor

Analog zur Berechnung des Anfragevektors werden die einzelnen **Dokumente** ebenfalls mittels Beschreibungsvektoren repräsentiert.

Skalarprodukt

Innerhalb des **Matchings** wird nun das Skalarprodukt aus dem Anfragevektor und den einzelnen Dokumentenvektoren berechnet (d.h. die Summe über die Produkte der einzelnen Komponenten; näheres findet sich auf Seite 49 oder z.B. auf der Seite zum Skalarprodukt bei *Wikipedia* [11]. Ein hohes Skalarprodukt stellt eine hohe Übereinstimmung der Anfrage mit dem entsprechenden Dokument dar. Sortiert man die Dokumente nach den resultierenden Skalarprodukten absteigend, erhält man auf diesem Weg ein Ranking der Dokumente nach ihrer Relevanz.

Auch im Vektorraummodell ist ein **Rückkopplungsprozess** mittels Relevance Feedback möglich. Dabei werden die Dokumentvektoren von Ergebnisdokumenten, die ein Benutzer als relevant bzw. nicht relevant erachtet, in den Anfragevektor mit eingerechnet, um die Anfrage bzw. Anfragerepräsentation geeignet zu modifizieren.

Beispiel: Berechnung einer Suchanfrage im Vektorraummodell

Um das Vorgehen beim Vektorraummodell besser verstehen zu können, erscheint die Betrachtung eines kleinen Beispiels sinnvoll.

[11] *http://de.wikipedia.org/wiki/Skalarprodukt*, letzter Abruf: 18.10.2006

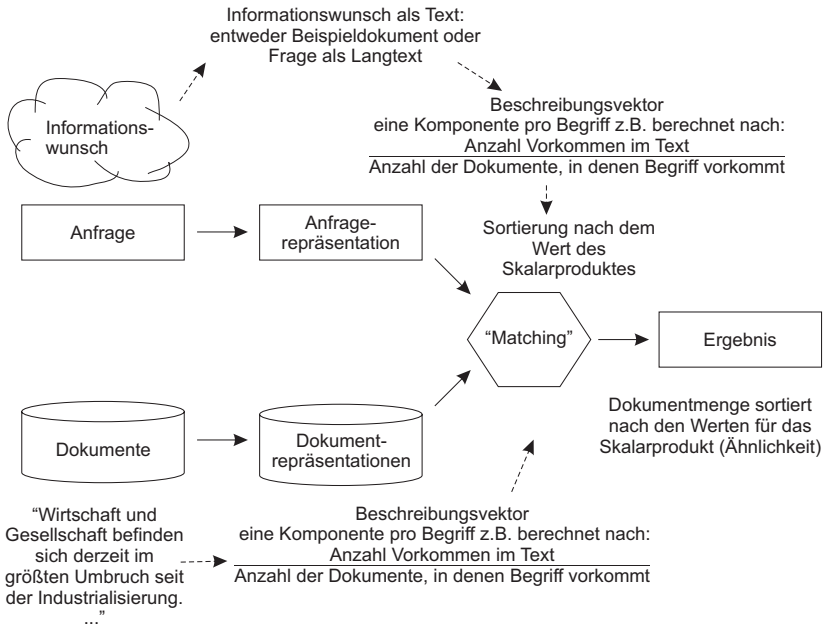


Abbildung 1.10 — Modell der Anfragebearbeitung beim Vektorraummodell

Ausgangspunkt der Überlegungen ist beim Vektorraummodell immer ein Vokabular mit t Begriffen. Dieses Vokabular entnehmen wir in diesem Beispiel den Dokumenten.

Dazu benötigen wir noch einige **Definitionen**:

- N sei die Anzahl der Dokumente in unserer Dokumentenkollektion.
- n_k ist die Anzahl der Dokumente in unserer Dokumentenkollektion die den Begriff bzw. Term k enthalten (offensichtlich gilt $0 \leq n_k \leq N$ für $k \in 1, \ldots, t$).
- tf_{dk} ist die Vorkommenshäufigkeit von Begriff k in Dokument D ($k \in 1, \ldots, t$).
- Ein Dokument D wird durch den Vektor $\mathcal{D} = (w_{d1}, w_{d2}, \ldots, w_{dt})$ repräsentiert.
- w_{dk} entspricht der Relevanz eines Dokumentes D für den Begriff k.

Mittels der folgenden Formel lassen sich nun auf Basis des Vokabulars die einzelnen Vektorkomponenten für ein Dokument berechnen.

$$w_{dk} = \frac{tf_{dk} \cdot \log \frac{N}{n_k}}{\sqrt{\sum_{i=1}^{t} \left(tf_{di} \cdot \log \frac{N}{n_i} \right)^2}} \tag{1.1}$$

Die Interpretation dieser Formel zerfällt im Wesentlichen in drei Teilaspekte:

- Zunächst verwendet die Formel die Vorkommenshäufigkeit tf_{dk} eines Terms, um das Gewicht für einen Term bei einem Dokument, in dem dieser Term häufiger auftritt, zu erhöhen. Damit wird letztlich das Gewicht des Terms durch einfaches Abzählen bestimmt.

Trennschärfe

- Zusätzlich sollen besonders »trennscharfe« Begriffe stärker gewichtet werden. Im Prinzip läuft dies darauf hinaus, die über die Dokumentenkollektion hinweg eher seltenen Terme in ihrem Gewicht etwas anzuheben. Hierzu wird der Faktor $\log \frac{N}{n_k}$ verwendet[12], der auch als *invers document frequency* (IDF) bezeichnet wird. Der Quotient $\frac{N}{n_k}$ ist dabei für Terme, die nur in wenigen Dokumenten vorkommen, recht hoch, während er für Terme, die in vielen Dokumenten vorkommen, recht niedrig ist. Man spricht deshalb von der inversen Dokumentenfrequenz des Terms. Durch den Logarithmus in $\log \frac{N}{n_k}$ wird der Effekt etwas abgeschwächt. Dennoch bleiben die Faktoren für seltene Begriffe höher als für häufige, was auch genau der gewünschte Effekt ist.

IDF

- Schließlich bleibt noch das Problem, dass bei einer Formel, die sich nur aus Termfrequenz und inverser Dokumentenfrequenz bilden würde, lange Dokumente bevorteilt wären. Sie hätten relativ hohe tf_{dk}-Werte, was in der Folge bei Anfragen zu recht hohen Skalarprodukten führen würde. Kurze Dokumente hätten kaum eine Chance in einem Ergebnis weit vorne zu landen. Als Ausweg aus diesem Problem eliminiert man den Einfluss der Länge der Dokumente durch eine Normierung auf Einheitsvektoren.

Einheitsvektoren

[12] Wie in technischen Anwendungen (so z.B. auf Taschenrechnern) üblich, verstehen wir hier und im Folgenden unter »log« den dekadischen Logarithmus zur Basis 10. Man beachte, dass es an dieser Stelle eigentlich egal ist, ob man den natürlichen

Analog zu den Dokumenten wird die Anfrage durch den Vektor $\mathcal{Q} = (w_{q1}, w_{q2}, \ldots, w_{qt})$ repräsentiert. Für die Berechnung der Koeffizienten in diesem Vektor könnte man dabei wie bei den Dokumenten auch die vorherige Formel 1.1 benutzen. Jedoch würden sich, da die Anzahl der Gesamtdokumente der Anfrage 1 beträgt, Probleme mit einer zu geringen Gewichtung der Terme in der Anfrage ergeben. Aus diesem Grund wird für die Berechnung der einzelnen Vektorkomponenten der Anfrage folgende Formel verwendet. Diese Formel weist jedem in der Anfrage vorkommenden Term eine Gewichtung von mindestens 0,5 zu.

$$w_{qk} = \begin{cases} \left(0.5 + \frac{0.5 \cdot tf_{qk}}{\max\limits_{1 \le i \le t} tf_{qi}}\right) \cdot \log \frac{N}{n_k} & \text{wenn} \quad tf_{qk} > 0 \\ 0 & \text{wenn} \quad tf_{qk} = 0 \end{cases} \tag{1.2}$$

Die Ähnlichkeit zwischen einer Anfrage und einem Dokument kann dann durch das Skalarprodukt beschrieben werden:

$$similarity(\mathcal{Q}, \mathcal{D}) = \sum_{k=1}^{t} w_{qk} \cdot w_{dk} \tag{1.3}$$

Diese Ähnlichkeitswerte können nun verwendet werden, um eine Sortierung – ein Ranking – für die Dokumente im Hinblick auf die Anfrage zu gewinnen.

Beispiel zum Vektorraummodell

Um das Vorgehen beim Vektorraummodell noch verständlicher zu machen, betrachten wir ein einfaches Beispiel in dem Dokumente D_1 bis D_5 und eine Anfrage Q gegeben sind:

- D_1 = »Häuser in Italien«
- D_2 = »Häuser in Italien und um Italien«

Logarithmus (zur Basis $e = 2,71828\ldots$) oder den dekadischen Logarithmus verwendet, da für ein beliebiges x gilt $\ln(x) = 2,3026\ldots \cdot \log(x)$, so dass sich beide nur um einen konstanten Faktor unterscheiden.

- D_3 = »Gärten und Häuser in Italien«
- D_4 = »Gärten in Italien«
- D_5 = »Gärten und Häuser in Frankreich«
- Q = »Häuser in Italien«

Damit besteht das Vokabular nach der Entfernung der Stoppworte »in«, »um« sowie »und« aus den Worten:

Häuser, *Italien*, *Gärten*, *Frankreich*

Das Vokabular hat damit die Dimension $t = 4$ und die Anzahl der Dokumente beträgt $N = 5$.

Für die Beschreibungsvektoren legen wir nun eine Reihenfolge fest:

- *Häuser* = Komponente 1
- *Italien* = Komponente 2
- *Gärten* = Komponente 3
- *Frankreich* = Komponente 4

Für D_2 enthalten wir beispielsweise folgende Termfrequenzen:

$tf_{2;1} = 1$, $tf_{2;2} = 2$, $tf_{2;3} = 0$, $tf_{2;4} = 0$

Ferner gilt:

$n_1 = 4$, $n_2 = 4$, $n_3 = 3$, $n_4 = 1$

Daraus ergibt sich nun folgender Dokumentenvektor:

$$\mathcal{D}_2 = \begin{pmatrix} \frac{1 \cdot \log \frac{5}{4}}{\sqrt{\left(1 \cdot \log \frac{5}{4}\right)^2 + \left(2 \cdot \log \frac{5}{4}\right)^2 + \left(0 \cdot \log \frac{5}{3}\right)^2 + \left(0 \cdot \log \frac{5}{1}\right)^2}} \\ \frac{2 \cdot \log \frac{5}{4}}{\sqrt{\left(1 \cdot \log \frac{5}{4}\right)^2 + \left(2 \cdot \log \frac{5}{4}\right)^2 + \left(0 \cdot \log \frac{5}{3}\right)^2 + \left(0 \cdot \log \frac{5}{1}\right)^2}} \\ \frac{0 \cdot \log \frac{5}{3}}{\sqrt{\left(1 \cdot \log \frac{5}{4}\right)^2 + \left(2 \cdot \log \frac{5}{4}\right)^2 + \left(0 \cdot \log \frac{5}{3}\right)^2 + \left(0 \cdot \log \frac{5}{1}\right)^2}} \\ \frac{0 \cdot \log \frac{5}{1}}{\sqrt{\left(1 \cdot \log \frac{5}{4}\right)^2 + \left(2 \cdot \log \frac{5}{4}\right)^2 + \left(0 \cdot \log \frac{5}{3}\right)^2 + \left(0 \cdot \log \frac{5}{1}\right)^2}} \end{pmatrix} = \begin{pmatrix} 0,4472 \\ 0,8944 \\ 0 \\ 0 \end{pmatrix}$$

Wie wir sehen, wird aufgrund der höheren Termfrequenz von $tf_{2;2} = 2$ gegenüber $tf_{2;1} = 1$ das Wort »Italien« im Vergleich zu dem Wort »Häuser« höher gewichtet, obwohl sie beide gleiche Vorkommensfrequenzen von $n_1 = 4$ und $n_2 = 4$ besitzen.

Insgesamt erhalten wir folgende Beschreibungsvektoren:

$$\mathcal{D}_1 = \begin{pmatrix} 0,7071 \\ 0,7071 \\ 0 \\ 0 \end{pmatrix}, \mathcal{D}_2 = \begin{pmatrix} 0,4472 \\ 0,8944 \\ 0 \\ 0 \end{pmatrix}, \mathcal{D}_3 = \begin{pmatrix} 0,3716 \\ 0,3716 \\ 0,8508 \\ 0 \end{pmatrix},$$
$$\mathcal{D}_4 = \begin{pmatrix} 0 \\ 0,4003 \\ 0,9164 \\ 0 \end{pmatrix}, \mathcal{D}_5 = \begin{pmatrix} 0,1310 \\ 0 \\ 0,2999 \\ 0,9449 \end{pmatrix}, \mathcal{Q} = \begin{pmatrix} 0,0969 \\ 0,0969 \\ 0 \\ 0 \end{pmatrix}$$

Der Anfragevektor ergibt sich dabei nach der hierfür beschriebenen Formel. Da der maximale tf-Wert $= tf_{q;1} = tf_{q;2} = 1$ ist, erhalten wir für die erste und zweite Komponente des Anfragevektors je $(0,5 + (0,5 * 1)/1) * \log \frac{5}{4} = 1 * \log \frac{5}{4} = 0,0969$.

Und im Vergleich mit der Anfrage ergeben sich folgende Ähnlichkeitswerte, die sich zu einem Ranking sortieren lassen.

1. D_1 mit $similarity(\mathcal{Q}, \mathcal{D}_1) = 0,137$
2. D_2 mit $similarity(\mathcal{Q}, \mathcal{D}_2) = 0,130$
3. D_3 mit $similarity(\mathcal{Q}, \mathcal{D}_3) = 0,072$
4. D_4 mit $similarity(\mathcal{Q}, \mathcal{D}_4) = 0,039$
5. D_5 mit $similarity(\mathcal{Q}, \mathcal{D}_5) = 0,013$

Damit ist das Dokument 1 zur Anfrage am ähnlichsten, dicht gefolgt von Dokument 2, in dem ja der Begriff *Italien* zweimal vorkam. Das zusätzliche Vorkommen des Begriffs *Gärten* wirft Dokument 3 zurück, obwohl auch dieses Dokument beide Anfrageterme enthält. Der Grund hierfür ist recht leicht zu verstehen. Durch das seltenere Vorkommen des Begriffs *Gärten* in den Dokumenten, wird die dritte Komponente im Beschreibungsvektor des Dokumentes recht hoch gewichtet, was aufgrund der

Normierung zu Lasten der ersten beiden Komponenten geht. Dadurch fällt deren Bewertung, die ja in die Berechnung des Skalarproduktes eingeht, recht schwach aus. Diese Überlegungen lassen sich analog für die Dokumente 4 und 5 fortsetzen.

1.4.3 Probabilistisches Information Retrieval

Probabilistisches Information Retrieval

Ein weiteres Modell des Information Retrieval stellt das Probabilistische Information Retrieval dar. Wir wollen es hier am Beispiel des so genannten *Parameter-Lernens* kurz betrachten, um damit bereits die Vielfalt der denkbaren Modelle des IR zu verdeutlichen.

Das Vorgehen lässt sich in folgende Punkte gliedern:

- Man wähle zuerst einige quantifizierbare, die Relevanz eines Dokumentes D in Relation zu einer Anfrage A bestimmende Eigenschaften aus. Beispiele für solche Eigenschaften könnten sein:
  - die Anzahl der übereinstimmenden Terme in Anfrage und Dokument,
  - die Anzahl der Terme aus der Anfrage, die im Titel des Dokumentes auftreten,
  - die Anzahl der Terme im Dokument oder
  - die Vorkommenshäufigkeit der drei »wichtigsten« gemeinsamen Terme im Dokument.
  - ...
- Man fasse nun die Werte dieser Eigenschaften (es sollten weit mehr als die oben angeführten vier sein) für ein konkretes Dokument D und eine konkrete Anfrage A zu einem Vektor V_{DA} zusammen. Dieser Vektor beschreibt nun die Eigenschaften der Beziehung zwischen Dokument und Anfrage.
- Man definiere eine Skala für die Relevanz eines Dokumentes, z.B. auf einer Skala von $0 - 10$. Der Wert 0 bezeichnet ein auf eine Anfrage bezogen irrelevantes Dokument und 10 ein sehr relevantes Dokument.
- Man bestimme nun für eine Beispielmenge von Dokumenten und entsprechenden Anfragen manuell die Relevanzwerte R. Diese Relevanzwerte liegen auch im Intervall zwischen 0 und 10.

- Man bestimme eine Funktion F, die
  - aus der Menge der Beschreibungsvektoren in das Intervall $[0; 10]$ abbildet und
  - für die gegebene Stichprobe die Abweichungsquadrate minimiert.
- Nun kann man F auf beliebige Kombinationen aus Anfragen und Dokumenten anwenden. Auf diese Art erhält man für eine beliebige Anfrage ein Ranking der Dokumente. Das Ranking kann durch Weiterentwicklung von F weiter verbessert werden.

Den Dreh- und Angelpunkt dieses Verfahrens bildet die Funktion F, bzw. ihre Bestimmung. Die Funktion muss zum einen handhabbar aber gleichzeitig leistungsfähig genug sein, um aus einem erlernten Trainingsset von Dokumenten und Anfragen auch für neu hinzukommende Dokumente und Anfragen eine adäquate Abbildung zu liefern. Zum anderen muss sich der Optimierungsaufwand für diese Funktion F in Grenzen halten.

Wir werden auf dieses und andere Modelle des probabilistischen IR später noch ausführlicher eingehen. An dieser Stelle mag der obige sehr grobe Überblick genügen, um einen groben Eindruck über die Breite der Modelle des IR zu vermitteln.

1.5 Formulierung eines Informationswunsches

Ein offensichtlich sehr grundlegendes Problem bei der Entwicklung eines IR-Modells oder IR-Systems ist die Formulierung des Informationswunsches. Ein Informationswunsch kann auf verschiedene Arten und Weisen von einem Benutzer formuliert bzw. von einem Information Retrieval System als Anfrage erwartet werden:

Informationswunsch

- Eine erste Möglichkeit ist die mehr oder weniger **unstrukturierte Angabe** von Suchtermen. Z.B. sucht die folgende Anfrage nach Dokumenten, welche jeweils die einzelnen Wörter enthalten: *Prozessor Intel Architektur Pentium*
- Eine zweite Möglichkeit stellt die Angabe eines ggf. syntaktisch erweiterten **Booleschen Ausdrucks** dar, z.B.: *Bundespräsident*

NEAR(4) Wahl AND FDP
Viele Suchmaschinen unterstützen bspw. einen sog. NEAR-Operator, der die Suche von benachbarten Wörtern ermöglicht. *NEAR(4)* stellt dabei eine Suchbedingung dar, bei der die betreffenden Suchwörter in Ergebnisdokumenten nicht mehr als die angegebene Zahl von Wörtern entfernt sein dürfen. In diesem Beispiel dürfen somit in Ergebnisdokumenten die Wörter *Bundespräsident* und *Wahl* nicht mehr als 4 Wörter entfernt voneinander stehen.

- Eine dritte Möglichkeit ist die Formulierung eines Informationswunsches in der Form eines **natürlich sprachlichen Textes**, z.B.: *Gesucht werden alle Zeitungsartikel, die sich mit der Rolle der FDP bei der Wahl des Bundespräsidenten beschäftigen.* Dies stellt aus dem Verständnis des Information Retrieval das Ideal für ein IR-System dar.

- Eine weitere Möglichkeit stellt die Anfrage an eine **Datenbank**, typischerweise formuliert in SQL, dar:
  SELECT $*$ **FROM** Artikel **WHERE** Text **LIKE** '%Bundespraesident%'

Von der Konzeption her kann man mit sehr unterschiedlichen Ideen an die Gestaltung der Möglichkeiten zur Formulierung der Anfrage herangehen. Möchte man eine möglichst präzise Anfrageformulierung ermöglichen, so bieten sich komplexe Boolesche Ausdrücke durchaus an. Andererseits wird der Benutzer dadurch bisweilen gezwungen, seinen Informationswunsch in einer Art und Weise zu formulieren, die seinem eigenen unvollständigen (oder zumindest unvollständig formalisierten) Wissen über den Informationswunsch nicht entspricht. Deshalb bieten Freitexteingaben letztlich die größeren Möglichkeiten.

1.6 Darstellung und Beurteilung von Suchergebnissen

Neben der Eingabe der Anfrage ist die zweite Schnittstelle des Systems zum Benutzer die Ergebnisausgabe. Ein wichtiger Aspekt hierbei ist die adäquate Darstellung des Ergebnisses. Ein anderer Aspekt ist der der Qualität des Ergebnisses.

Darstellung von Suchergebnissen

Zur Beurteilung der Darstellung von Suchergebnissen muss man zunächst Verfahren des Information Retrieval unterscheiden, die entweder **strukturierte** oder **unstrukturierte Ergebnisse** liefern. Das Ergebnis, das bspw. das Boolesche Retrieval oder auch eine Datenbankanfrage liefert, stellt eine unstrukturierte Menge von Dokumenten dar. Bei sehr großen Ergebnismengen müsste man per se die vollständige Menge an Elementen auf deren Relevanz überprüfen.

Eine Verbesserung stellt die Strukturierung der Ergebnismenge dar. Hierzu sollte über die absteigende Sortierung nach der Relevanz, ein sog. **Ranking** der Dokumente in der Ergebnismenge erfolgen. Bspw. kann man selbst mittels des Booleschen Retrievals über das *Coordination Level Match* ein Ranking ermöglichen, indem man in einem Suchergebnis die Dokumente zuerst anzeigt, die möglichst viele der Suchterme enthalten. Das Vektorraummodell erzeugt grundsätzlich ein Ranking.

Beurteilung von Suchergebnissen

Bei der Beurteilung von Suchergebnissen wollen wir zunächst in diesem Abschnitt unstrukturierte Ergebnisse betrachten. Die Beurteilung der Qualität von unstrukturierten Ergebnissen ist etwas einfacher. Strukturierte Ergebnisse werden wir später betrachten.

Für die Beurteilung der Ergebnisse gibt es die beiden Begriffe **Recall** und **Precision**.

Recall
Precision

- **Recall** drückt die Fähigkeit eines Information Retrieval Systems aus, relevante Dokumente in der Ergebnismenge zu liefern, d.h. wie hoch ist der Anteil der relevanten Dokumente im Ergebnis im Verhältnis zu allen relevanten Dokumenten in der gesamten Dokumentenkollektion. Ein Recall von bspw. 80% bedeutet, dass sich unter den gefunden Dokumenten 80% aller verfügbaren relevanten Dokumente befinden. Ziel ist, einen möglichst hohen Recall, also möglichst 100% zu erreichen.

- **Precision** gibt die Quote der gefundenen relevanten Dokumente unter allen gefundenen Dokumenten an, d.h. wie gut ist ein Information Retrieval System in der Lage, möglichst nur relevante

Dokumente in der Ergebnismenge zu liefern. Auch hier sind natürlich wieder 100% erstrebenswert. Eine Precision von 40% bedeutet hingegen, das 60% der Dokumente im Ergebnis nicht relevant sind.

Man beachte bei diesen Definitionen, dass vereinfachend angenommen wird, dass zu jedem Dokument in der Kollektion im Hinblick auf die Anfrage eindeutig ermittelt werden kann, ob es relevant oder nicht relevant ist. Man spricht dabei auch von einer zweiwertigen Relevanzskala.

Relevanzskala

$$\text{Recall} = \frac{\text{gefundene relevante Dokumente}}{\text{relevante Dokumente insgesamt}}$$

$$\text{Precision} = \frac{\text{gefundene relevante Dokumente}}{\text{gefundene Dokumente insgesamt}}$$

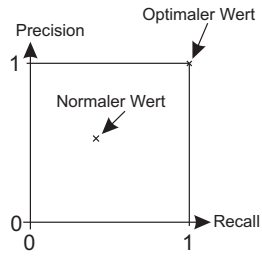


Abbildung 1.11 — Recall und Precision

Ziel ist natürlich die Realisierung eines Information Retrieval Systems, das auf eine Suchanfrage möglichst alle relevanten Dokumente (hoher Recall) und auch nur diese (hohe Precision) aus einem Dokumentenbestand findet (vgl. Abbildung 1.11). Typische Systeme erreichen allerdings heute nur Werte von ca. 40% sowohl für Recall und auch Precision.

Hat man es nun mit einem System zu tun, das ein Ranking der Dokumente liefert, so kann man zu diesem System einen so genannten Recall/Precision-Graph anfertigen, indem man nach jedem Rang die Werte für Recall und Precision bestimmt und in ein entsprechendes Koordinatensystem einträgt. Wir werden hierauf im nächsten Hauptabschnitt ausführlich eingehen.

Recall/Precision-Graph

Beispiel: Recall und Precision

Eine Suchanfrage liefert 10.000 Dokumente aus einem Gesamtbestand an Dokumenten mit N gleich 1.000.000. In dem Gesamtbestand befinden sich insgesamt 100 für die Suchanfrage relevante Dokumente, in der Ergebnismenge sind allerdings nur 80 davon enthalten.

Der Recall liegt somit bei einem guten Wert von 80%. Allerdings liegt die Precision bei einem sehr schlechten Wert von 0,8%. Dieses Beispiel zeigt u.a., dass Recall und Precision nicht isoliert betrachtet werden dürfen.

1.7 Zusammenfassung und Aufbau des Kurses

In diesem ersten Abschnitt haben wir den Begriff *Information Retrieval* sowie die historische Entwicklung und neuere Einflüsse eingeführt. Darüber hinaus wurden verschiedene Modelle des Information Retrieval skizziert.

Im weiteren Verlauf des Kurses wollen wir nun folgende Themen behandeln:

- die Evaluierung und Beurteilung von IR-Systemen
- die Berücksichtigung der Vagheit in natürlichsprachigen Texten, DublinCore, Semantic Web
- Einfache IR-Modelle und ihre Implementierung
- Das Vektorraummodell
- Alternativen zur globalen Suche, wie Klassifikationsansätze, Clustering und Browsing
- Weitere IR-Modelle
- Multimedia Information Retrieval
- Internetsuchmaschinen
- Ein Ausblick auf neuere Entwicklungen und innovative Anwendungen für IR-Systeme

2 Evaluierung und Beurteilung von IR-Systemen

Mit der **Evaluierung** und Beurteilung von IR-Systemen werden im Wesentlichen zwei Ziele verfolgt: *Evaluierung*

- Zum einen soll durch eine Evaluierung eine **vergleichende Betrachtung zwischen mehreren IR-Systemen** ermöglicht werden. Dies ist z.B. in Entscheidungssituationen wichtig, in denen aus mehreren angebotenen Systemen eines für den Kauf und späteren Einsatz ausgewählt werden muss.
- Das zweite Ziel unterstützt die **Konstruktion und Konfiguration eines IR-Systems**. Hierbei ist von Interesse, wie sich Modifikationen am System auf das Retrievalergebnis auswirken, d.h. ob die Ergebnisse besser oder schlechter werden. Dies kann z.B. dann wichtig sein, wenn man an einem bestehenden System eine Veränderung vornimmt – so könnte man etwa eine neue Art der Verarbeitung von Wortformen in das System integrieren. Die Frage ist nun, ob sich diese Veränderung positiv oder negativ auf das Verhalten des Systems auswirkt.

Um nun verschiedene IR-Systeme miteinander vergleichen zu können, müssen geeignete Evaluierungsmaße gefunden werden. *Evaluierungsmaße*

Exkurs: Evaluierung bei Datenbanken

Um uns dem Problem des Vergleichs zweier Systeme zu nähern, ist zunächst ein Blick auf die Evaluierung von verschiedenen Datenbanken interessant. Dabei interessieren primär zwei Ziele:

- Zum einen ist dies der **Funktionsumfang** des Datenbanksystems. Bei relationalen Datenbanken ist hier beispielsweise der von *Funktionsumfang*

SQL

der Datenbank unterstützte SQL -Standard von Interesse. Darüber hinaus kann das Vorhandensein von erweiterten Funktionalitäten für einen spezifischen Einsatzzweck wichtig sein – z.B. für die Verwaltung von Bildern oder Audiodokumenten.

- Im standardisierten Bereich, d.h. bei Anwendungen, die primär *Standard SQL* benutzen, ist vor allem die **Performance** wichtig – also ihre Laufzeit- und Speicherplatzeffizienz . Für die Ermittlung der Performance sind im Allgemeinen hersteller- und datenbankabhängige und -unabhängige Benchmarks verfügbar. Da von den Systemen bei derartigen »Standardanfragen« erwartet wird, dass sie absolut korrekt arbeiten, muss das Ergebnis bei allen Systemen gleich sein – ein System, das ein falsches Ergebnis liefert, würde ohnehin vom Vergleich ausgeschlossen. Wenn aber die Ergebnisse gleich sind, dann entscheidet nur die Frage wie schnell sie erbracht werden können.

Laufzeiteffizienz
Speicherplatzeffizienz

Evaluierungskriterien für IR-Systeme

Für die Evaluierung von IR-Systemen sind natürlich ebenfalls der **Funktionsumfang** und die **Laufzeit- und Speicherplatzeffizienz** von Interesse. Das dominierende Kriterium ist jedoch zunächst die **Qualität der Ergebnisse** . Der Grund hierfür ist, dass bei IR-Systemen keineswegs davon ausgegangen werden kann, dass sie auf eine Anfrage gleichwertige oder gar gleiche Antworten liefern. Jedes IR-System mag zu einer Anfrage unterschiedliche Dokumente für relevant halten. Es ist daher wichtig eine möglichst objektive Beurteilung der Ergebnisse mehrerer IR-Systeme zu ermöglichen. Mit dieser Problemstellung werden wir uns in diesem Abschnitt ausführlich beschäftigen.

Ergebnisqualität

Man mag sich fragen, warum wir uns zu einem so frühen Zeitpunkt im Kurs mit Fragen der Evaluierung von IR-Systemen beschäftigen. Dies liegt daran, dass Fragen zur Qualität des Ergebnisses bei IR-Systemen sehr zentral sind. Es ist deshalb nützlich, im weiteren Verlauf des Kurses immer ein »Handwerkszeug« zur Qualitätsbeurteilung zur Hand zu haben, um alle betrachteten Techniken und Modelle im Hinblick auf die Qualität ihrer Ergebnisse hinterfragen zu können.

2.1 Begriffsbildung

In diesem Kapitel wollen wir uns zunächst mit der Definition der Begriffe Effizienz, Effektivität und Relevanz beschäftigen.

2.1.1 Effizienz vs. Effektivität

In diesem Abschnitt soll der Unterschied zwischen den Begriffen Effizienz und Effektivität im Hinblick auf das Gebiet der Informatik im Allgemeinen und des Information Retrievals im Speziellen betrachtet werden.

Effizienz

Laufzeiteffizienz

Effizienz bezeichnet den möglichst sparsamen Umgang mit Ressourcen. In der Informatik sind dies typischerweise die Laufzeit- und die Speicherplatzeffizienz. Plakativ ausgedrückt kann man davon sprechen, ob das, was gemacht wird, richtig (*effizient*) ausgeführt wird.

Eine vergleichende Betrachtung verschiedener Systeme im Hinblick auf ihre Effizienz ist nur bei gleichwertigen bzw. identischen Ergebnissen sinnvoll. Es ist nicht sinnvoll, zwei Systeme hinsichtlich ihrer Effizienz miteinander zu vergleichen, wenn eines der beiden »korrekt« arbeitet, das andere aber nicht.

Effektivität

Effektivität setzt die Qualität eines Ergebnisses ins Verhältnis zum Aufwand, der für die Ermittlung des Ergebnisses erforderlich ist. Letztlich stellt Effektivität die Frage, ob die richtigen Dinge ausgeführt werden, denn wenn für eine Ergebnisverbesserung, die der Benutzer kaum oder gar nicht wahrnimmt, ein sehr hoher zusätzlicher Aufwand betrieben werden muss, dann sollte man es bei der einfacheren Lösung belassen. Das Kriterium der Effektivität zielt damit immer auf eine technische Lösung, bei der Aufwand und Nutzen in einem günstigen Verhältnis zueinander stehen.

Für das Information Retrieval ist eine reine Betrachtung der Effizienz eines IR-Systems nicht sinnvoll. Da bei Systemen des Information Retrieval vor allem die Qualität der Ergebnisse zählt, stellt die Effektivität eine sinnvolle Betrachtungsweise dar. Es werden Systeme benötigt, die in vertretbarer Zeit für den Benutzer brauchbare Ergebnisse liefern.

2.1.2 Relevanz

Relevanz

Ein sehr zentraler Begriff im Zusammenhang mit der Evaluierung von IR-Systemen ist der Begriff der *Relevanz* von Dokumenten. Dabei gibt es durchaus unterschiedliche Ansätze, diese Relevanz zu beurteilen. Während wir z.B. bei unseren Betrachtungen zum Probabilistischen Information Retrieval (Kapitel 1.4.3) in der Einführung die Relevanz eines Dokumentes in Bezug auf eine Anfrage auf einer linearen Skala von 0 bis 10 als mehr oder weniger ausgeprägt definieren konnten, definieren wir jetzt Dokumente binär als entweder *relevant* oder *nicht relevant*. Dies ist die übliche Methode der Relevanzeinstufung – wenngleich nicht die einzig denkbare.

Relevanzeinstufung, binäre

Bei der Evaluierung von IR-Systemen geht man basierend auf dieser binären Relevanzbeurteilung davon aus, dass bezogen auf eine Anfrage q die Dokumentenkollektion in zwei Teile zerfällt.

- R_q bezeichnet die Menge der zur Anfrage relevanten Dokumente in der Dokumentenkollektion und
- $\bar{R}_q$ die zur Anfrage nicht relevanten Dokumente. (Im Allgemeinen wird dabei R_q gegenüber der Dokumentenkollektion sehr klein sein.)

Die Annahme, dass die Dokumentenkollektion bezüglich der Anfrage q in die beiden Teilmengen R_q und $\bar{R}_q$ zerfällt, führt dazu, dass ein IR-System genau R_q als Antwort auf eine Anfrage liefern sollte. Damit kann man die Qualität des Ergebnisses durch seine Abweichung von R_q beschreiben.

Probleme der zweiwertigen Definition der Relevanz

Hinter der strikten Aufteilung in die beiden Mengen R_q und $\bar{R}_q$ verbergen sich offensichtlich einige Probleme:

- Zum einen hängt die Aufteilung vom **Vorwissen** des Anfragenden ab. So kann z.B. die Relevanz bestimmter Dokumente nur mit Vorwissen erkannt werden. Andererseits sind einführende Überblicksartikel nur für den Laien von Interesse. Letztlich kann eine Aufteilung der Dokumentenkollektion in die zu einer Anfrage relevanten und nicht relevanten Dokumente damit immer nur subjektiv sein.

- Zum anderen kann die Relevanz von Dokumenten **voneinander abhängig** sein. So kann z.B. die Relevanz eines Dokumentes b erst durch das Lesen eines Dokumentes a deutlich werden. Die Folge ist, dass wenn Dokument a im Ergebnis ist, dadurch auch Dokument b relevant wird. Fehlt Dokument a im Ergebnis, dann bleibt Dokument b irrelevant. Diese gegenseitigen Abhängigkeiten zwischen der Relevanz von Dokumenten werden von unserem einfachen binären Relevanzmodell natürlich nicht abgedeckt.
- Schließlich erscheint eine zweiwertige Relevanzskala **zu vereinfachend**. Eine mehrstufige oder gar stetige Skala wäre sicherlich realistischer. Allerdings ergibt sich daraus das Problem, dass eine allgemein anerkannte Einordnung der Dokumente in die Relevanzskala noch schwieriger würde, als dies beim Einsatz der zweiwertigen Skala der Fall ist. In der Praxis hat sich daher die zweiwertige Relevanzskala als gut handhabbar erwiesen.

2.2 Recall und Precision

Um nun die Abweichung des Ergebnisses einer Anfrage von R_q konkret fassen zu können, wollen wir zunächst unstrukturierte Ergebnisse betrachten, bei denen das System eine *Menge* von Dokumenten als Ergebnis liefert. Diese Menge kann mit R_q verglichen werden. Auf die Beurteilung strukturierter Ergebnisse werden wir später eingehen (siehe 2.2.4 Recall/Precision-Werte bei Systemen mit Ranking).

Zur Beurteilung der Ergebnisse werden in aller Regel die beiden Kenngrößen **Recall** und **Precision** verwendet:

Recall
Precision

- **Recall** drückt die Fähigkeit eines Information Retrieval Systems aus, relevante Dokumente in der Ergebnismenge zu liefern, d.h. wie hoch ist der Anteil der relevanten Dokumente im Ergebnis im Verhältnis zu allen relevanten Dokumenten in der Dokumentenkollektion. Ein Recall von beispielsweise 80% bedeutet, dass sich unter den gefundenen Dokumenten 80% aller verfügbaren relevanten Dokumente befinden. Ziel ist, einen möglichst hohen Recall (maximal 100%) zu erreichen.
- **Precision** gibt die Quote der gefundenen relevanten Dokumente unter allen gefundenen Dokumenten an, d.h. wie gut ist ein Information Retrieval System in der Lage, irrelevante Dokumente in

der Ergebnismenge zu vermeiden. Auch hier sind natürlich 100% erstrebenswert. Eine Precision von 40% bedeutet hingegen, dass 60% der Dokumente im Ergebnis nicht relevant sind.

$$\text{Recall} = \frac{\text{gefundene relevante Dokumente}}{\text{relevante Dokumente insgesamt}}$$

$$\text{Precision} = \frac{\text{gefundene relevante Dokumente}}{\text{gefundene Dokumente insgesamt}}$$

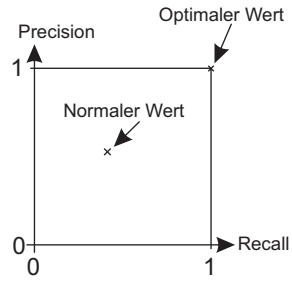


Abbildung 2.1 — Recall und Precision

Ein einführendes Beispiel zu Recall und Precision

Eine Suchanfrage liefert 1.000 Dokumente aus einem Gesamtbestand mit N gleich 1.000.000 Dokumenten. Im Gesamtbestand befinden sich 100 für die Suchanfrage relevante Dokumente, in der Ergebnismenge sind 80 davon enthalten.

Der Recall liegt somit bei einem guten Wert von 80%. Allerdings liegt die Precision nur bei 8%.

Dieses Beispiel zeigt u.a., dass Recall und Precision nicht einzeln betrachtet werden dürfen.

Eine genauere Betrachtung von Recall und Precision

Ausgangspunkt für die Bestimmung von **Recall** und **Precision** ist

- zum einen die Unterscheidung der Dokumente in *relevante Dokumente* und *nicht relevante Dokumente* und
- zum anderen die Unterscheidung der Dokumente in Dokumente, die sich *in der Ergebnismenge* befinden, und solche, die sich *nicht in der Ergebnismenge* befinden.

Diese Teilmengen werden wie folgt bezeichnet:

- Relevante Dokumente, die sich im Ergebnis befinden, werden als **Hits** bezeichnet. *Hits*
- Nicht relevante Dokumente, die sich im Ergebnis befinden, werden als **Noise** bezeichnet, weil sie wie Nebengeräusche bei einer Audioübertragung die Konzentration auf das Wesentliche – die relevanten Dokumente – erschweren. *Noise*
- Relevante Dokumente, die sich nicht im Ergebnis befinden, werden als **Misses** und *Misses*
- nicht relevante Dokumente, die sich auch nicht im Ergebnis befinden, als **Rejected** bezeichnet. *Rejected*

	relevant	nicht relevant	Σ
im Ergebnis	a (Hits)	b (Noise)	$a+b$
nicht im Ergebnis	c (Misses)	d (Rejected)	$c+d$
Σ	$a+c$	$b+d$	$a+b+c+d$

Tabelle 2.1 — Notationen zur Berechnung der Recall- und Precision-Werte

Auf Basis dieser Einteilung können wir den Recall und die Precision wie folgt definieren:

$$Recall = \frac{a}{a+c} = \frac{\text{Anzahl der relevanten Dokumente im Ergebnis}}{\text{Gesamtzahl der relevanten Dokumente}} \quad (2.1)$$

$$Precision = \frac{a}{a+b} = \frac{\text{Anzahl der relevanten Dokumente im Ergebnis}}{\text{Gesamtzahl der Dokumente im Ergebnis}} \quad (2.2)$$

Der Recall beantwortet bezogen auf eine Menge von Suchergebnissen die Frage *»Wie vollständig ist die Antwort?«* und die Precision beantwortet die Frage *»Wie genau ist die Antwort?«*.

Grafische Mengendarstellung von Recall und Precision

Die einzelnen Mengen, die zur Berechnung von Recall und Precision herangezogen werden, lassen sich grafisch wie in Abbildung 2.2 darstellen.

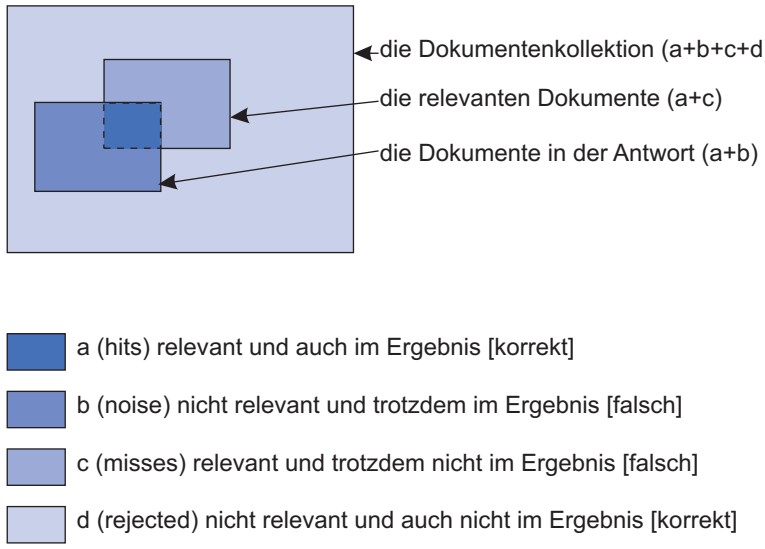


Abbildung 2.2 — Definitionen zur Berechnung von Recall und Precision

Eine äquivalente Definition von Recall und Precision

Äquivalent zu den oben gegebenen Definitionen kann man Recall und Precision auch auf Basis der Mengen R_q, das ist die Menge aller relevanten Dokumente in der Kollektion im Hinblick auf die Anfrage q, und Erg, das ist die Ergebnismenge, wie in Abbildung 2.3 definieren.

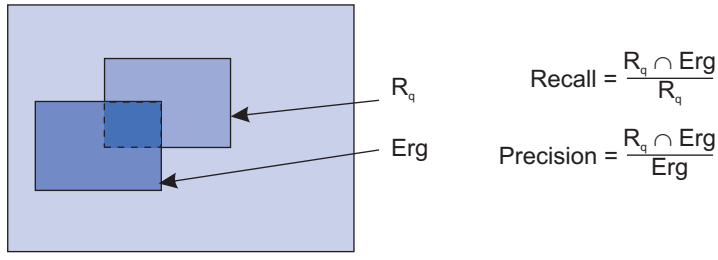


Abbildung 2.3 — Alternative Schreibweise zur Berechnung von Recall und Precision

Das F-Measure

F-Measure

Das F-Measure [Rij79] kombiniert die beiden klassischen Maße Recall und Precision in nur einer Kennzahl. Es ist definiert als gewichtetes harmonisches Mittel aus Recall und Precision. Das Gewicht α kann hierbei Werte zwischen 0 und 1 annehmen. Häufig findet die ungewichtete Variante mit $\alpha = 0.5$ Anwendung; somit sind Recall und Precision gleichgewichtet:

$$F - Measure = \frac{Precision \cdot Recall}{(1-\alpha) \cdot Recall + \alpha \cdot Precision}$$

2.2.1 Vergleich mehrerer Systeme aufgrund von Recall/Precision-Werten

Ausgangspunkt für den Vergleich mehrerer IR-Systeme ist zunächst die Ermittlung entsprechender Recall- und Precision-Werte für die einzelnen Systeme. Diese Werte können wir dann in einen so genannten **Recall/-Precision-Graphen** eintragen. Abbildung 2.4 zeigt einen Graphen mit Recall/Precision-Paaren für drei Systeme.

Recall/Precision-Graph

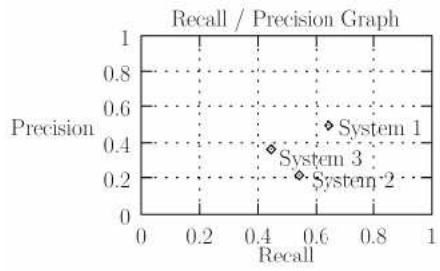


Abbildung 2.4 — Recall/Precision-Paare für drei IR-Systeme in einem Recall/Precision-Graphen

Ein ideales IR-System würde jeweils Recall- und Precision-Werte von 1 erreichen und demzufolge in der oberen rechten Ecke des Recall/Precision-Graphs angesiedelt sein. Ein sehr schlechtes System dagegen wird mit niedrigen Recall- und Precision-Werten in der linken unteren Ecke des Recall/Precision-Graphen wiederzufinden sein. So würde ein System, das letztlich nur zufällig irgendwelche Dokumente im

Ergebnis präsentiert, eine Precision von $|R_q|/N$ erzielen, wobei N die Gesamtzahl der Dokumente in der Kollektion und $|R_q|$ die Anzahl der relevanten Dokumente in der Kollektion bezeichnen. Ein solches »Zufallssystem« stellt sozusagen die Bottom-Line für reale Systeme dar. (Die Schreibweise $|X|$ werden wir auch im weiteren Verlauf des Kurses immer wieder verwenden, um die Anzahl der Elemente in einer Menge X zu bezeichnen.)

Zufallssystem

In unserem Beispiel sehen wir, dass *System 1* im Vergleich zu den beiden anderen Systemen als das beste System gelten kann, da es sowohl den höchsten Recall- als auch den höchsten Precision-Wert erreicht. Der Vergleich zwischen *System 2* und *System 3* fällt dagegen schwer, weil *System 2* einen höheren Recall- und dafür aber einen niedrigeren Precision-Wert besitzt. Hier ist also keine unmittelbare Aussage über die Vorteilhaftigkeit möglich.

Abbildung 2.5 zeigt, in welchen Bereichen sich im Hinblick auf ein bestimmtes System – in unserem Fall *System 3* – konkrete Aussagen darüber machen lassen, ob andere Systeme schlechter oder besser sind. In den nicht gekennzeichneten Bereichen ist keine Aussage möglich, weil immer eines der beiden Systeme einen besseren Recall und das andere eine bessere Precision hat, oder umgekehrt.

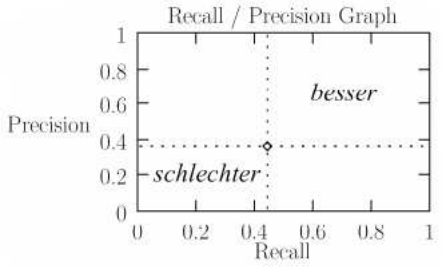


Abbildung 2.5 — Vorteilhaftigkeitsbeurteilung aufgrund von Recall/Precision-Werten

2.2.2 Bestimmung des Recall

Zur Bestimmung der Precision reicht die Betrachtung des vom IR-System gelieferten Ergebnisses Erg_q aus. Man muss also lediglich $|Erg_q|$ Dokumente betrachten und hinsichtlich ihrer Relevanz beurteilen. Da der

Umfang des Ergebnisses für typische Anfragen meist nicht sehr groß sein wird, ist dies eine lösbare Aufgabe. Wesentlich problematischer gestaltet sich die Bestimmung des Recalls, da hierzu $|R_q|$, die Anzahl der relevanten Dokumente in der gesamten Kollektion, bekannt sein muss, um $\frac{|R_q \cap Erg_q|}{|R_q|}$ zu berechnen. Wenn man zum Beispiel an eine Internetsuchmaschine denkt, wird schnell deutlich, dass eine exakte Bestimmung von $|R_q|$ kaum möglich ist.

In der Literatur werden deshalb unterschiedliche Näherungsmethoden zur Bestimmung des Recall vorgeschlagen (vgl. Fuhr[Fuh96]).

- Vollständige Relevanzbeurteilung einer repräsentativen Stichprobe
- Dokument-Source-Methode
- Anfrageerweiterung
- Abgleich mit externen Quellen

Vollständige Relevanzbeurteilung einer repräsentativen Stichprobe

Relevanzbeurteilung, repräsentative Stichprobe

In diesem Fall wählt man eine zufällige Stichprobe S im Umfang $|S|$ aus der Grundgesamtheit aller N Dokumente. In der Stichprobe ermittelt man nun den Anteil der relevanten Dokumente und rechnet diesen Anteil auf die gesamte Dokumentenkollektion hoch, um eine Abschätzung für $|R_q|$ zu erhalten.

Das Problem bei diesem Ansatz ist, dass im Allgemeinen der Anteil der relevanten Dokumente im Verhältnis zur Größe der gesamten Dokumentenkollektion sehr gering sein wird (z.B. $N = 1.000.000$ und $|R_q| = 20$). In diesem Fall muss die Stichprobe hinreichend groß sein, um mit statistischen Mitteln eine sichere Aussage treffen zu können. Eine umfangreiche Stichprobe, von beispielsweise einigen zehntausend Dokumenten würde aber einen sehr großen Aufwand bei der manuellen Relevanzbeurteilung der einzelnen Dokumente bedingen.

Relevanzbeurteilung, manuell

Aus diesem Grund ist diese Methode wenig praktikabel und wird entsprechend selten angewendet.

Dokument-Source-Methode

Dokument-Source-Methode

Einen anderen Weg beschreitet die Dokument-Source-Methode . Dabei

wählt man ein zufälliges Dokument aus der Dokumentenkollektion und formuliert eine Suchanfrage, zu der das gewählte Dokument relevant sein müsste. Diese Anfrage stellt man nun an das IR-System und überprüft, ob das Dokument tatsächlich in der Antwort enthalten ist. Wiederholt man dieses Verfahren über mehrere zufällig ausgewählte Dokumente und entsprechende Anfragen, so kann man den Recall durch das Verhältnis der Anfragen, bei denen das jeweils ausgewählte Dokument im Ergebnis war, zur Gesamtzahl der durchgeführten Anfragen annähern.

Ein Problem bei dieser Methode stellen die Anfragen dar, die meist wenig mit realen Benutzeranfragen gemein haben, da sie genau auf ein Dokument abzielen. Damit ist aber auch fraglich, ob sich der ermittelte Recall-Wert auf »normale« Anfragen übertragen lässt.

Anfrageerweiterung

Anfrageerweiterung

Das am häufigsten verwendete Verfahren ist die Anfrageerweiterung . Ziel ist es, eine Anfrage so zu erweitern, dass eine Obermenge der ursprünglichen Antwortmenge gefunden wird, die wesentlich größer ist und weitere relevante Dokumente enthält. Bspw. nimmt man in eine Suchanfrage Synonyme mit auf oder ersetzt Und- durch Oder-Verknüpfungen. Ergänzend kann man auch mehrere Frageformulierungen von verschiedenen Bearbeitern erstellen lassen und die Vereinigungsmenge der Antwortmengen betrachten.

Man nimmt nun vereinfachend an, dass die so gewonnene erweiterte Antwortmenge sicher alle zur ursprünglichen Anfrage relevanten Dokumente enthält, und kann so den Recall für die ursprüngliche Anfrage berechnen.

Ein Problem dieser Methode ist, dass man im Allgemeinen entgegen der Annahme nur eine Teilmenge der relevanten Dokumente in der erweiterten Antwort erhält. Dadurch wird der Recall fast immer zu hoch geschätzt. Dabei hängt die Genauigkeit der Schätzung des Recall maßgeblich von der Güte der Anfrageerweiterung ab. Nur wenn diese sehr gewissenhaft vorgenommen wird, kann man näherungsweise korrekte Schätzungen für den Recall erwarten.

Pooling

Eine Sonderform dieses Verfahrens ist das so genannte **Pooling** . Dabei werden die Ergebnismengen aller an einem Vergleich beteiligten Systeme für eine bestimmte Anfrage q vereinigt. Man nimmt dabei an, dass alle relevanten Dokumente von mindestens einem System gefunden werden. Man schätzt $|R_q|$ also durch die Anzahl der relevanten Dokumente in der

Vereinigungsmenge der Ergebnisse ab. Offensichtlich führt auch dieses Verfahren zu tendenziell zu guten Recall-Werten.

Abgleich mit externen Quellen

Abgleich mit externen Quellen

Eine der besten Methoden zur Bestimmung des Recalls stellt der Abgleich mit externen Quellen dar. Man versucht, parallel zur Suche in einem Information Retrieval System, mit davon unabhängigen Methoden relevante Dokumente zu bestimmen. Bspw. kann man den Anfragesteller oder andere Fachleute befragen, welche relevanten Dokumente – die natürlich auch dem System bekannt sein müssen – sie kennen. Der Anteil der Dokumente, die das System als Antwort liefert, an den bekannten, in der Kollektion vorhandenen relevanten Dokumenten ist dann eine Näherung für den Recall.

Pooling

Ein gravierendes Problem stellt auch hier der Aufwand dar, der zur manuellen Bestimmung von Relevanzurteilen notwendig ist. Trotz dieses Aufwands verwenden die TREC-Kollektionen (siehe Kapitel 2.3.1 Die TREC-Kollektion), die einige Hunderttausend Dokumente beinhalten, dieses Verfahren, um zu einem Katalog von Anfragen jeweils manuell bewertete Relevanzurteile angeben zu können. Dabei wird auch auf das Konzept des *Pooling* zurückgegriffen: Da bei TREC mehrere Systeme miteinander verglichen werden, betrachtet man im Wesentlichen nur Dokumente, die von wenigstens einem System für relevant erachtet wurden. Damit grenzt sich die Anzahl der Dokumente, die einem manuellen Relevanzurteil unterzogen werden müssen, auf die Vereinigungsmenge aller von den verschiedenen IR-Systemen gelieferten Ergebnisse ein.

2.2.3 Mittelwertbildung über mehrere Anfragen

Mittelwertbildung

Ein Vergleich zwischen mehreren IR-Systemen muss sich auf eine repräsentative Menge von Suchanfragen abstützen, die jeweils auf allen zu vergleichenden Systemen durchgeführt werden. Hat man aber die Ergebnisse mehrerer Anfragen vorliegen, so stellt sich das Problem einer geeigneten Mittelwertbildung über mehrere Anfragen und/oder mehrere Dokumentenkollektionen hinweg zur Gewinnung gemittelter Recall- und Precision-Werte. Hierzu kann man zwei Ansätze wählen.

- Makrobewertung
- Mikrobewertung

Die Makrobewertung

Makrobewertung
Ansatz, nutzerorientierter

Den ersten Ansatz bildet die **Makrobewertung** , die auch als *nutzerorientierter Ansatz* bezeichnet wird.

Dieser Ansatz ermittelt die durchschnittlichen Recall- und Precision-Werte als wirkliche Durchschnittswerte, d.h. die Summe der Einzelwerte wird durch die Anzahl der Experimente geteilt. Wir greifen hierzu die Notation für die Bestimmung von Recall und Precision wieder auf (vgl. hierzu Abschnitt 2.2) und gehen davon aus, dass über m Experimente gemittelt werden soll:

$$Recall_{\varnothing u} = \frac{1}{m} \cdot \sum_{i=1}^{m} \frac{a_i}{a_i + c_i} \quad Precision_{\varnothing u} = \frac{1}{m} \cdot \sum_{i=1}^{m} \frac{a_i}{a_i + b_i} \qquad (2.3)$$

Wie wir sehen, gehen hier alle m Experimente unabhängig von der Ergebnisgröße mit dem gleichen Gewicht in die Durchschnittsberechnung ein.

Dieses Verfahren hat jedoch zwei Probleme:

- Anfragen, die leere Ergebnisse liefern, führen bei der Berechnung der Precision zu 0-Werten im Nenner, da $a_i + b_i = 0$ ist. Als Konsequenz könnte man Anfragen mit leerem Ergebnis außer Acht lassen. Dies würde jedoch die Aussagekraft des berechneten Mittelwertes verfälschen.

- Ein zweites Problem ergibt sich aus der Gleichgewichtung der einzelnen Anfragen innerhalb des Durchschnittsergebnisses. So werden Anfrageergebnisse mit beispielsweise 50 Ergebnisdokumenten, die eine viel höhere Aussagekraft über die Qualität eines Systems besitzen, gleichgewichtet mit Ergebnissen betrachtet, die beispielsweise nur ein einziges, dafür aber relevantes Dokument enthalten und so eine Precision von 100% liefern.

Aus diesen Gründen wendet man in der Praxis häufiger das im Folgenden betrachtete Maß der *Mikrobewertung* an.

Die Mikrobewertung

Der zweite Ansatz ist die **Mikrobewertung** , die auch als *systemorientierte Sichtweise* bezeichnet wird.

Mikrobewertung

Sichtweise, systemorientierte

Dieser Ansatz betrachtet die Summe der einzelnen Suchanfragen gewissermaßen als eine große Anfrage:

$$Recall_{\varnothing 2} = \frac{\sum_{i=1}^{m} a_i}{\sum_{i=1}^{m} (a_i + c_i)} \quad Precision_{\varnothing 2} = \frac{\sum_{i=1}^{m} a_i}{\sum_{i=1}^{m} (a_i + b_i)} \tag{2.4}$$

Anfragen mit einer größeren Ergebnismenge werden damit im Gesamtergebnis stärker gewichtet. Einzelne Anfragen, die als Ergebnis eine leere Menge liefern, stellen kein Problem mehr dar, da diese in das Ergebnis mit einbezogen werden können.

Ein Beispiel

Wir wollen nun anhand der folgenden vier Anfragen ($m = 4$) die beiden obigen Verfahren kurz vorführen:

Anfrage	$\lvert Erg_q \rvert = a + b$	$\lvert R_q \rvert = a + c$	$\lvert R_q \cap Erg_q \rvert = a$	$Recall$	$Precision$
1	10	7	1	0,14	0,1
2	12	13	2	0,15	0,17
3	200	138	120	0,87	0,6
4	4	9	2	0,22	0,5
Summe	226	167	125		

Tabelle 2.2 — Werte zu vier Beispielanfragen

Für die *Makrobewertung* erhalten wir einen Recall von $\frac{1}{4} \cdot (0,14 + 0,15 + 0,87 + 0,22) = 0,35$ und eine Precision von $\frac{1}{4} \cdot (0,1 + 0,17 + 0,6 + 0,5) = 0,34$.

Für die *Mikrobewertung* erhalten wir einen Recall von $\frac{125}{167} = 0,75$ und eine Precision von $\frac{125}{226} = 0{,}55$.

Dass die Werte bei der Mikrobewertung in diesem Fall deutlich besser ausfallen liegt daran, dass die relativ gut beantwortete Anfrage 3 durch die großen absoluten Zahlen die Mittelwertbildung nach der Mikrobewertung eindeutig dominiert, während die gute Anfrage 3 bei der Makrobewertung eben nur mit genau $\frac{1}{4}$ in die Recall- und Precision-Berechnung eingeht.

2.2.4 Recall/Precision-Werte bei Systemen mit Ranking

Bisher sind wir vereinfachend davon ausgegangen, dass ein zu untersuchendes IR-System eine unstrukturierte Menge als Ergebnis liefert, für die wir als Ganzes einen Recall- und einen Precision-Wert berechnen. Dabei konnten wir einfach unsere Formeln für Recall und Precision anwenden – auch wenn die Recall-Bestimmung in der Praxis, wie erwähnt, durchaus problematisch ist.

Ranking

Als wünschenswert für reale IR-Systeme hatten wir aber in der Einführung festgehalten, dass diese Systeme ein Ranking der Dokumente liefern sollten. Dies bedeutet, dass die Dokumente im Ergebnis nach ihrer Relevanz – bzw. nach der Relevanzeinschätzung durch das System – sortiert sind. In diesem Fall erscheint ein einziges Paar von Recall- und Precision-Werten nicht angemessen. Vielmehr sollte man eine Folge von Recall- und Precision-Werten betrachten, die dem sequentiellen Durcharbeiten des Rankings entspricht.

Beispiel

Nehmen wir als Beispiel eine Anfrage an, zu der die Menge R_q der relevanten Dokumente wie folgt gegeben ist: $R_q = d_3, d_5, d_9, d_{25}, d_{39}, d_{44}, d_{56}, d_{71}, d_{89}, d_{123}$. Die d_i stehen dabei für die Dokumente aus der Dokumentenkollektion.

Das zu evaluierende IR-System liefert nun das folgende Ranking von 15 Dokumenten, wobei die tatsächlich relevanten Dokumente durch ein »+« markiert sind:

1. d_{123} +

2. d_{84}
3. d_{56} +
4. d_6
5. d_8
6. d_9 +
7. d_{511}
8. d_{129}
9. d_{187}
10. d_{25} +
11. d_{38}
12. d_{48}
13. d_{250}
14. d_{113}
15. d_3 +

So ist etwa das Dokument d_{123}, das vom System als höchstrangig relevant eingestuft wurde, auch für den Benutzer wirklich relevant. Das vom System auf dem zweiten Rang als relevant eingestufte Dokument d_{84} ist jedoch für den Benutzer nicht relevant. Dokument d_{56} wurde vom System auf Rang drei eingestuft und ist auch für den Benutzer relevant.

Ein Weg zur Bestimmung der Recall- und Precision-Werte ist nun, nach jedem gefundenen Dokument im Ranking den Recall- und den Precision-Wert zu bestimmen. In unserem Beispiel würde dies zu den Werten in Tabelle 2.2.4 führen.

Abbildung 2.6 zeigt die Werte in einem Recall/Precision-Graph.

Problematisch ist bei dieser Vorgehensweise, dass es zu einem Recall-Wert mehrere Precision-Werte gibt. Will man mehrere Systeme an einem bestimmten Recall-Niveau von z.B. 20% hinsichtlich ihrer Precision miteinander vergleichen, so ergeben sich mehrere mögliche Precision-Werte.

Eine andere Variante betrachtet daher zu jedem erreichbaren Recall-Wert – in diesem Zusammenhang auch *Recall-Punkt* genannt – nur den

Recall-Punkt

betrachtete Dokumente bisher	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
relevante Dokumente bisher	1	1	2	2	2	3	3	3	3	4	4	4	4	4	5
Recall	0,1	0,1	0,2	0,2	0,2	0,3	0,3	0,3	0,3	0,4	0,4	0,4	0,4	0,4	0,5
Precision	1	0,5	0,67	0,5	0,4	0,5	0,43	0,38	0,33	0,4	0,36	0,33	0,31	0,29	0,33

Tabelle 2.3 — Recall- und Precision-Werte nach den einzelnen Dokumenten im Ergebnisranking

höchsten Precision-Wert. Dieses Verfahren entspricht einer Sichtweise, bei der man Recall- und Precision-Werte jeweils nur unmittelbar nachdem im Ranking ein relevantes Dokument vorgekommen ist ermittelt. In unserem Beispiel bedeutet dies nach dem ersten, dem dritten, dem sechsten, dem zehnten und dem fünfzehnten Dokument im Ranking.

Wenn wir nun $P(r)$ für die Precision in Abhängigkeit von einem bestimmten Recall-Wert schreiben, dann erhalten wir nach dem ersten Dokument $P(1/10) = 1/1 = 1,0$. Nach dem dritten Dokument erhalten wir $P(2/10) = 2/3 = 0,67$. Nach dem sechsten Dokument erhalten wir $P(3/10) = 3/6 = 0,5$. Nach dem zehnten Dokument erhalten wir $P(4/10) = 4/10 = 0,4$. Nach dem fünfzehnten Dokument erhalten wir $P(5/10) = 5/15 = 0,33$. Außerdem ergibt sich für die übrigen denkbaren Recall-Punkte 6/10, 7/10, 8/10, 9/10 und 10/10 ein Precision-Wert von näherungsweise $10/N$, wenn man davon ausgeht, dass das System diese Elemente mehr oder weniger zufällig aus den restlichen Dokumenten der Kollektion wählen müsste. Für große N liegen, $P(6/10)$, $P(7/10)$, ..., $P(10/10)$ damit nahe null.

Trägt man die resultierenden Recall- und Precision-Werte in einen Recall/Pecision-Graphen ein, ergibt dies ein Bild wie in Abbildung 2.7.

Man beachte, dass die Verbindungslinien in diesem Graphen keine Bedeutung haben und nur der Veranschaulichung dienen. Im Allgemeinen ergeben sich aus einer linearen Interpolation *keine* sinnvollen Precision-Werte zwischen den Recall-Punkten. Dies ist allerdings problematisch, wenn man zum Beispiel Systeme miteinander vergleichen will und dazu über mehrere Anfragen die Precision-Werte mitteln möchte. Man behilft

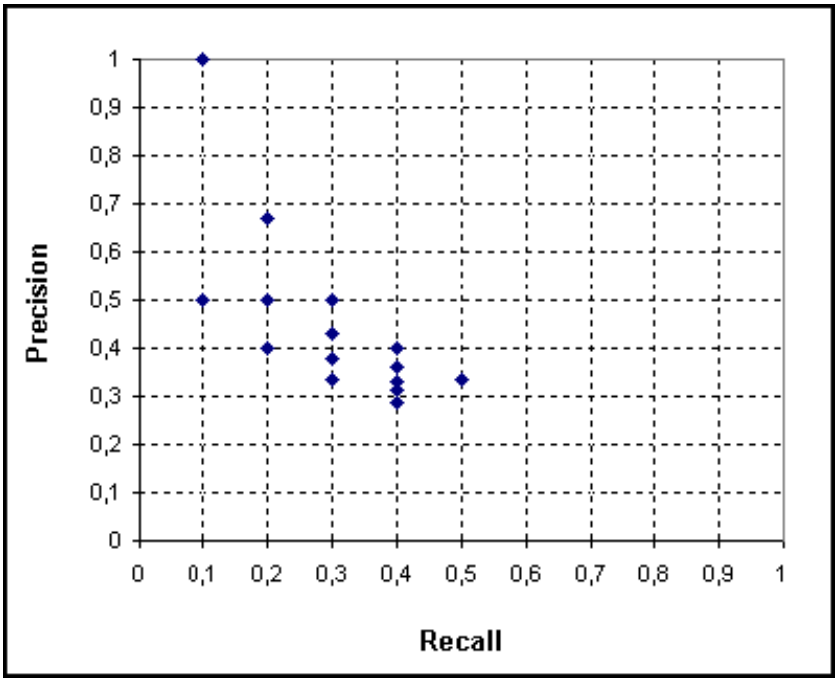


Abbildung 2.6 — Recall- und Precision-Werte nach den einzelnen Dokumenten im Ergebnisranking in einem Recall/Precision-Graph

sich üblicherweise mit dem Trick, dass man für Recall-Werte, die nicht explizit existieren, den Precision-Wert des nächsthöheren existierenden Recall-Wertes verwendet. Dazu rechnet man zunächst nach dem oben beschriebenen Muster für die Recall-Punkte $\frac{i}{|R_q|}$ die Precision-Werte aus $(i \in 1, \ldots, |R_q|)$ und definiert $P(r)$ dann zu $P(\frac{i}{|R_q|})$ mit $r \leq \frac{i}{|R_q|}$ und $r > \frac{(i-1)}{|R_q|}$. Für unser Beispiel ergibt dies den Graph wie in Abbildung 2.8 zu sehen.

2.2.5 Vergleich mehrerer Systeme, die ein Ranking liefern

Um mehrere IR-Systeme miteinander zu vergleichen, bestimmt man üblicherweise die Precision-Werte an vorgegebenen Recall-Punkten.

Abbildung 2.7 — Recall- und Precision-Werte nach den einzelnen relevanten Dokumenten im Ergebnisranking

Beispiele für vorgegebene Recall-Punkte sind:

0,1; 0,2; 0,3; 0,4; 0,5; 0,6; 0,7; 0,8; 0,9 ;(1,0) oder 0,25; 0,5; 0,75

Es ist dabei sinnvoll für jeden Recall-Punkt den Mittelwert über eine Anzahl von Anfragen und Dokumentenkollektionen zu bilden. Ob man aus den ermittelten Zahlen allerdings eine haltbare Aussage über die Vorteilhaftigkeit der Verfahren ableiten kann, muss in jedem Fall mit Testverfahren aus der Statistik nachgeprüft werden, auf die wir an dieser Stelle allerdings nicht eingehen wollen.

2.3 Experimente und Testkollektionen

Ein kritischer Punkt bei der Bestimmung von Recall/Precision-Werten ist die Auswahl der Dokumentenkollektion mit der die entsprechenden

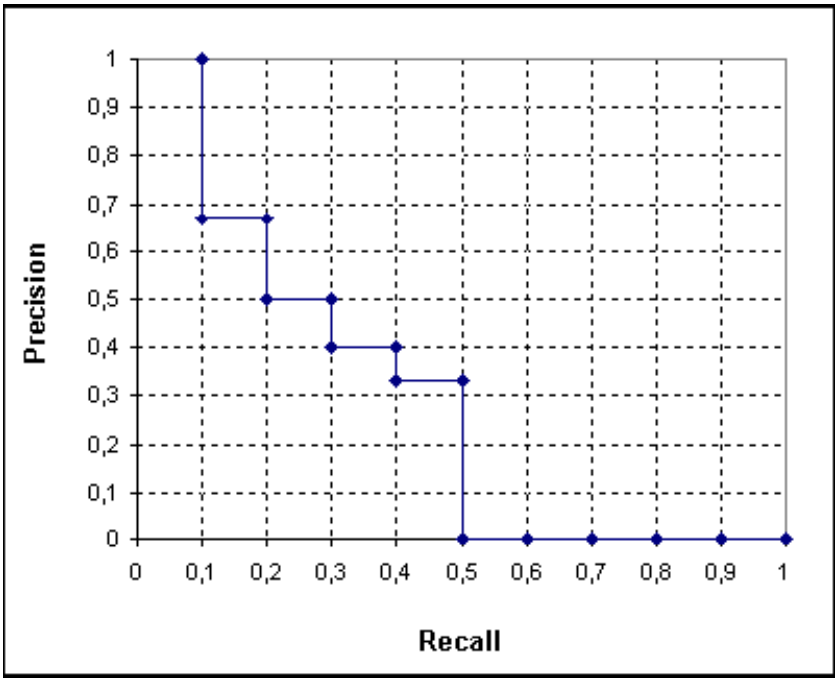


Abbildung 2.8 — Verwendung des Precision-Wertes zum nächsthöheren existierenden Recall-Wert

Experimente durchgeführt werden. Diese Kollektion sollte möglichst repräsentativ sein und darüber hinaus wäre wünschenswert, dass zu dieser Kollektion bereits Anfragen formuliert sind, zu denen Relevanzbeurteilungen vorliegen. Auf diese Weise können die Recall-Werte für die Testläufe mit wenig Aufwand bestimmt werden.

Der Bedarf nach allgemein anerkannten Testkollektionen für Experimente mit IR-Systemen hat zu einer Vielzahl von Testkollektionen geführt, deren prominentester Vertreter die so genannte *TREC-Kollektion* ist. *TREC-Kollektion*

2.3.1 Die TREC-Kollektion

TREC steht für *Text REtrieval Conference* und bezeichnet eine seit 1992 jährlich stattfindende Konferenzreihe. Ziel von TREC war und ist die

Schaffung von Testkollektionen, die vergleichbare Experimente mit einer umfangreichen Kollektion von Dokumenten ermöglichen (siehe auch *http://trec.nist.gov/*, letzter Abruf: 19.10.2006).

Tasks

Dazu wurde für jede TREC-Konferenz eine **Menge von Referenz-Experimenten** entwickelt. Diese Experimente werden als **Tasks** bezeichnet. Zwei Beispiele für solche Tasks sind:

- Zu gegebenen Anfragen soll mit beliebigem Aufwand ein Ranking der 1000 relevantesten Dokumente erstellt werden (**ad hoc task**).

ad hoc task

- Innerhalb von 5 Minuten sollen 15 Dokumente selektiert werden, die möglichst alle relevant sein sollten (**high precision task** – der Recall spielt hier also nur eine untergeordnete Rolle).

high precision task

Die Forschergruppen, die an der Konferenz teilnehmen wollen, müssen mit ihren Systemen die Referenz-Experimente durchführen. Die Teilnehmer können dabei beliebige Retrieval-Techniken anwenden.

Alle Verfahren werden auf die gleiche, mehrere Gigabyte umfassende Datenkollektion angewendet. Dabei sind vorgegebene Informationswünsche (**topics**) zu bearbeiten. Die Ergebnisse werden dann miteinander verglichen.

topics

An TREC nehmen neben zahlreichen Universitäten auch Unternehmen wie AT&T Labs Research, NEC Corp., Eurospider, Fujitsu Laboratories, Ltd. oder IBM – Almaden Research Center teil.

Woraus besteht die TREC-Kollektion?

Im Folgenden wollen wir am Beispiel der Kollektion für die siebte TREC-Konferenz (TREC-7) den Aufbau einer TREC-Kollektion verdeutlichen. Die Kollektion besteht aus:

- den Dokumenten im Umfang von insgesamt 5,8 Gigabyte,
- insgesamt 400 *topics* – jedes Jahr kommen 50 neu hinzu,
- zu jedem *topic* gibt es eine von Experten manuell ausgewählte Menge von relevanten Dokumenten.

Die Dokumentenkollektion kann jeweils beim NIST (National Institute of Standards and Technology in Maryland, USA) gegen eine Gebühr auf CD bestellt werden. Die *topics* und die Relevanzurteile stehen auf der WWW-Seite *http://trec.nist.gov/* (letzter Abruf: 19.10.2006) bereit.

Die Dokumente selbst sind typischerweise Artikel aus Zeitungen und wissenschaftlichen Publikationen. Sie liegen in einem XML-Format vor, bei dem über die verschiedenen Quellen hinweg die gleichen übergeordneten Strukturen verwendet werden.

Tabelle 2.4 gibt die Zusammensetzung der TREC7-Kollektion wieder (nach Voorhees und Harman[VH98]). Die Tabelle gibt die Quelle der Dokumente, die Größe der Teilkollektion in Megabyte, die Anzahl der Dokumente sowie den Median und den Mittelwert über die Anzahl der Wörter pro Dokument an.

Ein Beispieldokument aus der Financial Times ist folgendermaßen strukturiert:

```
<DOC>
   <DOCNO>FT911-3</DOCNO>
   <PROFILE>AN-BEOA7AAIFT</PROFILE>
   <DATE>910514</DATE>
   <HEADLINE>
      FT 14 MAY 91 /International Company News:Contigas
          plans DM900m east German project
   </HEADLINE>
   <BYLINE>By DAVID GOODHART </BYLINE>
   <DATELINE>BONN </DATELINE>
   <TEXT>
      CONTIGAS, the German gas group 81 per cent owned by
          the utility Bayernwerk, said yesterday that it
          intends to invest DM900m (Dollars 522m) in the
          next four years to build a new gas distribution
          system in the east German state of Thuringia. ...
   </TEXT>
</DOC>
```

Die Anfragen (topics)

Die Anfragen (topics) bestehen bei den neueren TREC-Konferenzen aus folgenden Bestandteilen: *topics*

- Der Titel (**title**) ist wenige Worte lang und versucht möglichst gut den Gegenstand der Anfrage zu beschreiben.

- Die Beschreibung (**description**) gibt die ursprüngliche Benutzeranfrage in der Regel in einem Satz wieder.

- Der **narrative** genannte Teil steht für einen Langtext, der versucht, genauer zu beschreiben, welche Dokumente sich für die gegebene Anfrage qualifizieren sollten und welche nicht.

Durch Verwendung der unterschiedlichen Teile der Topic-Beschreibung kann der Effekt unterschiedlicher Anfragelängen auf die Performance untersucht werden. Beispielsweise ähnelt der Titel in seiner Kürze eher den Anfragen von Nutzern einer Suchmaschine im Internet. Im Gegensatz dazu muss bei Verwendung des *narrative* ein System die Inhalte linguistisch verstehen, da im *narrative* häufig auch beschrieben wird, welche Inhalte oder Bestandteile ein Dokument ausdrücklich irrelevant machen.

Von Konferenz zu Konferenz gibt es eine rege Diskussion über Inhalt und Länge der einzelnen Bestandteile der Topics. So sollen beispielsweise die ein bis drei Worte langen Titel bei TREC-7 Anfragen im WWW annähern.

Das folgende Beispiel für ein Topic aus TREC-7 macht die Aufteilung in die einzelnen Bereiche deutlich.

```
<top>

<num>Number: 352

<title>British Chunnel impact

<desc>Description:

What impact has the Chunnel had on the British economy and/
    or the life style of the British?

<narr>Narrative:

Documents discussing the following issues are relevant:
   - projected and actual impact on the life styles of the
       British
   - Long term changes to economic policy and relations
   - major changes to other transportation systems linked
       with the Continent

Documents discussing the following issues are not relevant:
   - expense and construction schedule
```

```
    - routine marketing ploys by other channel crossers (i.e
        ., schedule changes, price drops, etc.)

</top>
```

Ermittlung der relevanten Dokumente

Um die Bestimmung des Recalls für die am Vergleich teilnehmenden Systeme zu erlauben, muss zu jedem Topic die Menge der relevanten Dokumente bekannt sein. Bei TREC werden diese Dokumente aus einem *Pool* von möglicherweise relevanten Dokumenten ermittelt. Der *Pool* wird dadurch erzeugt, dass man über alle Einreichungen zu diesem Topic die Vereinigung über deren K relevanteste Dokumente bildet, typischerweise ist $K = 100$.

Pool

Die Dokumente aus dem *Pool* werden dann menschlichen Experten zur Beurteilung vorgelegt.

Diese *Pooling Method* basiert offensichtlich auf der Annahme, dass (fast) alle relevanten Dokumente im *Pool* enthalten sein werden, weil jedes relevante Dokument von mindestens einem System gefunden wird. Da diese Annahme nicht zutreffen muss, sind die Recall-Werte bei TREC-Messungen tendenziell eher zu gut.

Pooling Method

TREC Tracks

Wie bereits erwähnt werden bei jeder TREC-Konferenz verschiedene Aufgabenstellungen in so genannten Tracks betrachtet. Die Entwicklung dieser Aufgabenstellungen über die Jahre hinweg gibt auch einen recht interessanten Überblick über die aktuellen Forschungsrichtungen im Information Retrieval.

Die folgenden kurzen Beschreibungen der Tracks stammen von den offiziellen Web-Seiten zu TREC *http://trec.nist.gov/* (letzter Abruf: 19.10.2006).

TREC Tracks im Jahr 2005

- *Enterprise Track*

A new track in TREC 2005. The purpose of the enterprise track is to study enterprise search: satisfying a user who is searching the data of an organization to complete some task.

- *Genomics Track*

  The purpose of the track is to study retrieval tasks in a specific domain, where the domain of interest is genomics data (broadly construed to include not just gene sequences but also supporting documentation such as research papers, lab reports, etc.) The genomics track first ran in TREC 2003 and will run again in TREC 2005.

- *HARD Track*

  The goal of HARD is to achieve High Accuracy Retrieval from Documents by leveraging additional information about the searcher and/or the search context, through techniques such as passage retrieval and using very targeted interaction with the searcher. The hard track first ran in TREC 2003 and will run again in TREC 2005.

- *Question Answering Track*

  A track designed to take a step closer to information retrieval rather than document retrieval.

  [Dabei geht es darum konkrete Fragen zu beantworten wie: »Welches Unternehmen kaufte in den vergangenen Jahren nennenswerte Mengen von Volkswagen-Aktien?«]

- *Robust Retrieval Track*

  A track that includes a traditional ad hoc retrieval task task, but with the focus on individual topic effectiveness rather than average effectiveness. The robust retrieval track first ran in TREC 2003 and will run again in TREC 2005.

- *SPAM Track*

  A new track in TREC 2005. The goal of the SPAM track is to provide a standard evaluation of current and proposed spam filtering approaches, thereby laying the foundation for the evaluation of more general email filtering and retrieval tasks.

- *Terabyte Track*

This track first ran in TREC 2004 and will run again in TREC 2005. The purpose of the terabyte track is to investigate whether/how the IR community can scale traditional IR test-collection-based evaluation to significantly larger document collections than those currently used in TREC. The retrieval task will be an ad hoc task using a static collection of approximately 1 terabyte of spidered web pages (probably from the .GOV domain). The terabyte track first ran in TREC 2004 and will run again in TREC 2005.

Weitere TREC Tracks vorhergehender Jahre

- *Cross-Language Track*

  A track that investigates the ability of retrieval systems to find documents that pertain to a topic regardless of the language in which the document is written.

- *Filtering Track*

  A task in which the user's information need is stable (and some relevant documents are known) but there is a stream of new documents. For each document, the system must make a binary decision as to whether the document should be retrieved (as opposed to forming a ranked list).

- *Interactive Track*

  A track studying user interaction with text retrieval systems. Participating groups develop a consensus experimental protocol and carry out studies with real users using a common collection and set of user queries. The interactive track was run as an adjunct to the Web Track in TREC 2003, and is not slated to run in TREC 2004.

- *Novelty Track*

  A track to investigate systems' abilities to locate new (i.e., non-redundant) information. This track ran in TREC 2004.

- *Video Track*

  TREC 2001 and 2002 contained a video track devoted to research in automatic segmentation, indexing, and content-based retrieval of digital video. Beginning in 2003, the track became an independent evaluation (TRECVID).

- *Web Track*

  A track featuring search tasks on a document set that is a snapshot of the World Wide Web.

2.3.2 Die CACM- und CISI-Kollektionen

Neben der TREC-Kollektion gibt es noch weitere Testkollektionen, die alle deutlich kleiner sind. Eine Auswahl von weiteren Testkollektionen ist:

ADI-Kollektion
- ADI (82 Dokumente aus dem Bibliothekswesen),

CACM-Kollektion
- CACM (3204 Artikel aus den Communications of the ACM, einer Informatik-Fachzeitschrift),

INSPEC-Kollektion
- INSPEC (12684 abstracts on electronics, computer, and physics),

CISI-Kollektion
- CISI (1460 Dokumente aus dem Bibliothekswesen) und

Medlars-Kollektion
- Medlars (medizinische Artikel).

Von diesen Kollektionen wollen wir die CACM und die CISI etwas genauer betrachten[1]. Ferner sei an dieser Stelle auch auf weitere Evaluierungsinitiativen für spezielle Dokumenttypen hingewiesen. So beschäftigt sich die *INitiative for the Evaluation of XML Retrieval (INEX)*[2] mit der Suche in und nach XML-Dokumenten und das *Cross-Language Evaluation Forum (CLEF)*[3] betrachtet Situationen, in denen entweder die Anfrage und die Dokumente in unterschiedlichen Sprachen vorliegen oder Dokumente in verschiedenen Sprachen gleichzeitig betrachtet werden sollen.

Die CACM-Kollektion

CACM-Kollektion

Bei der CACM-Kollektion handelt sich um 3204 Artikel, die in den *Communications of the ACM* zwischen 1958 und 1979 veröffentlicht wurden. Damit wird ein beträchtlicher Teil der Informatik-Literatur thematisch

[1] Informationen zu diesen Kollektionen finden Sie im Web unter http://www.dcs.gla.ac.uk/idom/ir_resources/test_collections/, letzter Abruf: 06.11.2006

[2] http://inex.is.informatik.uni-duisburg.de/, letzter Abruf: 06.11.2006

[3] http://www.clef-campaign.org/, letzter Abruf: 06.11.2006

abgedeckt, da die CACM eine der Top-Zeitschriften in der Informatik ist.

Neben den Dokumenten selbst enthalten die Datensätze noch weitere Informationen:

- die Namen der Autoren,
- das Datum des Erscheinens,
- Wortstämme aus der Überschrift und den Abstracts,
- Kategorien, die aus der *ACM Computing Reviews Klassifikation* abgeleitet sind,
- direkte Querverweise zwischen Artikeln (Paare mit der Bedeutung a referenziert b),
- bibliographische gemeinsame Beziehungen (Tripel (a,b,c) mit der Bedeutung a und b enthalten eine Referenz auf c) und
- die Anzahl gemeinsamer Zitate für jedes Paar von Dokumenten.

Damit eignet sich diese Kollektion insbesondere zum Test von IR-Verfahren, die Zitate mit betrachten. Schon frühzeitig kamen Verfahren auf, die Zitationen für die Bewertung der Relevanz von Dokumenten verwendeten. Analoge Verfahren finden sich heute auch in Internetsuchmaschinen wie Google; wir werden darauf an entsprechender Stelle im Kurs noch genauer eingehen.

Die CACM-Kollektion umfasst auch 52 Anfragen. Ein Beispiel für eine derartige Anfrage ist: *What articles exist which deal with TSS (Time Sharing System), an operating system for IBM computers?*

Zu jeder Anfrage existieren ferner 2 boolesche Anfrageformulierungen und die Menge der relevanten Dokumente. Da die Anfragen sehr spezifisch sind, gibt es im Durchschnitt ca. 15 relevante Dokumente.

Die CISI-Kollektion

CISI-Kollektion

Die CISI-Kollektion enthält 1460 Dokumente und wurde aus einer Vorläuferkollektion abgeleitet, die am *Institute of Scientific Information* (CISI) entwickelt wurde. Die Kollektion kann hauptsächlich zur Evaluierung

von Verfahren genutzt werden, die auf Ähnlichkeiten und Kreuzzitaten aufbauen.

Auch in dieser Kollektion werden zusätzliche Informationen bereitgestellt:

- die Namen der Autoren,
- Wortstämme aus der Überschrift und den Abstracts sowie
- die Anzahl gemeinsamer Zitate für jedes Paar von Dokumenten.

Es gibt 35 Beispielanfragen in natürlicher Sprache, zu denen auch boolesche Anfrageformulierungen vorliegen. Ferner gibt es weitere 41 Anfragen, die nur in natürlicher Sprache vorliegen. Zu jeder Anfrage gibt es im Durchschnitt 50 relevante Dokumente.

Der Umstand, dass viele der relevanten Dokumente keine Begriffe mit der Anfrage gemein haben, macht den Einsatz von fortgeschrittenen Suchtechnologien nötig, die Wortstämme oder Zitationen benutzen.

2.4 Kriterienkataloge zur Evaluierung von IR-Systemen

Kriterienkataloge

Obwohl die Abweichung des vom System ermittelten Ergebnisses von R_q ein wesentliches Kriterium für die Qualität eines IR-Systems ist, müssen bei der Bewertung von IR-Systemen neben Recall und Precision auch Kriterien wie die Effektivität und die Benutzbarkeit berücksichtigt werden. Hierzu wollen wir zwei Vorschläge für entsprechende Kriterienkataloge betrachten:

Sechs Kriterien nach Cleverdon, Mills und Keen

Cleverdon, Mills und Keen[CMK66] definieren für die Evaluierung von IR-Systemen die folgenden sechs Kriterien:

Recall

- **Recall** bezeichnet die Fähigkeit des Systems, alle relevanten Objekte zu finden.

- **Precision** beschreibt die Fähigkeit des Systems, nur die Objekte zur präsentieren, die relevant sind. *Precision*
- **Time Lag** betrachtet die durchschnittliche Zeit, die ein System zwischen dem Stellen einer Anfrage und der Präsentation der Antwort benötigt. *Time Lag*
- **Effort** adressiert den intellektuellen und physischen Aufwand, der von einem Benutzer zur Erlangung der Antwort erforderlich ist. Dieser Punkt berücksichtigt u.a. die Kompliziertheit der Eingabe einer Anfrage, beispielsweise als Boolescher Ausdruck oder als natürlich-sprachliche Eingabe, oder auch, ob ein System in Zweifelsfällen Rückfragen an den Benutzer stellt. Auch der physische Aufwand – etwa zum Suchen nach Büchern in einer Bibliothek – fällt unter diesen Punkt. *Effort*
- **Form of Presentation** drückt die Qualität der Ergebnisdarstellung aus, die zudem die Nützlichkeit für den Benutzer beeinflusst. *Form of Presentation*
- **Coverage of the Collection** gibt den Umfang bzw. den Grad an, zu dem das System relevante Dokumente enthält. Dieser Punkt betrifft jedoch mehr die Dokumentenkollektion selbst als das auf dieser Kollektion aufsetzende Retrieval System. *Coverage of the Collection*

Kriterien nach Vickery

Vickery[Vic70] orientiert sich mit seinem Kriterienkatalog an Cleverdon, Mills und Keen. Er unterscheidet dabei aber nach Kriterien, die sich auf die Verfügbarkeit von Informationen beziehen und Kriterien, die sich auf die Selektivität der Antwort beziehen.

Zunächst also zu den Kriterien, die sich auf die **Verfügbarkeit von Informationen** beziehen:

- **Coverage** umfasst den Anteil der möglicherweise nützlichen Literatur, der bei der Bearbeitung der Anfrage in Betracht gezogen wurde. Wenn man z.B. davon ausgeht, dass ein IR-System potentiell Zugang zu mehreren Dokumentenkollektionen hat, dann stellt sich hier die Frage, ob für eine konkrete Anfrage auch wirklich alle Kollektionen mit möglicherweise relevanten Dokumenten betrachtet wurden. Noch plastischer wird dieses Kriterium, wenn man an eine manuell durchgeführte Recherche denkt. Hier stellt sich die *Coverage*

Frage, ob alle potentiell nützlichen Quellen/Dokumentenkollektionen auch in die Recherche einbezogen wurden.

Recall

- **Recall** beschreibt den Anteil der möglicherweise nützlichen Dokumente, die im Ergebnis der Anfrage enthalten sind und

Response

- **Response** adressiert die durchschnittliche Antwortzeit eines Information Retrieval Systems.

Neben diese Kriterien, die der Frage nachgehen, wie vollständig und wie schnell verfügbar die Antwort ist, stellt Vickery Kriterien zur Beurteilung von Information Retrieval Systemen, die sich auf die **Selektivität der Antwort** beziehen:

Precision

- **Precision** bezieht sich auf die Fähigkeit eines Systems, unnütze Dokumente im Ergebnis zu unterdrücken.

Usability

- **Usability** betrachtet den Wert des Ergebnisses für den Benutzer. Das Ergebnis solle dabei zuverlässig und verständlich sein. Dies bezieht sich hier insbesondere auf die Dokumente im Ergebnis, die aus verlässlichen Quellen stammen sollten und die von der Verständlichkeit zur Anfrage des Benutzers passen sollten.

Presentation

- **Presentation** trifft schließlich eine Aussage über die Form, in der die Ergebnisse dem Benutzer von einem IR-System präsentiert werden. Auch hier sind Klarheit und Verständlichkeit wünschenswert.

Ergebnisqualität
Effizienz
Benutzbarkeit

Die angeführten Kriterien adressieren grob die drei Bereiche *Ergebnisqualität* , *Effizienz* und *Benutzbarkeit* . Ihre Überprüfung für ein konkretes System ist dabei recht komplex. Während Messungen zur Effizienz noch recht einfach in Form von Zeitmessungen möglich sind, sind Fragen der Benutzbarkeit nur durch fundierte Studien mit mehreren Benutzern zu adressieren.

2.5 Zusammenfassung

In diesem Abschnitt haben wir insbesondere die Evaluierungsmaße Precision und Recall kennengelernt. Wir sind dabei auch auf die Möglichkeiten zur effizienten Bestimmung des Recall eingegangen.

Im Falle von IR-Systemen, die ein Ranking als Ergebnis liefern, muss der Verlauf der Recall- und Precision-Werte bei der Betrachtung des Rankings in einem Recall/Precision-Graph dargestellt werden. Hierfür haben wir verschiedene Möglichkeiten vorgestellt.

Um Recall/Precision-Betrachtungen auf eine allgemein anerkannte Basis zu stellen wurden zahlreiche Testkollektionen vorgestellt. Wir haben hierzu die TREC-, die CACM- und die CISI-Kollektion kurz vorgestellt. Wesentlich ist dabei, dass zu den Kollektionen auch Anfragen existieren, zu denen die relevanten Dokumente bekannt sind.

Schließlich haben wir neben Recall und Precision noch weitere Qualitätskriterien für IR-Systeme angesprochen, die im Wesentlichen die Effizienz und die Benutzbarkeit der Systeme adressieren.

Anzumerken ist dabei auch, dass die einseitige Konzentration der Evaluierungsbemühungen auf Recall und Precision seit einigen Jahren zunehmend kritisiert wird. So haben Hersh und seine Coautoren [HTP+00] in ihren Arbeiten keinen Zusammenhang zwischen den bei einem Systemvergleich mit Recall und Precision erzielten Ergebnissen und den Ergebnissen einer Nutzerbefragung festgestellt. Bei den Untersuchungen wurden die bei TREC-8 erzielten Ergebnisse hinsichtlich der Vorteilhaftigkeit einzelner Systeme mit den von Bibliothekaren nach einer systematischen Bearbeitung der gleichen Aufgaben mit den gleichen Systemen in Fragebögen abgegebenen Einschätzungen verglichen. Das Ranking der Systeme nach TREC und die Einschätzung durch die Bibliothekare ergaben dabei unterschiedliche Ergebnisse. Ähnliche Ergebnisse werden auch aus anderen Untersuchungen berichtet [ACL05].

Daraus ergibt sich für die Zukunft natürlich die Frage, wie IR-Systeme zu evaluieren sind. Da umfangreiche und aufwändige Nutzerstudien nicht in allen Fällen möglich sind, werden Betrachtungen von Recall und Precision sicher noch lange üblich bleiben. Das Vertrauen in diese Maße ist aber doch etwas gesunken.

	Size (megabytes)	# Docs	Medien # Words/Doc	Mean # Words/Doc
Disk 1				
Wall Street Journal, 1987-1989	267	98732	245	434,0
Associated Press newswire, 1989	254	84678	446	473,9
Computer Selects articles, Ziff-Davis	242	75180	200	473,0
Federal Register, 1989	260	25960	392	1315,9
abstracts of U.S. DOE publications	184	226087	111	120,4
Disk 2				
Wall Street Journal, 1990-1992 (WSJ)	242	74520	301	508,4
Associated Press newswire, 1988 (AP)	237	79919	438	468,7
Computer Selects articles, Ziff-Davis (ZIFF)	175	56920	182	451,9
Federal Register, 1988 (FR88)	209	19860	396	1378,1
Disk 3				
San Jose Mercury News, 1991	287	90257	379	453,0
Associated Press newswire, 1990	237	78321	451	478,4
Computer Selects articles, Ziff-Davis	345	161021	122	295,4
U.S. patents, 1993	243	6711	4445	5391,0
Disk 4				
the Financial Times, 1991-1994 (FT)	564	210158	316	412,7
Federal Register, 1994 (FR94)	395	55630	588	644,7
Congressional Record, 1993 (CR)	235	27922	288	1373,5
Disk 5				
Foreign Broadcast Information Service (FBIS)	470	130471	322	543,6
the LA Times	475	131896	351	526,5

Tabelle 2.4 — Zusammensetzung der TREC7-Kollektion

3 Berücksichtigung der Vagheit in Sprache

Methoden zum Umgang mit Textdokumenten in natürlicher Sprache

Gegenstand des Information Retrieval ist (wie wir im Abschnitt 1 gesehen haben) die Suche nach Dokumenten. Dabei stehen Textdokumente nach wie vor klar im Vordergrund. Ziel des Information Retrieval ist es nun, die Semantik – also den Inhalt – der Dokumente zu adressieren. Dabei sollte von der konkreten Begriffswahl und Formulierung abstrahiert werden. Hierzu sind im Laufe der Jahre zahlreiche Techniken entwickelt worden. Als wesentliche Bereiche sind die *Stoppworteliminierung*, die *Grund- und Stammformreduktion*, die *Mehrwortgruppenbehandlung* und die *Betrachtung von Begriffsbeziehungen* zu nennen. Diese vier Bereiche wollen wir in diesem Kapitel nach einem einführenden Beispiel betrachten.

Es erscheint dabei wichtig zunächst das Begriffsverständnis bezüglich der Begriffe »Wort«, »Begriff«, »Term« und »Konzept« zu klären:

- **Wort** werden wir im Folgenden meist mit einem starken Bezug zur Zeichenkette verwenden. Der Begriff *Wort* bezieht sich damit eher auf die syntaktische Ebene. *Wort*

- Um stärker die Bedeutung zu adressieren wird in der Regel von einem **Begriff** – oder aus dem Englischen kommend von einem **Term** – gesprochen. *Begriff* *Term*

- Um einen noch stärkeren Bezug zur Semantik auszudrücken spricht man schließlich von einem **Konzept**. Damit wird in der Regel eine Gruppe von Wörtern betrachtet, die sehr inhaltsverwandt sind und hinter denen ein inhaltliches *Konzept* steht. Beispiele für Konzepte könnten »Datenbank«, »Geldinstitut« oder »Bewegung« sein. *Konzept*

Dieses Kapitel beinhaltet folgende Unterkapitel:

3.1 Ein einführendes Beispiel
3.2 Stoppworteliminierung
3.3 Stamm- und/oder Grundformreduktion
3.4 Mehrwortgruppenidentifikation
3.5 Terminologische Kontrolle
3.6 Ein sprachunabhängiger Ansatz
3.7 Auszeichnung mittels Metadaten
3.8 Zusammenfassung

3.1 Ein einführendes Beispiel

Repräsentation

Grundlage des Information Retrieval ist eine geeignete Repräsentation der Dokumente und Anfragen. Ein einfacher Ansatz hierzu ist bspw. die Repräsentation der Dokumente und Anfragen durch die Menge der in ihnen vorkommenden Wörter.

Betrachten wir dazu den folgenden kurzen Text:

> »Wirtschaft und Gesellschaft befinden sich derzeit im größten Umbruch seit der Industrialisierung. Die Ursache hierfür liegt in der globalen Verfügbarkeit leistungsfähiger und zugleich kostengünstiger Informations- und Kommunikationstechnologien. Das Informationszeitalter wird Realität.«

Die Menge der in diesem kurzen Text enthaltenen Wörter, die im Folgenden angegeben ist, wäre nun ein erster Kandidat für die Repräsentation dieses Textes (die Wörter sind hier alphabetisch geordnet):

> befinden, Das, der, derzeit, Die, Gesellschaft, globalen, größten, hierfür, im, in, Industrialisierung, Informations, Informationszeitalter, Kommunikationstechnologien, kostengünstiger, leistungsfähiger, liegt, Realität, seit, sich, Umbruch, und, Ursache, Verfügbarkeit, wird, Wirtschaft, zugleich

Eine genauere Betrachtung dieser Repräsentation führt uns allerdings zu einer Reihe von Problemen:

- Zum einen enthält die Menge Wörter, die für sich betrachtet keinen Sinn tragen. Dies sind bspw. Wörter wie *und*, *im*, *der* oder *wird*. Diese Wörter werden im Information Retrieval als Stoppworte bezeichnet. Sie sollten mittels einer **Stoppworteliminierung** aus der zur Repräsentation eines Dokumentes verwendeten Menge von Wörtern ausgeschlossen werden. Dies reduziert zum einen den Speicherplatzbedarf und es führt andererseits dazu, dass die für ein Dokument wirklich charakteristischen Begriffe stärker hervortreten.

  Stoppwort

- Des Weiteren finden sich im obigen Beispiel zusammengesetzte Wörter, so genannte Mehrwortgruppen. Würden wir im Beispiel nach den einzelnen Begriffen *Kommunikation* und *Technologie* suchen, so würden wir das Beispieldokument nicht finden. Dies bedeutet, dass wir Techniken der **Mehrwortgruppenidentifizierung** anwenden sollten, um den Recall bei solchen Anfragen zu erhöhen.

  Mehrwortgruppe

- Ein weiteres Problem stellt sich, wenn wir nach Dokumenten suchen, die den Begriff *kostengünstig* enthalten. Auch hier würden wir das oben aufgeführte Beispieldokument nicht finden, da dort nur das Wort *kostengünstiger* enthalten ist. Um dieses Problem zu adressieren, können Techniken zur **Stammform- oder Grundformreduzierung** angewendet werden.

  Grund- und Stammformreduzierung

- Schließlich würden wir den obigen Text auch nicht finden, wenn wir nach dem Wort *Grund* suchen würden, weil im Text lediglich das Wort *Ursache* vorkommt, das aber zumindest sinnverwandt mit *Grund* ist. Einen Ansatz zur Lösung dieses Problems bilden **Thesauri**, in denen inhaltliche Beziehungen zwischen Begriffen verwaltet werden. Dies erlaubt zum Beispiel bei einer Anfrage auch alle **Synonyme** der Anfragebegriffe zu betrachten.

  Thesaurus

  Synonym

Die angesprochenen vier Bereiche sollen nun in den Folgeabschnitten genauer betrachtet werden.

3.2 Stoppworteliminierung

Das Ziel der Stoppworteliminierung ist es, Terme, die nicht zur Semantik der Dokumente beitragen, nicht zu verwalten. Dabei ist zu beachten,

Stoppworteliminierung

dass »nicht zur Semantik beitragen« in diesem Fall nicht bedeutet, dass die Terme im Satzgefüge nicht zur Bedeutung des Satzes und damit des Textes beitragen. Vielmehr ist hier die isolierte Betrachtung der Wörter im Sinne einer unstrukturierten Wortmenge gemeint, und in diesem Zusammenhang trägt dann z.B. das isoliert betrachtete Wort *und* keine für die Anfragebearbeitung nutzbare Semantik.

Durch eine Stoppworteliminierung werden mehrere Effekte erreicht:

- Zunächst erreicht man eine **Reduzierung des Speicherplatzbedarfs** für die Repräsentation der Dokumente, da häufig vorkommende Wörter eliminiert werden. In der Praxis können sich hieraus Speicherplatzeinsparungen von 30 bis 50% ergeben.

Matchingalgorithmus

- Durch die Reduktion des Speicherplatzbedarfs wird vielfach auch die **Performance der Matchingalgorithmen** verbessert, weil diese durch die Stoppworteliminierung mit geringeren Datenmengen arbeiten müssen.

Recall
Precision
Vektorraummodell

- Schließlich kann die Eliminierung von Stoppwörtern auch zur **Verbesserung von Recall und Precision** beitragen. Wenn wir z.B. an eine Anfragebearbeitung mit dem Vektorraummodell denken, dann würden die Komponenten in den Beschreibungsvektoren, die mit Begriffen wie *und* oder *der* assoziiert sind, bei fast allen Dokumenten hohe Werte haben, weil diese Begriffe recht häufig vorkommen. Dadurch würde aber der Einfluss der in den Dokumenten vorkommenden charakteristischen Fachbegriffe auf das als

Ähnlichkeitsmaß

Ähnlichkeitsmaß verwendete Skalarprodukt abgeschwächt.

Zur Realisierung einer Stoppworteliminierung gibt es nun verschiedene Ansätze: das manuelle Führen einer Stoppwortliste sowie die automatische Entfernung hochfrequenter Begriffe. Beide Ansätze werden wir im Weiteren vorstellen.

Führen einer Stoppwortliste

Der erste Ansatz besteht darin, dass man für eine konkrete Sprache (z.B. Englisch oder Deutsch) manuell eine Liste der Stoppworte erstellt. Eine Liste für Englisch könnte z.B. folgende Begriffe enthalten: *able, about, above, according, accordingly, across, actually, after, afterwards, again,*

against, ain't, all, allow, allows, almost, alone, along, already, also, although, always, am, among, amongst, an, and, another, any, anybody, anyhow, ...

Technisch wird man eine solche Stoppwortliste typischerweise in einer Datei ablegen, die beim Start des Information Retrieval Systems in eine geeignete Datenstruktur geladen wird (z.B. eine Hashtabelle oder einen AVL-Baum). Dies bietet eine effiziente Möglichkeit der Anfrage, ob einzelne Wörter ein Stoppwort darstellen und somit aus der Repräsentation von Dokumenten und Anfragen zu entfernen sind.

Als Nachteil manuell erstellter Stoppwortlisten erscheint ihr Erstellungs- und Pflegeaufwand. Andererseits gibt es aber für alle gängigen Sprachen umfassende Stoppwortlisten und die manuelle Editierbarkeit ermöglicht eine individuelle Anpassung an konkrete Anwendungen, in denen einzelne Worte vielleicht ausnahmsweise keine Stoppworte sind oder noch zusätzliche Stoppworte aufgenommen werden sollten. So wird z.B. in einer Dokumentenkollektion mit Artikeln einer Informatik-Fachzeitschrift der Begriff *Rechner* ein sinnvolles Stoppwort sein.

Referenzen zu einigen Beispielen für Stoppwortlisten in verschiedenen Sprachen finden sich unter *http://www.unine.ch/info/clef/*.

Eliminierung hoch- und niedrigfrequenter Begriffe

Eine Alternative zum manuellen Führen einer Stoppwortliste stellt die Eliminierung aller hochfrequenten Begriffe dar. Dies sind Wörter, die z.B. in mehr als 20% aller Dokumente vorkommen, wobei der Prozentsatz natürlich unterschiedlich gewählt werden kann.

In einem Artikel zu diesem Thema schlägt Crouch zusätzlich vor, auch selten vorkommende Begriffe zu eliminieren. Konkret empfiehlt Crouch [Cro90]:

- Begriffe, die in weniger als 1% aller Dokumente auftreten, nicht zu betrachten, weil sie **zu spezifisch** sind, und
- Begriffe, die in mehr als 10% der Dokumente auftreten, nicht zu betrachten, weil sie **zu allgemein** sind.

Zu beachten ist hierbei, dass solche Ansätze nur angewendet werden können, wenn ein konkreter Datenbestand existiert, der eine repräsentative Ermittlung der Vorkommenshäufigkeiten erlaubt.

Bei der Anwendung der von Crouch vorgeschlagenen Grenzen darf darüber hinaus der Einfluss der Größe der Dokumente nicht unterschätzt werden.

Ein Beispiel

In einem Beispiel wurden die Kapitel und Verse des Neuen Testaments der Bibel betrachtet. Berücksichtigt man nur Substantive, so enthält jedes Dokument im Durchschnitt:

- 4,09 Begriffe, wenn man die Verse als Dokumente betrachtet, bzw.
- 76,34 Begriffe, wenn man die Kapitel als Dokumente betrachtet.

Abbildung 3.1 visualisiert das Verhältnis zwischen der relativen Anzahl der Begriffe, die in mindestens einem bestimmten Prozentsatz aller Dokumente vorkommen und der relativen Anzahl der Dokumente. Die Bereiche links und rechts des grauen Bereichs repräsentieren die Begriffe, die nach Crouch als nieder- bzw. hochfrequent einzustufen sind.

Damit erscheinen die von Crouch angegebenen Grenzen relativ sinnvoll, wenn man die Kapitel als Dokumente betrachtet. 18% der Begriffe würden demnach als zu speziell und 8% als zu allgemein eingestuft.

Betrachtet man dagegen die Verse als Dokumente würden 96% der Begriffe als zu speziell eingestuft und demnach als Stoppworte betrachtet. Ursache hierfür ist der extrem knappe Umfang der Dokumente mit ca. 4 Begriffen pro Dokument.

Zusammenfassend lässt sich aus der Praxis feststellen, dass Begriffe, die in mehr als 10% aller Dokumente vorkommen, sinnvoll als Stoppworte betrachtet werden können. Ebenso können fast bedenkenlos Begriffe als Stoppwort aufgefasst werden, die nur in einem Dokument vorkommen. Dies führt auch dazu, dass Tippfehler, die nur in einem Dokument auftreten, nicht zu zusätzlichen (falschen) Wörtern in der Dokumentenrepräsentation werden. Dies ist insbesondere im Hinblick auf das Vektorraummodell von Bedeutung.

3.3 Stamm- und/oder Grundformreduktion

Flexionsform

Innerhalb von Dokumenten sind Wörter in verschiedenen Flexionsformen

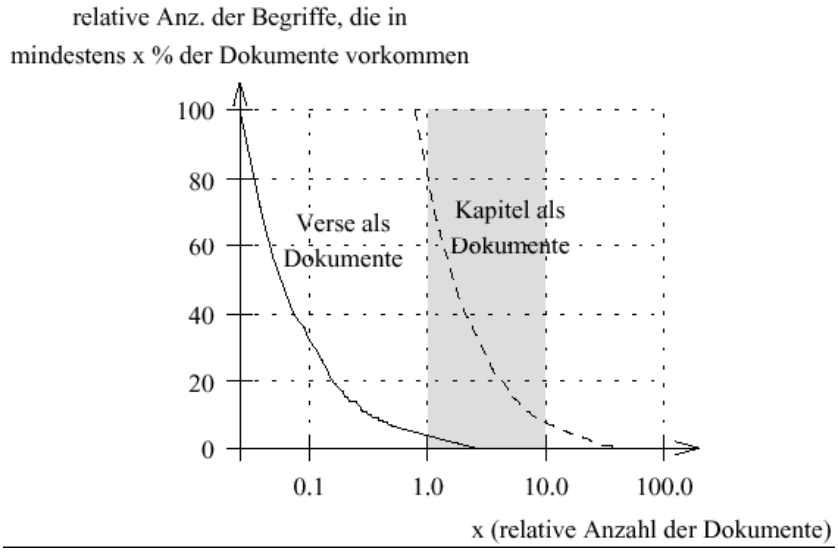


Abbildung 3.1 — Verhältnis zwischen der relativen Anzahl der Begriffe, die in mindestens einem bestimmten Prozentsatz aller Dokumente vorkommen, und der relativen Anzahl der Dokumente

(*Beugungsformen*) zu finden. Dies kann zum einen durch die *Konjugation*, also die Beugung des Zeitwortes, oder durch die *Deklination*, die Beugung des Haupt-, Eigenschafts-, Für- und Zahlwortes, geschehen.

Beugungsform
Konjugation
Deklination

Beispiele hierfür sind:

- **Konjugation**: laufen – lief – gelaufen
- **Deklination**: das Haus – des Hauses – die Häuser

Daneben wird der gleiche Wortstamm in verschiedenen *Derivationsformen* verwendet, bspw.:

- proben – Erprobung – Probe

Wenn man keine Stamm- und/oder Grundformreduktion vornimmt, so findet man zu einem in einer Anfrage gegebenen Wort nur die Dokumente, in denen dieses Wort in der gleichen Form auftritt. Um alle potentiell relevanten Dokumente mit einer Anfrage zu finden, sollte sich

Precision
Recall

das Matching aber nicht auf die einzelnen Wortformen, sondern auf die Wortstämme beziehen. Dies kann zwar die Precision geringfügig beeinträchtigen. Der Recall wird dafür aber in der Regel stark profitieren.

Nutzung im Rahmen der Anfragebearbeitung

Zur Nutzung von Verfahren der Grund- und Stammformreduktion im Rahmen der Anfragebearbeitung in einem IR-System gibt es nun zwei grundsätzliche Ansätze.

Erweiterung der Anfrage

Anfrageerweiterung

Der erste Ansatz geht von einer Erweiterung der Anfrage aus. Dabei bleiben in der Repräsentation sämtliche Wortformen erhalten. Eine Anfrage wird dagegen um alle denkbaren Wortformen erweitert.

Die **Vorteile** dieses Ansatzes liegen in einem vollständigen Informationsgehalt der Repräsentationen, der gegebenenfalls auch Anfragen ermöglicht, die sich auf spezielle Wortformen beziehen.

Als **Nachteil** ist zu werten, dass die Repräsentationen sehr umfangreich werden, da alle Wortformen aufgenommen werden müssen, und gleichzeitig durch den Umfang der Repräsentationen die Performance des Systems sinken wird.

Zusammenfassend kann man feststellen, dass im Allgemeinen die Nachteile bei diesem Vorgehen überwiegen.

Generelle Abbildung der Terme auf die Grund- oder Stammform

Abbildung auf die Grund- und Stammform

Der zweite Ansatz geht von einer generellen Abbildung der Begriffsterme von Anfragen und Dokumenten auf die Grund- und Stammform aus. Die Reduzierung auf die Grund- oder Stammform erfolgt bei der Umwandlung der Anfragen und Dokumente in die jeweilige Repräsentation. In der Repräsentation sind folglich nur noch die Grund- oder Stammformen der Worte enthalten.

Die **Vorteile** liegen in einer deutlich reduzierten Zahl der zu verwaltenden Terme, die gleichzeitig zu einer Steigerung der Systemperformance beiträgt. Ferner ist eine Rückführung auf die Grund- oder Stammform

leichter durchzuführen als eine Expansion auf alle möglichen Wortformen.

Als **Nachteil** ist zu werten, dass eine gezielte Suche nach einer Wortform nun nicht mehr möglich ist. Man verliert damit bei manchen Anfragen an Precision.

Precision

Zusammenfassend lässt sich aber feststellen, dass die Vorteile für den praktischen Einsatz die Nachteile überwiegen.

3.3.1 Grund- und Stammformreduktion

Im Folgenden wollen wir nun genauer auf mögliche Verfahren zur Grund- und Stammformreduktion eingehen. Hierzu ist zunächst ein klares Begriffsverständnis erforderlich.

Grundformreduktion

Die Grundformreduktion führt Wörter auf ihre *grammatikalische Grundform* zurück. Das heißt, dass Substantive auf den Nominativ Singular und Verben auf den Infinitiv zurückgeführt werden. Dies geschieht im Allgemeinen durch Abtrennen der Flexionsendung und eine anschließende Rekodierung. Dabei bildet eine Grundformreduktion immer auf existente Wörter ab.

So wird bspw. das englische Wort *applies* zuerst durch Abschneiden der Flexionsendung auf *appl* reduziert und anschließend durch Anhängen eines *y* auf seine Grundform *apply* rekodiert.

Stammformreduktion

Die Stammformreduktion führt Wörter auf ihren Wortstamm zurück. Dabei handelt es sich aber im Allgemeinen nicht um eine tatsächlich vorkommende Form eines Wortes. Zudem sollte der Stamm für ein Verb und entsprechende Substantive gleich sein.

So werden bspw. die Wörter *computer*, *compute*, *computation*, *computerization* allesamt auf die künstliche Stammform *comput* zurückgeführt.

Bezeichnungen nach Kuhlen

Kuhlen [Kuh77] unterscheidet zwischen drei Verfahren zur Grund- und Stammformreduktion.

Lexikographische Grundform

Grundform, lexikographische

Die **Lexikographische Grundform** stellt die Form dar, in der das Wort in einem Wörterbuch zu finden ist. Die durch Flexion möglicherweise entstandenen graphematischen Veränderungen der Grundform werden rückgängig gemacht. D.h. die Wörter werden jeweils *deflektiert* und anschließend *rekodiert*.

Formale Grundform

Grundform, formale

Unter der **formalen Grundform** versteht Kuhlen *»... Wortfragmente, bei denen die »normalen« englischen und fremdsprachigen (hauptsächlich lateinischen) Flexionsendungen abgetrennt werden, ohne dass die entstandenen Wortfragmente rekodiert würden.«.*

Stammform

Stammform

Unter der **Stammform** versteht Kuhlen nach linguistischen Prinzipien die Zeichenketten, die durch *Deflexion* und Abtrennen von Derivationsendungen entstehen. Diese Zeichenketten sollen soweit wie möglich durch *Rekodierung* vereinheitlicht werden.

Beispiel

Sehen wir uns die verschiedenen Reduktionsformen nach Kuhlen an einem Beispiel (vgl. Tabelle 3.1) an (aus [Kuh77]).

3.3.2 Überblick zu Verfahren zur Grund- und Stammformreduktion

Grund- und Stammformreduktion

Mögliche Verfahren zur Grund- und Stammformreduktion setzen auf verschiedenen Basistechnologien auf. Dies sind im Allgemeinen:

Formale Grundform	Textwörter	Lexikalische Grundform	Stammform
absorb	absorb absorbed absorbing absorbs absorber absorbers	absorb ... ... absorber ...	absorb ... ...
absorbab ...	absorbable absorbably	absorbable ...	... ...
absorbanc	absorbance absorbances absorbancy absorbancies	absorbance ... absorbancy ...	... ...
absorbant	absorbant absorbants absorbantly	absorbant	... ...
absorbtion ... absorbtiv ...	absorbtion absorbtions absorbtively absorbtive	absorbtion ... absorbtive ...	... ...

Tabelle 3.1 — Ein Beispiel zur Abgrenzung von formaler Grundform, lexikalischer Grundform und Stammform

- Trunkierung
- Regeln
- Wörterbücher

Welche Verfahren im Speziellen angewendet werden können, hängt sehr stark von der Sprache ab, die mit einem System verarbeitet werden soll. So ist das Englische schwach flektiert und deshalb gut mit Regeln zu behandeln. Italienisch ist stärker flektiert, aber immer noch mit – allerdings deutlich mehr – Regeln handhabbar. Die Deutsche Sprache kann mit Regeln nicht sinnvoll abgedeckt werden. Man muss daher auf Wörterbücher ausweichen.

3.3.3 Verfahren auf Basis einfacher Trunkierung

Trunkierung

regulärer Ausdruck

Die Beschreibung eines Wortstamms erfolgt hier durch **Trunkierung**. Eine Trunkierung wird dabei im Allgemeinen explizit in der Anfrage vorgenommen. Dies kann beispielsweise in Form von regulären Ausdrücken

erfolgen.

Reguläre Ausdrücke werden bspw. in UNIX (nach [Gul88]) nach folgendem Schema gebildet:

Bedeutung:	**Metazeichen:**
Beliebiges einzelnes Zeichen	.(Punkt)
Beliebige Zeichenkette (auch die leere)	.*
Beliebige Wiederholung des vorangestellten Zeichens (auch keine)	*
Beliebige Wiederholung des vorangestellten Zeichens (mindestens 1)	+
0 oder 1 Wiederholung des vorangestellten Zeichens	?
Eines der Zeichen aus ...	[...]
Eines der Zeichen aus dem Bereich ...	[a-e]
Eines der Zeichen aus den Bereichen ...	[a-eh-x]
Alle Zeichen außer ...	[^...]
Fluchtsymbol	\

Tabelle 3.2 — Regeln zur Bildung regulärer Ausdrücke in UNIX

Mittels regulärer Ausdrücke lässt sich nun eine einfache Anfrage mit Trunkierung folgendermaßen angeben:

H[aä]us.∗ AND Finanz.∗

Diese Anfrage unterstellt, dass sich die einzelnen Anfragepattern jeweils auf Wörter im Text beziehen. Das heißt: *Suche alle Dokumente, in denen mindestens ein Wort vorkommt, das »H[aä]us.*« entspricht und ein Wort, das »Finanz.*« entspricht.*

Ein Beispiel für ein derartiges System stellt der grep-Befehl in UNIX dar, der Verzeichnisse nach bestimmten Dateien durchsucht.

Die Nachteile der Trunkierung sind zum einen in der sehr aufwendigen Anfrageformulierung zu sehen. Reguläre Ausdrücke können in der Formulierung extrem aufwendig werden, wenn sie zu sinnvollen Ergebnissen führen sollen. Zum anderen werden die syntaktischen Regeln der natürlichen Sprache nicht vom System unterstützt. Bspw. müssen typische Zeitformen der englischen Sprache für jedes gesuchte Wort neu eingegeben werden.

Eine Implementierung für ein System mit regulären Ausdrücken kann auf Basis invertierter Listen erfolgen. Wir werden invertierte Listen im weiteren Verlauf des Kurses noch ausführlicher betrachten.

3.3.4 Lovins-Algorithmus zur Grundformreduktion

Algorithmus, Lovins

Algorithmus, regelbasiert

Der von Lovins vorgeschlagene Algorithmus zur Grundformreduktion [Lov68] soll hier exemplarisch für eine Reihe von im Prinzip sehr ähnlichen regelbasierten Algorithmen vorgestellt werden. Er ist insbesondere für Texte im Englischen geeignet.

Der Lovins-Algorithmus arbeitet zweistufig. Dabei trennt er in einem ersten Schritt Endungen ab und führt gegebenenfalls in einem zweiten Schritt eine Transformation der verbliebenen Endung des Wortes durch.

Das Ergebnis des Lovins-Algorithmus ist dabei nicht immer die beste mögliche Endung eines Wortes. Wichtig ist in diesen Fällen aber vor allem die Konsistenz. Das heißt, alle Wörter aus Anfragen und Dokumenten werden nach dem gleichen Verfahren umgeformt.

Vorgehen des Lovins-Algorithmus

Die Abtrennung der Endungen erfolgt in einem ersten Teilschritt anhand der folgenden Tabelle 3.3. Diese Tabelle enthält Endungen, die zum einen absteigend nach ihrer Länge und innerhalb der gleichen Länge alphabetisch aufsteigend sortiert sind.

Von der ersten Zeile der Tabelle beginnend vergleicht der Algorithmus die Endungen mit der Endung eines zu reduzierenden Wortes. Neben der Übereinstimmung der Endung muss zudem die in Tabelle 3.3 ebenfalls für eine Endung angegebene Bedingung erfüllt sein.

Die weiteren Teilschritte sind nun:

1. Falls ein Wort mit einem Konsonanten ungleich *s* endet, der von einem *s* gefolgt wird: Entferne das *s*

   Beispiele:

   - *stems* wird zu *stem*
   - aber *stress* bleibt *stress*

2. Falls ein Wort mit *es* endet: Entferne das abschließende *s*

   Beispiele:

   - *places* wird zu *place*

Länge	Endung	Bedingung
11	alistically	B
	arizability	A
	izationally	B
10	antialness	A
	arisations	A
	arizations	A
	entialness	A
9	allically	C
	antaneous	A
4	able	A
	ably	A
	ages	B
	ally	B
3	ism	B
1	e	A

Bedingungen:
A: keine Einschränkungen
B: verbleibender Wortstamm min. 3 Zeichen lang
C: verbleibender Wortstamm min. 4 Zeichen lang

Tabelle 3.3 — Bedingungen und Wortendungen nach Lovins

- *likes* wird zu *like*
- *theses* wird zu *these* (»Fehler!«)
- *indices* wird zu *indice* (»Fehler!«)
- *syntheses* wird zu *synthese* (»Fehler!«)

Offensichtlich führt diese Regel bei Wörtern griechischen Ursprungs, deren Singular mit *is* endet, zu Problemen. Man könnte versuchen, dies durch weitere Regeln in den Griff zu bekommen.

3. Überführe *iev* zu *ief* und *metr* zu *meter*.

   Beispiel:

   - *believable* wird zu *believ* und nun zu *belief*
     (1. Schritt = Abschneiden der Endung)

4. Falls ein Wort mit *ing* endet: Lösche das *ing*, es sei denn, das Wort besteht nach der Löschung nur aus einem Buchstaben oder aus *th*.

   Beispiele:

   - *thinking* wird zu *think*
   - *singing* wird zu *sing*
   - *sing* wird zu *sing* (keine Änderung)
   - *thing* wird zu *thing* (keine Änderung)
   - *preceding* wird zu *preced* (Fehler; kein Wort!)

   Den Fehler im letzten Beispiel könnte man z.B. durch eine nachgeschaltete Regel beheben, die besagt: Sofern ein Wort nach der Reduktion mit *et*, *ed* oder *es* endet, wird ein *e* hinzugefügt.

5. Falls ein Wort mit *ed* endet und ein Konsonant vorausgeht: Lösche das *ed*, es sei denn, das Wort besteht nach der Löschung nur aus einem Buchstaben.

   Beispiele:

   - *ended* wird zu *end*
   - *red* wird zu *red*
   - *proceed* wird zu *proceed* (es geht kein Konsonant voraus)
   - *proceeded* wird zu *proceed*

6. Falls nach dem Entfernen der Endung ein Wort mit *bb, dd, ..., tt* endet: Entferne einen der doppelten Buchstaben.

   Beispiele:

   - *embedded* wird zu *embedd* und nun zu *embed*

7. Falls ein Wort mit *ion* endet: Entferne *ion*, sofern das verbleibende Wort mehr als 2 Buchstaben hat. Sofern der letzte Buchstabe des Stammes ein Konsonant ist und der vorhergehende ein Vokal, füge zusätzlich ein *e* an.

   Beispiele:

   - *direction* wird zu *direct*
   - *polution* wird zu *polute*
   - *plantation* wird zu *plantate* (Fehler!?)

- *zion* wird zu *zion*
- *scion* wird zu *scion*
- *anion* wird zu *anion*
- *cation* wird zu *cate* (Fehler!)

Man beachte, dass *cation* (das Kation) ein Kunstwort aus *cathode* und *ion* ist. Folglich gibt es hier keinen eigentlichen Wortstamm.

Als Fazit lässt sich Folgendes zusammenfassen. Ein arbeitsfähiges System zur Grundformreduktion für die englische Sprache benötigt zwischen 10 bis 20 Regeln. Um die Qualität dieses Systems zu steigern sind ferner zahlreiche Ausnahmen z.B. für irreguläre Verben erforderlich. Das System sollte ferner die iterative Anwendung der Regeln unterstützen, so wird z.B. zuerst aus *directions direction* und durch wiederholte Anwendung der Regel die Grundform *direct*.

Betrachtung von Vorsilben

Der bisher vorgestellte Algorithmus betrachtet nur das Abschneiden der Endungen. Zusätzliche Teilsysteme oder Regeln zur Entfernung von Vorsilben sind zwar ebenfalls möglich, allerdings erscheinen sie für die Suche wenig sinnvoll. Folgendes Beispiel soll dies verdeutlichen.

Beispiel:

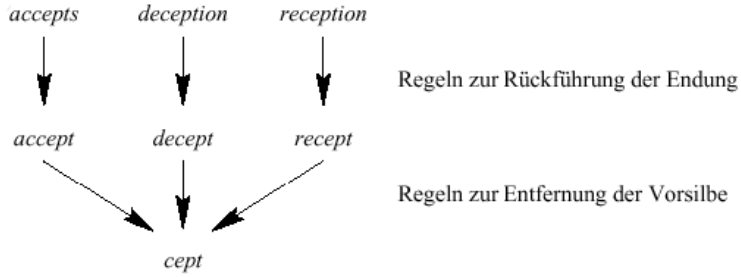


Abbildung 3.2 — Beispielhafte Rückführung der Endungen von drei Wörtern und anschließende Entfernung der Vorsilben

Obwohl alle drei Wörter aus Abbildung 3.2 vom gleichen lateinischen Stamm herrühren, sind die heutigen Bedeutungen doch recht unterschiedlich. In der Folge würden durch eine Entfernung der Vorsilben die

einzelnen Wörter ihre Unterscheidungskraft verlieren. Daher beschränkt man sich in der Praxis auf die Entfernung der Endsilben.

Weitere regelbasierte Algorithmen

Wie bereits erwähnt, ist der oben beschriebene Algorithmus von Lovins nur ein Beispiel für einen regelbasierten Algorithmus zur Grundformreduktion. Ein weiterer interessanter Algorithmus wurde 1980 von Porter vorgestellt [Por80]. Dieser Algorithmus von Porter – der hier aus Gründen des Kursumfangs leider nicht vorgestellt werden kann – wird heute in der Praxis häufig für englische Texte eingesetzt. Er basiert auf den gleichen Ideen wie der Algorithmus von Lovins, verfügt aber über einen etwas komplexeren Mechanismus zur Angabe von Bedingungen, die erfüllt sein müssen, damit die einzelnen Regeln angewendet werden können.

Algorithmus von Porter

In der Literatur und im Netz finden sich zahlreiche weitere Algorithmen zum »Stemming« für englischsprachige Texte. Als Beispiel sei hier nur der *Lancaster Stemming Algorithm*[1] genannt, zu dem sich auf den dortigen Webseiten auch Implementierungen in verschiedenen Programmiersprachen finden.

3.3.5 Verfahren, die auf Wörterbüchern basieren

Regelbasierte Verfahren zur Grund- und Stammformreduktion sind nur für schwach flektierte Sprachen, wie bspw. die englische Sprache, geeignet. Daher setzt man für stark flektierte Sprachen, wie die deutsche Sprache, wörterbuchbasierte Verfahren ein.

Ein entsprechendes Wörterbuch sollte sowohl Flexionsformen als auch Derivationsformen auf ihre Grundform rückführen können:

Wörterbuch

- Flexionsform → Grundform
  - lief → laufen
  - Häuser → Haus
- Derivationsform → Grundform
  - Lieblosigkeit → lieblos
  - Berechnung → rechnen

[1] *http://www.comp.lancs.ac.uk/computing/research/stemming/index.htm*, letzter Abruf: 26.10.2006

Ein Problem ist dabei die ständige Pflege, die derartige Wörterbücher benötigen. So müssen bspw. neue Wortschöpfungen aufgenommen werden. Zum anderen müssen für Fachvokabulare entsprechende Wörterbücher vorhanden sein oder das benutzte Wörterbuch muss entsprechend angepasst werden.

3.4 Mehrwortgruppenidentifikation

Mehrwortgruppe

Eine Mehrwortgruppe stellt ein Konzept dar, das aus mehreren Wörtern besteht. Vor allem im Deutschen – aber auch in anderen Sprachen – stellen Mehrwortgruppen ein Problem bei der Suche nach entsprechenden Dokumenten dar.

Wenn wir bspw. nach Dokumenten suchen, die den Begriff *»Bundeskanzlerwahl«* enthalten, so wollen wir sicher auch Dokumente finden, die den Text *»die Wahl des Bundeskanzlers«* enthalten. Wie schwierig sich eine optimale Identifikation von Mehrwortgruppen gestaltet wird aber deutlich, wenn wir uns klar machen, dass ein Dokument ja auch die Textpassage *»die Wahl des Bundeskanzlers fiel auf Saumagen mit Kraut«* enthalten kann. Hier ist der Gegenstand der Wahl eben nicht der Bundeskanzler sondern das Gericht, das er essen möchte. Eine optimale Lösung des Mehrwortgruppenproblems würde somit ein automatisches Textverstehen erforderlich machen.

Im Deutschen kann man nun im Hinblick auf die Betrachtung von Mehrwortgruppen zwei grundsätzliche Ansätze verfolgen:

- Man kann das Problem der Mehrwortgruppenidentifikation ignorieren und sich im IR-System mit der Betrachtung von Wörtern begnügen, die ggf. auf ihre Grund- oder Stammform zurückgeführt sind. In diesem Fall würden Begriffe wie *»Wahl«*, *»Bundeskanzler«* und *»Bundeskanzlerwahl«* als eigenständige Terme in den Dokumentrepräsentationen auftreten. Dies hätte unter anderem zur Folge, dass man dann, wenn man nach *»Wahl«* sucht, keine Dokumente findet, die nur das Wort *»Bundeskanzlerwahl«* aber nicht das Wort *»Wahl«* enthalten. Dieser Ansatz führt zwar zu einer ordentlichen Precision, der Recall leidet aber deutlich.

Precision
Recall

- Alternativ kann man versuchen, zusammengeschriebene Mehrwortgruppen in ihre Bestandteile zu zerlegen. Für das Wort *»Bundeskanzlerwahl«* würden dann in der Repräsentation eines Doku-

mentes die beiden Begriffe *»Bundeskanzler«* und *»Wahl«* aufgenommen. Dies erhöht bei der Anfragebearbeitung zwar den Recall, eine gezielte Suche nach *»Bundeskanzlerwahl«* ist nun aber nicht mehr möglich.

Recall

In der englischen Sprache gestaltet sich die Mehrwortgruppenidentifikation anders, da es fast keine zusammengeschriebenen Wörter wie im Deutschen gibt. Man wäre damit ohne explizite Betrachtung der Problematik automatisch bei der zweiten für das Deutsche skizzierten Möglichkeit. Allerdings bekommen natürlich auch im Englischen Begriffe oft erst durch ihre Zusammenstellung eine bestimmte Bedeutung, wie bspw. *»information retrieval«* oder *»artificial intelligence«*. Dies bedeutet: wenn ein Information Retrieval System nur die einzelnen Begriffe kennt, kann es keine besonders gute Precision erreichen.

Precision

In der englischen Sprache benötigt man daher einen Ansatz um so genannte *compound terms* oder *phrases* – eben Mehrwortgruppen – zu finden. Auf denkbare Ansätze hierzu wollen wir im Folgenden kurz eingehen.

3.4.1 Ermittlung von Mehrwortgruppen

Zur Ermittlung von Mehrwortgruppen gibt es im Prinzip zwei Ansätze.

Der pragmatische Ansatz basiert auf der Betrachtung des Wortabstandes. Das heißt, wenn die Wörter A und B eine bekannte Mehrwortgruppe bilden, und die Worte A und B in einem Text mit maximal n Worten dazwischen auftreten, dann interpretiert man dies als Auftreten der Mehrwortgruppe. Dieser Ansatz kann durch die Berücksichtigung der Reihenfolge und von Satzgrenzen noch verfeinert werden. Dabei können aber bestimmte Probleme auftreten. Bspw. haben die drei, nach dieser Auffassung vergleichbaren Mehrwortgruppen *»college junior«*, *»junior college«* und *»junior in college«* völlig unterschiedliche Bedeutungen.

Wortabstand

Einen anderen Ansatz bildet die Verwendung eines Parsers, der die grammatikalische Satzstruktur ermittelt. Der Parser versucht dabei einen gegebenen Satz auf eine der ihm bekannten möglichen Satzformen abzubilden. Das Problem ist, dass sich u.U. nicht alle Sätze an die Grammatik halten, und dass die möglichen Satzformen ausgesprochen komplex sind. Trotzdem gibt es insbesondere für die englische Sprache mehrere

Parser

Satzform

heute durchaus praktisch einsetzbare Parser. Ein Beispiel ist in [Str94] beschrieben. Dabei werden die Satzstrukturen rekonstruiert und immer dann, wenn die einzelnen Bestandteile einer Mehrwortgruppe in einer bestimmten Konstellation auftreten, kann davon ausgegangen werden, dass die Mehrwortgruppe gemeint ist.

3.4.2 Mehrwortgruppen-Identifikation im Darmstädter Indexierungsansatz

Darmstädter Indexierungsansatz

Ein einfaches, robustes Verfahren zur Identifikation von Mehrwortgruppen ist im Rahmen der Arbeiten zum Darmstädter Indexierungsansatz [Lus86] entwickelt worden.

Ausgangspunkt dabei ist ein Mehrwortgruppen-Wörterbuch, das automatisch erstellt wurde. Dieses Mehrwortgruppen-Wörterbuch wurde durch die Analyse von zahlreichen Texten gewonnen, indem in einem ersten Schritt besonders lange Worte als Mehrwortgruppen in den Texten identifiziert wurden, die dann in einem weiteren Schritt in ihre Einzelwörter zerlegt wurden. Dieser automatisierbare Ansatz wurde gewählt, damit möglichst wenig manuelle Tätigkeiten im Zusammenhang mit dem Aufbau dieses Wörterbuchs nötig werden. Dieses Verfahren funktioniert nur für deutsche Texte, auch wenn das im folgende gegebene Beispiel einen englischen Text präsentiert.

Bei der Analyse eines Textes wird nun beim Auftreten einer Komponente einer Mehrwortgruppe zunächst geprüft, ob die restlichen Komponenten dieser Mehrwortgruppe ebenfalls innerhalb eines vorgegebenen Maximalabstands im Text auftreten.

Wir wollen dies an einem Beispiel verdeutlichen. Gegeben sei der Text aus Abbildung 3.3:

In diesem Text werden unter anderem die folgenden Mehrwortgruppen identifiziert. Die zu diesen Mehrwortgruppen gehörenden Komponenten sind im Text jeweils fettgedruckt:

- TUNNEL JUNCTION
- TUNNEL CURRENT
- ELECTRICAL CONDUCTIVITY
- ALUMINIUM SUBSTRATES

Current-voltage spectra of metal/oxide/SnTe diodes. Pt. 1

In metal/oxide/SnTe **tunnel junctions** (where the oxide is Al_2O_3 or SiO_2 and the metal is lead or **aluminium**) on BaF_2 or $NaCl$ **substrates** the **tunnel current** $I(U)$ and its derivatives $I'(U)$ and $I''(U)$ were **measured** at 4.2 K. Additionally the Hall **coefficient** and **electrical conductivity** of the monocrystalline SnTe **films** were determined at the same temperature. The pronounced oscillations in I'' suggest the existence of a quantum size effect in the very thin SnTe films in several cases, although this is complicated by various other processes. The most important features of the different types are discussed briefly.

Abbildung 3.3 — Ein Beispieltext.

- MEASURE ELECTRICAL
- FILM COEFFICIENT

Wenn eine Mehrwortgruppe auf diese Art identifiziert wurde, dann werden die Werte folgender formaler Kriterien zur Beschreibung der Form des Vorkommens der Mehrwortgruppe verwendet:

1. Der Abstand zwischen den Komponenten, wobei bei der Bestimmung des Wortabstandes alle Wörter, nur Stoppwörter oder nur relevante Wörter gezählt werden können.
2. Die Reihenfolge der Komponenten, d.h. stimmt die Reihenfolge im Text mit der im Wörterbuch überein?
3. Sind alle Komponenten im gleichen Satz?
4. Für jede Komponente: Besteht Grundformgleichheit mit dem Wörterbucheintrag, oder nur Gleichheit in der Stammform?

Entsprechend den Werten dieser Kriterien kann man nun Klassen von Vorkommensformen bilden.

Durch den Vergleich mit einer (intellektuell erstellten) korrekten Identifikation von Mehrwortgruppen kann man dann für jede Klasse die

Wahrscheinlichkeit bestimmen, dass eine so identifizierte Mehrwortgruppe auch syntaktisch korrekt ist.

Im obigen Beispiel sind die ersten drei Mehrwortgruppen syntaktisch korrekt, die übrigen drei aber syntaktisch falsch. Allerdings ist *MEASURE ELECTRICAL* zwar syntaktisch falsch, aber semantisch korrekt, so dass bezogen auf das Retrieval kein eigentlicher Fehler vorliegt.

Es ist offensichtlich, dass dieses einfache Verfahren für keine der betrachteten Klassen vollkommen korrekt ist. Gleiches gilt aber für alle Arten von computerlinguistischen Verfahren, die letztlich versuchen Satzstrukturen zu verstehen. Daher muss man versuchen, die hieraus resultierende Unsicherheit beim Retrieval z.B. durch eine entsprechende Gewichtung der Vorkommen zu berücksichtigen.

Experimentelle Untersuchungen haben ergeben, dass von den oben genannten Merkmalen nur das erste und das vierte die Identifikationssicherheit signifikant beeinflussen, während die anderen Kriterien keinen nennenswerten Einfluss haben.

3.4.3 Ein Nachsatz zur Begriffswelt

Präkombination, Präkoordination und Postkoordination

Hier wollen wir noch ein paar Definitionen zur Begriffswelt nachreichen.

Präkombination

- Unter **Präkombination** versteht man in der Indexierungssprache angelegte Begriffsverknüpfungen, also Komposita oder Mehrwortausdrücke. Die Idee ist dabei, dass bereits bei der Erstellung der Dokumente ein so genanntes *kontrolliertes Vokabular* verwendet wird, das auch als *Indexierungssprache* bezeichnet wird. Das heißt Komposita dürfen in den Dokumenten nicht in anderen Formen vorkommen. Beispiele für Deskriptoren sind »Halswirbelfraktur« oder »Europäischer Binnenmarkt«.

Präkoordination

- Unter der **Präkoordination** versteht man Verknüpfungen, die im Zuge der Indexierung – also bei der Erstellung der Repräsentationen – hergestellt werden. Deskriptoren sind hier bspw. »Halswirbel«, »Europa«, »Binnenmarkt« oder »Fraktur«. Bei der Indexierung werden dann die entsprechenden Verknüpfungen hergestellt, also bspw. »Halswirbel / Fraktur« oder »Europa / Binnenmarkt«.

- Unter **Postkoordination** versteht man schließlich Verknüpfungen, die während der Suche mittels Boolescher oder sonstiger Operatoren gebildet werden. Beispiele hierfür sind »Europa UND Binnenmarkt« oder »(Binnenmarkt ODER Währungsunion) UND Europa«. *Postkoordination*

Die Begriffe helfen damit den Zeitpunkt (Dokumentenerstellung, Indexierung oder Anfrageformulierung) zu definieren, zu dem die Problematik der eindeutigen Verwendung von Mehrwortgruppen adressiert wird.

3.5 Terminologische Kontrolle

Terminologische Kontrolle

Das Gebiet der Terminologischen Kontrolle stammt aus den Ursprüngen des Information Retrieval – dem Bibliothekswesen. Um die Mehrdeutigkeit der Sprache in den Griff zu bekommen, versucht man für die Verschlagwortung eine »eindeutige« Begriffsverwendung zu erzwingen. Dies betraf und betrifft im Bibliothekswesen vor allem die manuelle Indexierung innerhalb einer Bibliothek und über verschiedene Bibliotheken hinweg.

Die manuelle Indexierung wird als zweistufiger Prozess aufgefasst, der im ersten Schritt das *Erkennen* der wieder auffindbar zu machenden Essenz eines Textes umfasst und im zweiten Schritt das *Wiedergeben* dieser Essenz in einer ausreichend gut voraussehbaren und ausreichend wiedergabegetreuen Form.

Dazu verwendet man eine künstliche Sprache in der die verwendeten Bezeichnungen eindeutig auf einen definierten Begriff bezogen und Begriffe nur durch eine Bezeichnung repräsentiert sind. Diesen einen definierten Begriff bezeichnet man als *Vorzugsbenennung*. *Vorzugsbenennung*

Indexierungssprache

Indexierungssprache

Unter einer Indexierungssprache versteht man eine künstliche Sprache, mit deren Hilfe der zweite Teil des Indexierungsprozesses – also das Wiedergeben der Essenz eines Textes – ausgeführt wird. Diese künstlichen Sprachen sind zudem geregelt. Das heißt, sie unterliegen einer terminologischen Kontrolle, wodurch nur bestimmte Begriffe zur Indexierung verwendet werden dürfen. Die Elemente einer Indexierungssprache werden auch *Deskriptoren* genannt. *Deskriptor*

Unter der Terminologiekontrolle versteht man alle Regeln und Aktivitäten, die einen Deskriptor eindeutig definieren und Mehrdeutigkeiten und Unklarheiten ausschalten. Mehrdeutigkeiten werden bspw. durch Synonyme, Homonyme oder Polyseme gebildet.

Synonymkontrolle

Synonymkontrolle
Äquivalenzklasse

Bei der Synonymkontrolle werden Bezeichnungen zu Äquivalenzklassen zusammengefasst. Man kann folgende Arten von Synonymie unterscheiden:

Schreibweisenvariante

- Schreibweisenvarianten, wie bspw. »Friseur«, »Frisör« oder »UN«, »UNO«, »Vereinte Nationen«

Konnotation
Sprachstil

- unterschiedliche Konnotationen, Sprachstile, Verbreitungsgebiete, wie bspw. »Telefon«, »Fernsprecher« oder »Pferd«, »Gaul« oder »Myopie«, »Kurzsichtigkeit«

Quasi-Synonym

- Quasi-Synonyme, wie bspw. »Schauspiel«, »Theaterstück« oder »Rundfunk«, »Hörfunk«

Deskriptor

Durch die Synonymkontrolle kann so zum Beispiel den Synonymen »Ross«, »Gaul«, »Mähre«, »Klepper« usw. der Deskriptor bzw. die Vorzugsbenennung »Pferd« zugeordnet werden. In diesem Fall ist der dem Deskriptor zugeordnete Begriff ebenfalls »Pferd«.

Thesaurus

Um die Synonymkontrolle ausführen zu können, verwendet man *Thesauri* (vergleiche hierzu auch den Abschnitt 3.5). In einem derartigen Thesaurus werden über die Zuordnung von Synonymen hinaus auch Begriffe mit geringen oder irrelevanten Bedeutungsdifferenzen zu Äquivalenzklassen zusammengefasst.

Im einzelnen können dies folgende Beziehungen sein:

- unterschiedliche Spezifität, wie bspw. »Sprachwissenschaft« und »Linguistik«,
- Antonyme, wie bspw. »Härte« und »Weichheit«,
- zu spezieller Unterbegriff, wie bspw. »Weizen« und »Winterweizen« oder

- Gleichsetzung von Verb und Substantiv bzw. Tätigkeit und Ergebnis, wie bspw. bei »Wohnen« und »Wohnung«.

Die Entscheidung, ob zwei Begriffe als Quasi-Synonyme zu behandeln sind, hängt dabei immer von der jeweiligen Anwendung ab.

Polysemkontrolle

Polysemkontrolle
Äquivalenzklasse

Bei der Polysemkontrolle werden mehrdeutige Bezeichnungen auf mehrere Äquivalenzklassen aufgeteilt. Dabei kann man noch zwischen Homographen und den eigentlichen Polysemen unterscheiden.

Homograph

Ein Beispiel für **Homographen** stellt das Wort *Tenor* dar. Unter *Tenor* versteht man »Haltung; Inhalt, Sinn, Wortlaut« während man unter einem *Tenor* eine »hohe Männerstimme; Tenorsänger« versteht. Bei diesem Beispiel kann eine Unterscheidung noch anhand der sprachlichen Betonung erfolgen. Bei schriftlichen Texten ist nur eine Unterscheidung anhand des Kontextes möglich.

Polysem

Ein Beispiel für eigentliche **Polyseme** stellt das Wort *Bank* dar, das sowohl eine Sitzgelegenheit aber auch ein Geldinstitut bezeichnen kann. Hier ist eine Unterscheidung nur aus dem Kontext möglich.

Morphologische Zerlegung

Morphologische Zerlegung
Morphem

Die Terminologiekontrolle muss zudem die Begriffszerlegung der Mehrwortgruppen gewährleisten. Dies bezeichnet man als Morphologische Zerlegung. Dabei werden die Mehrwortgruppen sprachlich in ihre Wortbildungselemente, die sog. Morpheme zerlegt.

Beispiele für Mehrwortgruppen und die entsprechenden Morpheme sind:

- Abfallbeseitigung: Abfall UND Beseitigung
- Krankenhausbibliothek: Krankenhaus UND Bibliothek

Allerdings gibt es auch Mehrwortgruppen, die sich nicht bzw. nicht sinnvoll in Morpheme zerlegen lassen. So erscheint eine Zerlegung von *Eisenbahn* in *Eisen UND Bahn* wie auch eine Zerlegung von *Handschuh* wenig sinnvoll.

Semantische Zerlegung

Die *Semantische Zerlegung* versucht dieses Problem zu lösen und zerlegt Mehrwortgruppen in begriffliche Einheiten, die in Kombination den Ausgangsbegriff reproduzieren. So lassen sich bspw. die Begriffe *Eisenbahn* in *Schienenverkehr UND Überlandverkehr* sowie *Handschuh* in *Hand UND Bekleidung* semantisch zerlegen.

Thesaurus

Thesaurus

Zur terminologischen Kontrolle werden dabei im Allgemeinen Thesauri verwendet.

Definition

[Kom75] definiert einen Thesaurus wie folgt:

> *»Ein Thesaurus ist eine natürlich-sprachlich basierte Dokumentationssprache, die die umkehrbar eindeutige Zuordnung von Begriffen und Bezeichnungen der natürlichen Sprache anstrebt, indem sie vollständige Vokabularkontrolle und terminologische Kontrolle ausübt und die Begriffe sowie Relationen zwischen ihnen durch Darstellung von Relationen zwischen den Bezeichnungen und gegebenenfalls zusätzliche Hilfsmittel darstellt.«*

Deskriptor

Wichtig ist in diesem Zusammenhang vor allem der Begriff des Deskriptors.

Calvin Mooers führte 1956 das Konzept des Deskriptors ein. Ein Deskriptor wurde nicht einfach als Wort aufgefasst, sondern als Repräsentant einer Wortfamilie bzw. eines Begriffes. Die Bedeutung wird durch Definition festgelegt und hat nur innerhalb eines Systems Gültigkeit.

Thesaurusrelationen

Thesaurusrelationen

Die Beziehungen zwischen den Wörtern werden in einem Thesaurus als Thesaurusrelationen bezeichnet. Bspw. werden nach DIN 1643-1 drei Grundtypen von Relationen unterschieden:

1. Äquivalenzrelationen, also die Beziehung zwischen wirklichen Synonymen,
2. Hierarchierelationen, die sich unterteilen in
   (a) Abstraktionsrelation (generische Relation: »Lkw ist ein Kraftfahrzeug«) und
   (b) Bestandsrelation (partitive Relation: »Ein Pkw besteht aus einem Motor, ...«) sowie
3. Assoziationsrelationen, die verwandte Begriffe bezeichnen, z.B. »SQL« und »Datenbanken«.

Die Relationen lassen sich nun wie in Tabelle 3.4 gezeigt darstellen, wenn man zwischen Abstraktions- und Bestandsrelationen nicht unterscheiden will.

Kurzzeichen		Bedeutung
Deutsch	Englisch	
TT	TT	Top Term (Kopfbegriff einer Hierarchie)
OB	BT	übergeordneter Begriff
UB	NT	untergeordneter Begriff
VB	RT	verwandter Begriff (Assoziationsrelation)

Tabelle 3.4 — In einem Thesaurus zu verwaltende Relationen, wenn man zwischen Abstraktions- und Bestandsrelationen nicht unterscheiden will

Abstraktionsrelation
Bestandsrelation

Will man zwischen Abstraktions- und Bestandsrelationen unterscheiden, so lassen sich die Relationen OB und UB in jeweils zwei Relationen unterteilen (s. Tabelle 3.5).

Dabei stehen in den Tabellen: B für *broader*, N für *narrower*, T für *term*, G für *generalization* und P für *participation*.

Nutzung eines Thesaurus in einem IR-System

Anfrageerweiterung

Ein Thesaurus kann in einem IR-System zunächst zur *Anfrageerweiterung* eingesetzt werden. Dabei werden zu den An-

Kurzzeichen		Bedeutung
Deutsch	Englisch	
OA	BTG	Oberbegriff (Abstraktionsrelation)
UA	NTG	Unterbegriff (Abstraktionsrelation)
SP	BTP	Verbandsbegriff (Bestandsrelation)
TP	NTP	Teilbegriff (Bestandsrelation)

Tabelle 3.5 — Aufteilung der Relationen OB und UB, wenn man zwischen Abstraktions- und Bestandsrelationen unterscheiden will

Synonym *Recall* *Precision*

fragetermen auf Basis des Thesaurus Synonyme sowie Unter- und Oberbegriffe hinzugefügt. Dies erhöht offensichtlich den Recall und beeinträchtigt andererseits die Precision . Es erscheint daher ratsam die Verwendung des Thesaurus optional anzubieten. Wenn ein Benutzer in einer ersten Anfrage einen zu geringen Recall erzielt hat, kann er in einem zweiten Versuch eine automatische Erweiterung der Anfrage um Synonyme sowie Unter- und Oberbegriffe vornehmen lassen.

Neben dieser Anfrageerweiterung kann ein Thesaurus auch bereits bei der Erstellung der Dokumentenrepräsentationen eingesetzt werden. In diesem Fall sollte der Thesaurus auch Informationen zu Vorzugsbenennungen enthalten, damit diese statt der tatsächlich im Text vorkommenden Wörter in die Repräsentationen aufgenommen werden können. Dies reduziert den Speicherplatzbedarf und erhöht ebenfalls den Recall. Man kann nun aber nicht mehr nach einem konkreten Wort suchen und muss im Allgemeinen eine geringere Precision in Kauf nehmen.

Recall

Precision

3.6 Ein sprachunabhängiger Ansatz

Ein sprachunabhängiger Ansatz mit Hilfe von n-grams

Ein Problem der bisher in diesem Kapitel behandelten Ansätze zur Grund- und Stammformreduktion, zur Mehrwortgruppenbetrachtung und zu Thesauri liegt in der festen Bindung an eine Sprache. Dies hat zur Folge, dass immer dann, wenn ein IR-System Dokumente in einer neu-

en Sprache (z.B. Spanisch) verwalten soll, entsprechende Erweiterungen vorgenommen werden müssen.

Ein Ansatz, der sprachunabhängig eingesetzt werden kann, ist die Indexierung auf Basis so genannter *n-grams*. Ein n-gram ist zunächst nichts anderes, als ein Substring der Länge n eines gegebenen Wortes.

n-gram

Beispiel

Für $n = 3$ und das Wort *Eisenbahn* ergeben sich folgende *Trigrams*:

`eis`, `ise`, `sen`, `enb`, `nba`, `bah`, `ahn`

Für $n = 2$ ergeben sich folgende *Bigrams*:

`ei`, `is`, `se`, `en`, `nb`, `ba`, `ah`, `hn`

Gerade bei höheren Werten für n nimmt man oft zusätzlich auch ein Begrenzerzeichen für die Wortgrenzen auf:

`#ei`, `eis`, `ise`, `sen`, `enb`, `nba`, `bah`, `ahn`, `hn#`

Nutzung von n-grams

Man kann nun zur Indexierung statt Wörter n-grams verwenden. Ein Dokument wird dann z.B. durch die Menge der in ihm vorkommenden 5-grams repräsentiert. In der Literatur gibt es Berichte über zahlreiche erfolgreiche Einsätze von n-grams. So beschreibt Frakes ein sprachabhängiges Stemming mit Hilfe von n-grams [Fra92]. In einer anderen Publikation beschreiben James Mayfield und Paul McNamee einen Ansatz, bei dem das Vektorraummodell auf Basis von 5-grams angewendet wird [MM97]. Sie vergleichen dabei drei Ansätze miteinander:

1. Wörter werden als Terme (= Dimensionen des Vektorraumes) verwendet. Diese Wörter werden keiner Stammformreduktion unterzogen.
2. Stammformen werden als Terme verwendet. Dabei wird der Algorithmus von Porter angewendet.
3. 5-grams werden als Terme verwendet. Dazu wird zuerst die Interpunktion aus dem zu indexierenden Text entfernt und dieser in Kleinbuchstaben umgewandelt. Zahlen im Text werden auf ein einziges Zeichen abgebildet

Der Vergleich der Ansätze auf Basis der TREC-Kollektion zeigt, dass für kurze Anfragen Wörter und Stammformen als Dimensionen im Vektorraummodell günstiger abschneiden als 5-grams. Für lange Anfragen arbeiten dagegen die 5-grams besser.

Ein Problem bei der Verwendung von n-grams ist allerdings, dass es keine einfache Möglichkeit gibt, einem Anfragenden zu erklären, warum ein Dokument als wichtig erachtet wurde. Bei der Verwendung von Wörtern oder Stammformen können bspw. die entsprechenden Wörter im gefundenen Dokument hervorgehoben werden. Dies ist beim Einsatz von n-grams kaum sinnvoll.

3.7 Auszeichnung mittels Metadaten

Ein anderer Weg, um der Vagheit der natürlichen Sprache zu begegnen, ist die Auszeichnung von Dokumenten mit so genannten Metadaten. Dabei werden für einzelne Dokumente oder Objekte einer Kollektion zusätzlich zum eigentlichen Dokumentinhalt bestimmte Felder eingeführt und verwaltet, die z.B. Titel, Verfasser oder Erstellungsdatum eines Dokumentes wiedergeben. Ein Beispiel dafür ist der *Dublin Core*-Standard.

Einen Schritt weiter geht die Vision des *Semantic Web*, die sich eine vollständige Erschließung und das Verständnis der Inhalte von Dokumenten durch Maschinen zum Ziel gesetzt hat.

3.7.1 Dublin Core

Dublin Core

Die Dublin Core Metadata Initiative (DCMI) konstituierte sich 1995 zum ersten Mal in Dublin, Ohio, auf einem Workshop, der vom *Online Computer Library Center* (OCLC) und dem *National Center for Supercomputing Applications* (NCSA) veranstaltet wurde.

Ziel war die Entwicklung eines standardisierten Metadatensatzes, der zur Beschreibung von digitalen Dokumenten bzw. so genannten *document-like objects* dienen sollte. Der Standard soll dabei so ausgelegt sein, dass er eine allgemein verständliche Semantik ausweist und zudem erweiterbar ist. Die Metadaten sollen, anders als bspw. in Bibliotheken, bereits durch die Autoren der Dokumente selbst erstellt werden.

Der Kernsatz von Dublin Core umfasst heute 15 Elemente, die in Tabelle 3.6 mit ihrer Beschreibung (nach [RF96]) aufgeführt sind. Jedes

der Elemente kann optional angegeben werden. Elemente können zudem wiederholt aufgeführt werden, bspw. bei mehreren »Contributors«.

Element	Beschreibung
Title	Titel der Quelle; der vom Verfasser, Urheber oder Verleger vergebene Name der Ressource.
Creator	Die Person(en) oder Organisation(en), die den intellektuellen Inhalt verantworten. Z.B. Autoren bei Textdokumenten; Künstler, Photographen bzw. auch andere Bezeichnungen wie Komponist und Maler bei graphischen Dokumenten.
Subject	Thema, Schlagwort, Stichwort. Das Thema der Ressource bzw. Stichwörter oder Phrasen, die das Thema oder den Inhalt beschreiben.
Description	Eine textliche Beschreibung des Ressourceninhalts inklusive einer Kurzfassung (Abstract) bei dokumentähnlichen Ressourcen oder Inhaltsbeschreibungen bei graphischen Ressourcen.
Publisher	Die Einrichtung, die verantwortet, dass diese Ressource in dieser Form zur Verfügung steht, wie z.B. ein Verleger, ein Herausgeber, eine Universität oder ein Unternehmen.
Contributors	Zusätzliche Person(en) und Organisation(en), die einen bedeutsamen intellektuellen Beitrag zur Ressource geleistet haben.
Date	Das Datum, an dem die Ressource in der gegenwärtigen Form zugänglich gemacht wurde, bspw. 20021201 für den 01. Dezember 2002.
Type	Die Art der Ressource, z.B. Homepage, Roman, Gedicht, Arbeitsbericht, technischer Bericht, Essay oder Wörterbuch.
Format	Hier wird das datentechnische Format der Ressource eingetragen, z.B. Text/HTML, JPEG-Bilddatei etc.
Identifier	Hier wird eine Zeichenkette oder Zahl eingetragen, die diese Ressource eindeutig identifiziert. Beispiele für vernetzte Ressourcen sind URLs.
Source	In diesem Element wird – falls nötig – das gedruckte oder elektronische Werk, aus dem diese Ressource stammt, eingetragen.

Language	Hier werden die Sprache(n) des intellektuellen Inhalts der Ressource vermerkt.
Relation	Die Angabe in diesem Feld ermöglicht es, Verbindungen zwischen verschiedenen Ressourcen darzustellen, die einen formalen Bezug zueinander haben, aber als eigenständige Ressourcen existieren. Beispiele sind Bilder in einem Dokument, Kapitel eines Buches oder Einzelstücke einer Sammlung.
Coverage	Hier werden Angaben zur räumlichen Bestimmung (z.B. geographische Koordinaten) und zeitlichen Gültigkeit eingetragen, die die Ressource charakterisieren.
Rights	Vorgesehen für den Inhalt dieses Elements ist ein Link (z.B. über eine URL) zu einem Urhebervermerk, ein »Rights-Management«-Vermerk über die rechtlichen Bedingungen oder ggf. ein Link zu einem Server, der solche Informationen dynamisch erzeugt.

Tabelle 3.6 — Die 15 Kernelemente des Dublin Core Standards (nach [RF96])

Dublin Core stellt zunächst nur einen Satz von Elementen dar, die zur Definition von Metadaten dienen. Wie diese Metadaten im Einzelnen repräsentiert werden, wird erst auf einer zweiten Stufe geregelt. So kann die Beschreibung im head-Block einer HTML-Datei erfolgen (vgl. [Kun99]). Dabei werden die in der obigen Tabelle aufgeführten Elemente mit dem prefix »DC.« ergänzt:

```
<html>
  <head>
    <title>A Dirge</title>
    <link rel="schema.DC"
        href="http://purl.org/DC/elements/1.0/">
    <meta name="DC.Title"    content="A␣Dirge">
    <meta name="DC.Creator"  content="Shelley,␣Percy␣
        Bysshe">
    <meta name="DC.Type"     content="poem">
    <meta name="DC.Date"     content="1820">
    <meta name="DC.Format"   content="text/html">
    <meta name="DC.Language" content="en">
  </head>
  <body>
    <pre>
      Rough wind, that moanest loud
```

```
            Grief too sad for song;
            Wild wind, when sullen cloud
            Knells all the night long;
            Sad storm, whose tears are vain,
            Bare woods, whose branches strain,
            Deep caves and dreary main, –
            Wail, for the world's wrong!
        </pre>
    </body>
</html>
```

In Abschnitt 3.7.3 ab Seite 129 werden wir noch eine weitere Möglichkeit der Implementierung von Dublin Core in RDF kennen lernen.

3.7.2 Semantic Web

Semantic Web

Der Begriff des Semantic Web entstammt dem Vorschlag von Tim Berners-Lee [BL04] [BL98], das heute bekannte Internet bzw. World Wide Web (WWW) derart zu erweitern, dass die darin verfügbare Information nicht nur menschlichen Lesern verständlich wird, sondern vielmehr auch für Maschinen verstehbar wird. Wenn es gelänge, so Tim Berners-Lees Vision, dass Maschinen die Bedeutung, den Inhalt und den Kontext von im World Wide Web verfügbaren Objekten verstehen lernen, dann könnten jene mit diesen Inhalten arbeiten und bspw. selbstständig weitergehende Information suchen oder über den Inhalt eines Dokumentes mit anderen Maschinen in eine Kommunikation treten.

Eine direkte Folge der Forderung nach maschineller Verstehbarkeit von Dokumentinhalten ist die Anreicherung von Inhalten mit Metadaten. Diese übernehmen die Aufgabe, das in den Inhalten enthaltene Wissen in eine maschinenverständliche Form zu bringen und gleichzeitig damit die Möglichkeit für Maschinen zu eröffnen, aus dieser Information weitere Rückschlüsse zu ziehen.

Semantic Web, Schichtenarchitektur

Für die Realisierung des Semantic Web schlägt Berners-Lee die in Abbildung 3.4 dargestellte Schichtenarchitektur vor. In den einzelnen Schichten kommen dabei bewährte Methoden und standardisierte Technologien zum Einsatz, die im Folgenden kurz vorgestellt werden. Die Standardisierung der Architektur des Semantic Web sowie der dabei zum Einsatz kommenden Methoden und Technologien erfolgt zumeist durch das *World Wide Web Consortium*[2] (W3C).

[2] *http://www.w3.org/*, letzter Abruf: 26.10.2006

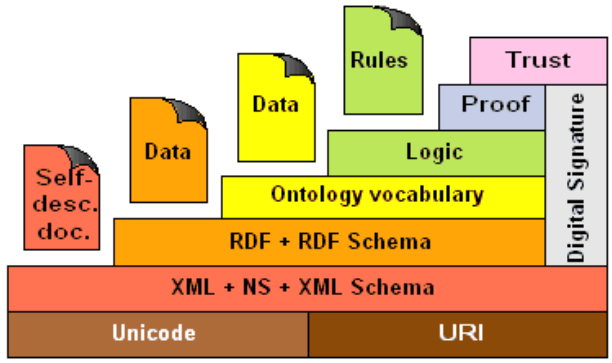


Abbildung 3.4 — Architektur des Semantic Web (aus Berners-Lee [BL00])

Unicode

Eine der grundlegenden Technologien für die Realisierung des Semantic Web stellt der Einsatz von *Unicode* für die Repräsentation von Daten dar. Unicode erlaubt nicht nur die Abbildung westlicher Alphabete und der in diesen enthaltenen Schriftzeichen sowie von Zahlen, sondern schließt auch die zahlreichen und verschiedenartigen asiatischen Alphabete mit ihren umfangreichen Schriftzeichen mit ein. Eine weitere Basistechnologie stellt die Verwendung von *Uniform Resource Identifiern* (URI) [BLFM98] dar, auf die im weiteren Verlauf noch näher eingegangen werden soll.

Uniform Resource Identifier

Resource Description Framework

Extensible Markup Language

Für die Repräsentation der Metadaten kommt das *Resource Description Framework* (RDF) [JJC04] zum Einsatz. Dieses wiederum nutzt die *Extensible Markup Language* (XML) [W3C] [BPSM+04] als Repräsentationsmedium. Beide Technologien sowie ihre Unterschiede in Bezug auf ihre Anwendung für das Semantic Web werden in den folgenden Abschnitten noch ausführlich dargelegt.

Ontologie

Über diesen Schichten liegt die Ebene der *Ontologie*. Ontologien sollen dabei helfen, Beziehungen zwischen den einzelnen Elementen herzustellen. Daneben erlaubt die darüber liegende Schicht die Definition von Regeln zwischen den Elementen der Ontologie bzw. auch für deren Beziehungen. Für die Beschreibung von Ontologien wurde vom W3C die *Web Ontology Language* (kurz OWL) vorgeschlagen und standardisiert [MH04] [DS04] [PSHH04].

Web Ontology Language

Für die weiteren Schichten des Semantic Web sind bisher nur abstrakte Vorschläge oder Forschungsansätze existent. So soll die Logik-Schicht

maschinelles Schließen auf den Metadaten, deren Beziehungen und Regeln ermöglichen. Die beiden weiteren Schichten *Trust* und *Proof* tragen dem Charakter des World Wide Web Rechnung: Es steht zwar ein mannigfaltiges Angebot an Information bereit, dieser Information sollte aber je nach deren Quelle mehr oder weniger Vertrauen entgegengebracht werden. So stellen zwar Verschlüsselung und digitale Signaturen eine bewährte Möglichkeit dar, die Quelle wie auch die Information selbst zu sichern, allerdings ist deren Anwendung im World Wide Web nicht zwingend vorgeschrieben. Darüber hinaus sagt die Anwendung von Verschlüsselung und digitalen Signaturen nichts über die Vertrauenswürdigkeit der Quelle selbst aus.

Trust
Proof
Verschlüsselung
Signatur, digitale

3.7.3 RDF

Das *Resource Description Framework*, kurz *RDF*, stellt eine allgemeine Sprache für die Abbildung von Informationen im World Wide Web dar. Insbesondere ist RDF dafür gedacht, Metadaten über Ressourcen im Internet abzubilden. Bei den Ressourcen muss es sich nicht um Dokumente oder Bilder handeln. Vielmehr können über RDF beliebige Objekte erfasst werden, wie bspw. Waren mit ihrem Preis und ihrer Verfügbarkeit in einem Online-Shop.

Resource Description Framework

Aussagen über Ressourcen

RDF bietet die Möglichkeit, Aussagen über Ressourcen zu machen. Sehen wir uns dazu die folgende Aussage an:

> Die Webseite **http://ai1.inf.uni-bayreuth.de/** hat einen **Autor**, dessen Name **Andreas Henrich** ist.

In dieser Aussage verwenden wir einen *Uniform Resource Locator* (URL), um eine Webseite bzw. Ressource eindeutig zu identifizieren. Weiterhin verwenden wir das Wort »Autor«, um eine Eigenschaft der Ressource anzugeben, der wir als Wert »Andreas Henrich« zuordnen.

RDF verwendet nun eine spezielle Terminologie für die einzelnen Bestandteile einer Aussage. So wird die beschriebene Ressource, also in unserem Beispiel die Webseite, als *Subjekt* (*subject*) bezeichnet. Der Teil der

Aussage, der eine Eigenschaft oder Charakteristik der Ressource kennzeichnet, wird als *Prädikat* (*predicate*) und der Teil, der den Wert dieser Eigenschaft beschreibt, als *Objekt* (*object*) bezeichnet.

Um nun eine natürlichsprachliche Aussage in ein maschinenverständliches Format überführen zu können sind zwei Anforderungen nötig:

1. Wir benötigen maschinenverständliche Identifikatoren, die eine eindeutige und unmissverständliche Kennzeichnung von Subjekt, Prädikat und Objekt einer Aussage ermöglichen.

2. Wir müssen ein maschinenlesbares Format zur Darstellung der Identifikatoren einsetzen, die einen einfachen Austausch zwischen verschiedenen Systemen ermöglichen.

Für beide Anforderungen existieren bereits bewährte Mechanismen, die wir im Folgenden betrachten werden.

Uniform Resource Identifier (URI)

In RDF treffen wir Aussagen über Ressourcen (Subjekte), die wir identifizieren müssen. In unserem einführenden Beispiel haben wir die URL einer Webseite gewählt und damit schon einen eleganten Weg gefunden, eine Ressource – hier eine Webseite – eindeutig zu identifizieren. Doch wie sieht es mit anderen Ressourcen aus, die nicht über eine URL verfügen, bspw. wenn wir die Adresse einer Person abbilden möchten?

Uniform Resource Identifier

Die Lösung dieses Problems stellt die Verwendung eines *Uniform Resource Identifiers*, kurz *URI*, dar[3]. URIs sind sehr ähnlich zu URLs – URLs stellen eine Untermenge von URIs dar. Ein URI bezieht sich aber nicht nur auf Ressourcen, die über einen Netzwerkanschluss verfügen, vielmehr können mit URIs beliebige Objekte eindeutig gekennzeichnet werden.

In RDF werden Uniform Resource Identifier dazu verwendet, sowohl Subjekte, also die Ressourcen, als auch Prädikate und Objekte eindeutig zu kennzeichnen.

[3] siehe *http://www.ietf.org/rfc/rfc2396.txt*, letzter Abruf: 26.10.2006

RDF-Modell

RDF verwendet URIs für die Identifikation von Objekten und XML zur Repräsentation der in RDF modellierten Inhalte.

Kommen wir nun auf unser Ausgangsbeispiel zurück:

> Die Webseite **http://ai1.inf.uni-bayreuth.de/** hat einen **Autor**, dessen Name **Andreas Henrich** ist.

Als RDF-Aussage lassen sich die Bestandteile dieser Aussage folgendermaßen auf URIs abbilden:

- **Subjekt**: `http://ai1.inf.uni-bayreuth.de/`
- **Prädikat**: `http://purl.org/dc/elements/1.1/creator`
- **Objekt**: `http://ai1.inf.uni-bayreuth.de/mitarbeiter/henrich/`

Wir verwenden dabei als Prädikat eine URI, die im Rahmen der Dublin Core Initiative spezifiziert und als Standard allgemein anerkannt ist. Die Prädikate nach Dublin Core sind dabei unter *http://dublincore.org/documents/dcmi-terms/index.shtml#H2* verfügbar. Als URI für das Objekt verwenden wir die URL der Mitarbeiter-Homepage.

Die in RDF modellierten Aussagen lassen sich auch als Knoten und Kanten in einem Graphen auffassen. In dieser Notation werden die Subjekte und die Objekte durch Knoten und das Prädikat durch eine benannte und gerichtete Kante zwischen zwei Knoten dargestellt (vgl. Abbildung 3.5).

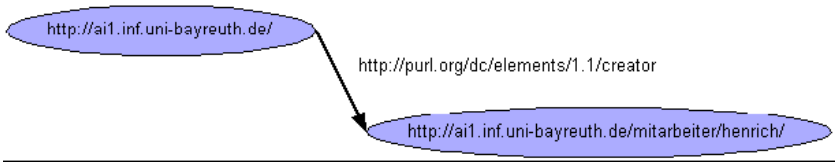


Abbildung 3.5 — RDF-Graph zum Beispiel (1)

Eine Sammlung von Aussagen wie

- Die Webseite **http://ai1.inf.uni-bayreuth.de/** hat einen **Autor**, dessen Name **Andreas Henrich** ist.
- Die Webseite **http://ai1.inf.uni-bayreuth.de/** hat ein **Erstellungsdatum** mit dem Wert **1.10.2001**.
- Die Webseite **http://ai1.inf.uni-bayreuth.de/** hat als **Sprache** den Wert **Deutsch**.

lässt sich entsprechend als Ansammlung von Knoten und Kanten darstellen (vgl. Abbildung 3.6).

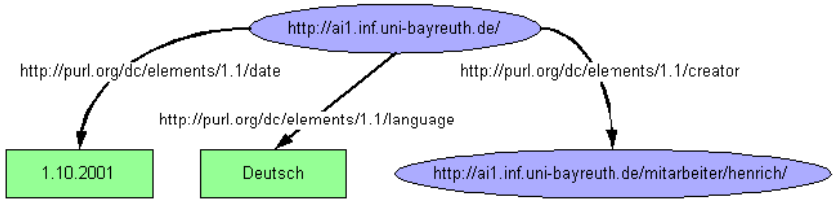


Abbildung 3.6 — RDF-Graph zum Beispiel (2)

Wie wir im obigen Beispiel sehen, können in RDF als Objekte nicht nur URIs zum Einsatz kommen. Handelt es sich um einfache Textwerte einer Eigenschaft, so können diese direkt angegeben werden – man spricht hier auch von einem *Literal*. In unserem Beispiel werden das Erstellungsdatum und die Sprache durch solche Literale dargestellt.

Literal

Die Aussagen können in RDF zudem verbunden werden. Auf diesem Weg ist es möglich Beziehungen zwischen Ressourcen abzubilden. Das folgende Beispiel in Abbildung 3.7 zeigt eine Erweiterung des obigen RDF-Graphen.

Hier werden zum Autor weitere Metadaten, nämlich die Straße und der Ort angegeben. Der Autor (creator) ist hier also gleichzeitig Objekt der einen Aussage und Subjekt der zweiten Aussage. Auf diese Art lassen sich komplexe Beziehungen zwischen verschiedenen Ressourcen abbilden.

RDF/XML Syntax Spezifikation

RDF definiert eine Syntax für die Abbildung der RDF-Graphen in XML. Die einzelnen Knoten und Kanten eines RDF-Graphen werden als Elemente, Attribute, Elementinhalte und Attributwerte in XML abgebildet.

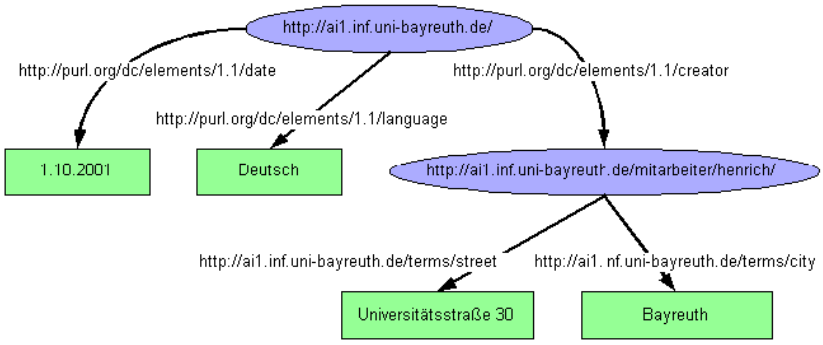


Abbildung 3.7 — RDF-Graph zum Beispiel (3)

Die URIs für Eigenschaften und Objekte werden mittels des Mechanismus der Namensräume abgebildet.

Das obige Beispiel stellt sich damit in RDF/XML wie folgt dar:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<rdf:RDF xmlns:rdf="http://www.w3c.org/1999/02/22-rdf-
    syntax-ns#" xmlns:dc="http://purl.org/dc/elements/1.1/"
    >

  <rdf:Description rdf:about="http://ai1.inf.uni-bayreuth.
      de/">
    <dc:date>1.10.2001</dc:date>
    <dc:language>Deutsch</dc:language>
    <dc:creator rdf:resource="http://ai1.inf.uni-bayreuth
        .de/mitarbeiter/Henrich/"/>
  </rdf:Description>
</rdf:RDF>
```

In diesem Beispiel verwenden wir als Prädikat ein Element aus dem Dublin Core Modell. Die Basis URI wird zu Beginn des XML-Dokuments als Namensraum definiert und dem Präfix »dc« zugeordnet. Im Folgenden kann so eindeutig auf Elemente des Dublin Core Modells referenziert werden, indem das Präfix vor die entsprechenden Elemente gesetzt wird.

Im Folgenden erweitern wir unser Beispiel um ein zusätzliches, lokal definiertes Vokabular, das wir mit *ex* benennen:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
  <rdf:RDF xmlns:rdf="http://www.w3c.org/1999/02/22-rdf-
      syntax-ns#" xmlns:dc="http://purl.org/dc/elements
```

```
        /1.1/" xmlns:ex="http://ai1.inf.uni-bayreuth.de/
        terms/">

      <rdf:Description rdf:about="http://ai1.inf.uni-
          bayreuth.de/">
        <dc:date>1.10.2001</dc:date>
        <dc:language>Deutsch</dc:language>
        <dc:creator rdf:resource="http://ai1.inf.uni-
            bayreuth.de/mitarbeiter/Henrich/">
          <ex:street>Universitätsstraße 30</ex:street>
          <ex:city>Bayreuth</ex:city>
        </dc:creator>
      </rdf:Description>
    </rdf:RDF>
```

RDF-Schema

RDF-Schema

Das Resource Description Framework definiert ein einfaches Datenmodell, um Eigenschaften von Ressourcen und Beziehungen zwischen Ressourcen abzubilden. Für dieses Ressourcenmodell lässt sich mittels RDF-Schema (RDFS) ein Typsystem erstellen. RDF-Schema bietet hierzu einen Mechanismus, um Klassen und ihre Eigenschaften zu definieren.

Klassen

Eine Klasse (*class*) im RDF-Schema entspricht dem Verständnis einer Klasse in objektorientierten Programmiersprachen. RDF Klassen können jedoch beliebige Objekte repräsentieren, wie bspw. Webseiten, Personen, Dokumenttypen, Dinge oder auch abstrakte Konzepte.

Ein RDF-Schema wird – wie wir gleich sehen werden – in RDF erstellt. RDF unterstützt bei der Definition von Klassen wie auch von Eigenschaften das Konzept der Vererbung.

Im Folgenden sind als RDF-Schema drei Klassen definiert: *Automobil*, *Pkw* und *Sportwagen*. Unsere Basisklasse *Automobil* wird von der RDF-Schema-Basisklasse *Resource* abgeleitet. Die Klasse *Pkw* wird von der Klasse *Automobil* und die Klasse *Sportwagen* wiederum von der Klasse *Pkw* abgeleitet.

```
<rdf:RDF xmlns:rdf="http://www.w3c.org/1999/02/22-rdf-
    syntax-ns#" xmlns:rdfs="http://www.w3c.org/2000/01/rdf-
    schema#">
```

```
    <rdf:Description rdf:ID="Automobil">
       <rdf:type rdf:resource="http://www.w3c.org/2000/01/
           rdf-schema#Class"/>
       <rdfs:subClassOf rdf:resource="http://www.w3c.org
           /2000/01/rdf-schema#Resource"/>
    </rdf:Description>

    <rdf:Description rdf:ID="Pkw">
       <rdf:type rdf:resource="http://www.w3c.org/2000/01/
           rdf-schema#Class"/>
       <rdfs:subClassOf rdf:resource="#Automobil"/>
    </rdf:Description>

    <rdf:Description rdf:ID="Sportwagen">
       <rdf:type rdf:resource="http://www.w3c.org/2000/01/
           rdf-schema#Class"/>
       <rdfs:subClassOf rdf:resource="#Pkw"/>
    </rdf:Description>
</rdf:RDF>
```

Eigenschaften

Eigenschaften (*properties*) zu den einzelnen Klassen können nun wie folgt definiert werden:

```
<rdf:RDF xmlns:rdf="http://www.w3c.org/1999/02/22-rdf-
    syntax-ns#" xmlns:rdfs="http://www.w3c.org/2000/01/rdf-
    schema#">

    <rdf:Description rdf:ID="zugelassenAuf">
       <rdf:type rdf:resource="http://www.w3c.org
           /1999/02/22-rdf-syntax-ns#Property"/>
       <rdfs:domain rdf:resource="#Automobil"/>
       <rdfs:range rdf:resource="http://ai1.inf.uni-bayreuth
           .de/classes#Person"/>
    </rdf:Description>

    <rdf:Description rdf:ID="MaxAnzahlPassagiere">
       <rdf:type rdf:resource="http://www.w3c.org
           /1999/02/22-rdf-syntax-ns#Property"/>
       <rdfs:domain rdf:resource="#Pkw"/>
       <rdfs:range rdf:resource="http://www.w3c.org/2001/
           XMLSchema#integer"/>
    </rdf:Description>
```

```
</rdf:RDF>
```

Da Eigenschaften eine durch eine Kante repräsentierte Beziehung abbilden, muss ihnen ein Ausgangstyp (*domain*) und ein Zieltyp (*range*) zugeordnet werden. Während der Ausgangstyp eine Klasse sein muss, kann als Zieltyp sowohl ein Literal als auch eine Klasse angegeben werden. In unserem Beispiel wird als Zieltyp für die Eigenschaft *zugelassenAuf* eine selbst definierte Klasse *Person* angegeben, während die Eigenschaft *MaxAnzahlPassagiere* ein Integer-Wert sein muss.

Im Folgenden ist nun ein exemplarisches RDF-Dokument aufgeführt, welches das soeben definierte RDF-Schema einsetzt.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3c.org/1999/02/22-rdf-
    syntax-ns#" xmlns:a="http://ai1.inf.uni-bayreuth.de/
    schemas/autos#">

  <a:Pkw rdf:ID="AndreasHenrichsPkw">
    <a:zugelassenAuf rdf:resource="http://ai1.inf.uni-
        bayreuth.de/mitarbeiter/henrich/"/>
    <a:MaxAnzahlPassagiere rdf:datatype="http://www.w3c.
        org/2001/XMLSchema#integer">5</
        a:MaxAnzahlPassagiere>
  </a:Pkw>
</rdf:RDF>
```

In diesem Beispiel verwenden wir für die Identifikation des Subjekts nicht wie gewohnt *rdf:about*, sondern *rdf:ID*. Die Verwendung des Attributs *ID* statt *about* kennzeichnet eine neue Ressource, die wir hier beschreiben. Wir müssen in diesem Fall dafür Sorge tragen, dass die ID eindeutig ist. Andere RDF-Dokumente können auf diese Ressource dann verweisen durch

- den relativen URI »*#AndreasHenrichsPkw*« oder
- einen absoluten URI, der sich aus dem Ablageort des RDF-Dokuments ergänzt um »*#AndreasHenrichsPkw*«, also bspw. »*http://ai1.inf.uni-bayreuth.de/parkplatz/#AndreasHenrichsPkw*« zusammensetzt.

3.8 Zusammenfassung

In diesem Kapitel haben wir uns mit Problemen der Grund- und Stammformreduktion, der Mehrwortgruppenidentifikation und der Synonymverwaltung beschäftigt. Fast alle behandelten Techniken sind dabei sprachabhängig. Im Deutschen müssen andere Techniken angewendet werden als im Englischen oder im Französischen.

Auch die Wirkung der einzelnen Ansätze kann von Sprache zu Sprache und auch von Themengebiet zu Themengebiet stark variieren. So kann für das Englische z.B. keine allgemeine Aussage getroffen werden, ob eine Stammformreduktion grundsätzlich sinnvoll ist oder nicht. Trotzdem – oder gerade deswegen – ist es bei der Konzeption eines IR-Systems wichtig eine klare Konzeption für die verschiedenen besprochenen Problemaspekte zu verfolgen und sie nach Möglichkeit mit Hilfe von Recall/Precision-Tests zu hinterfragen.

Abschließend haben wir mit der Auszeichnung durch zusätzliche Metadaten, dem Ansatz des Semantic Web und dem Resource Description Framework Ansätze kennen gelernt, die versuchen, die Vagheit der natürlichen Sprache zu umgehen.

4 Einfache IR-Modelle und ihre Implementierung

In diesem Kapitel wollen wir vier einfachere Modelle des Information Retrieval kennen lernen. Dies sind die *einfache Wortsuche*, das *Boolesche Retrieval*, das *Coordination-Level Match* sowie das *Fuzzy-Retrieval*. Zu diesen Modellen werden wir jeweils auch mögliche Implementierungsansätze beschreiben.

Dieses Kapitel beinhaltet folgende Unterkapitel:

4.1 Pattern-Matching
4.2 Wortsuche
4.3 Boolesches Retrieval
4.4 Coordination-Level-Match
4.5 Fuzzy-Set Modell
4.6 Zusammenfassung

Bevor wir die vier Modelle betrachten, erscheint es zunächst sinnvoll zu beschreiben, was ein IR-Modell festlegen sollte.

Information Retrieval Modelle

Modell

Im Bereich des Information Retrieval muss ein Modell folgende implizite wie auch explizite Aspekte abdecken:

- Es muss die **Annahmen** angeben, die dem Modell zugrunde liegen. Ein Beispiel für eine solche Annahme könnte sein, dass es für die Relevanz eines Dokumentes keine Rolle spielt, ob ein Wort einmal oder zehnmal in dem Dokument vorkommt. Das Boolesche Retrieval beruht z.B. auf dieser Annahme, während das Vektorraummodell die entgegengesetzte Annahme macht.

- Des Weiteren muss ein IR-Modell **Repräsentationen** für die Dokumente und die Anfragen vorgeben.
- Schließlich muss ein Modell eine **Retrieval-Funktion** definieren, die für ein Paar aus Anfrage und Dokument – bzw. für ihre Repräsentationen – eine Relevanzkennzahl berechnet, die üblicherweise als Näherung der Relevanzwahrscheinlichkeit angesehen wird.

4.1 Pattern-Matching

Pattern-Matching

Obwohl es sich bei den klassischen Pattern-Matching-Algorithmen nicht um Information Retrieval Modelle im eigentlichen Sinne handelt, sollen diese jedoch kurz angesprochen werden, weil sie in einigen IR-Modellen als Teilkomponenten verwendet werden und in gewisser Weise auch ein sehr einfaches IR-System bilden.

Die Aufgabenstellung des Pattern-Matching kann wie folgt umrissen werden: Gegeben ist eine Zeichenkette nach der in einem längeren Text gesucht werden soll. Diese Zeichenkette wird dabei auch als Suchpattern oder Suchmuster bezeichnet.

Suchmuster

Die einfachste Möglichkeit besteht nun darin, dass man das Suchmuster von links nach rechts sequenziell mit dem zu durchsuchenden Text vergleicht. Genauer formuliert wird das Suchpattern, beginnend mit seinem ersten Zeichen, über den zu durchsuchenden Text gelegt, so dass das erste Zeichen des Patterns mit dem ersten Zeichen des Textes korrespondiert. Treten beim Vergleich der Zeichen des Patterns mit den entsprechenden Zeichen des Textes nur Übereinstimmungen auf, so hat man einen Treffer – ein Match – gefunden. Gibt es an einer Stelle einen Unterschied, liegt an dieser Position des Textes kein Treffer vor. Der Vergleich zwischen Pattern und Text kann dabei offensichtlich abgebrochen werden, sobald die erste Ungleichheit aufgetreten ist. Bei einem solchen »Mismatch« wird das Pattern im Verhältnis zum Text um eine Position weiter gerückt und es erfolgt ein erneuter Vergleich.

Wenn wir die Länge des Patterns mit m und die Länge des Textes mit n bezeichnen, dann hat der beschriebene Algorithmus im schlechtesten Fall offensichtlich eine Laufzeit von $O(m \cdot n)$. Im Durchschnitt wird man allerdings sehr häufig bereits beim ersten oder zweiten Zeichen eine fehlende Übereinstimmung feststellen, so dass sich im Durchschnitt $O(n)$ ergibt.

Es gibt nun in der Literatur zahlreiche optimierte Varianten des Pattern-Matching (vgl. z.B. [OW02]). Hierzu zählen der Boyer-Moore-Horspool-Sunday-Algorithmus [Sun90] oder der Knuth-Morris-Pratt-Algorithmus [KJP77]. Diese Algorithmen sind zum Teil signifikant schneller als der oben beschriebene einfache Algorithmus und empfehlen sich für den praktischen Einsatz. Erhalten bleibt dabei allerdings die lineare Laufzeit $O(n)$, die dazu führt, dass ein Einsatz derartiger Pattern-Matching-Algorithmen bei großen Dokumentkollektionen nicht möglich ist.

4.2 Wortsuche

Wortsuche

Die Wortsuche stellt eines der einfachsten Modelle des Information Retrieval dar. Gesucht werden bei diesem Modell alle Dokumente, die eine bestimmte Zeichenkette als Wort enthalten. Wir können als Anforderungen definieren:

- Gegeben ist ein bestimmtes Wort W.
- Gesucht sind alle Dokumente, die W enthalten.

Eine mögliche Erweiterung des Modells wäre die Verwendung einer Stamm- oder Grundformreduktion. Dabei würden alle Dokumente gesucht, die ein Wort enthalten, dessen Stamm- oder Grundform der Stamm- oder Grundform von W entspricht.

Eine weitere Möglichkeit der Erweiterung stellt die Ausweitung auf Wortmengen $W_1, \ldots, W_k$ dar. Dies bedeutet, dass die Dokumente gesucht werden, die alle gegebenen Worte enthalten. Es findet also eine UND-Verknüpfung über die gesuchten Worte statt. Diese Erweiterung kann damit als erster Schritt in Richtung des Booleschen Retrieval verstanden werden, das wir im Abschnitt 4.3 betrachten werden.

Boolesches Retrieval

Ein Unterschied zwischen der Wortsuche und klassischen, exakten Pattern-Matching-Verfahren, wie z.B. Boyer-Moore oder Knuth-Morris-Pratt (vgl. z.B.: [BY92]), besteht darin, dass wir bei der Wortsuche nicht prüfen, ob ein gegebenes Pattern – bzw. Suchwort – in *einem* Text enthalten ist, sondern dass wir nun in einer Menge von Dokumenten nach den Dokumenten suchen, die das Suchwort enthalten. Ferner beziehen wir uns nun auf Wörter; es werden nicht alle Dokumente gefunden, die das Suchwort als Zeichenkette enthalten, sondern nur die, die es als eigenständiges Wort enthalten.

Pattern-Matching

Implementierungsansätze für die Wortsuche

Ein mögliches Vorgehen zur Realisierung der Wortsuche wäre natürlich, dass man alle Dokumente nacheinander betrachtet und für jedes Dokument mit einem Pattern-Matching Algorithmus prüft, ob dieses Dokument das Suchwort enthält. Dabei könnte ein nachgeschalteter Test prüfen, ob das Suchwort im Dokument tatsächlich als eigenständiges Wort enthalten ist.

Das Problem dieses Vorgehens liegt darin, dass der Zeitaufwand für die Suche linear mit dem Umfang der Dokumentenkollektion anwächst. Wir wollen daher nun zwei Implementierungsansätze betrachten, die dieses Problem abmildern: *invertierte Listen* und *Signaturen*. Wie wir später sehen werden, können diese Datenstrukturen auch zur Implementierung zahlreicher anderer IR-Modelle eingesetzt werden.

Indexstruktur

Der Lösungsansatz, um die Suche effizienter zu gestalten, basiert in beiden Fällen darauf, vorab eine so genannte *Indexstruktur* für die Dokumentenkollektion zu erstellen, in der dann wesentlich effizienter gesucht werden kann als in der Dokumentenkollektion selbst.

4.2.1 Invertierte Listen

Invertierte Listen

Die Idee hinter invertierten Listen besteht darin, dass bei einer normalen Darstellung Dokumente mit den in diesen enthaltenen Wörtern abgespeichert werden. Das heißt man greift zuerst auf ein Dokument und dann auf die darin abgelegten Wörter zu.

Da die Suche aber nicht von Dokumenten, sondern von Wörtern ausgeht, invertiert man nun diese Verwaltung von Dokumenten und Wörtern. Bei einer invertierten Liste speichert man zu einem Wort alle Dokumente ab, die dieses Wort enthalten. Etwas konkreter wird zu jedem Wort eine Liste der Dokumente verwaltet, die dieses Wort enthalten. Abbildung 4.1 veranschaulicht dies.

Auf der linken Seite der Abbildung ist – stark vereinfacht – die normale Verwaltung von Dokumenten mit den in diesen enthaltenen Wörtern dargestellt. Unsere Beispielkollektion besteht aus den Dokumenten D_1 bis D_4.

Auf der rechten Seite der Abbildung werden die vier Beispieldokumente in einer invertierten Liste verwaltet. Dabei wird von einer Tabelle ausgegangen, welche die Vereinigungsmenge aller in den Dokumenten

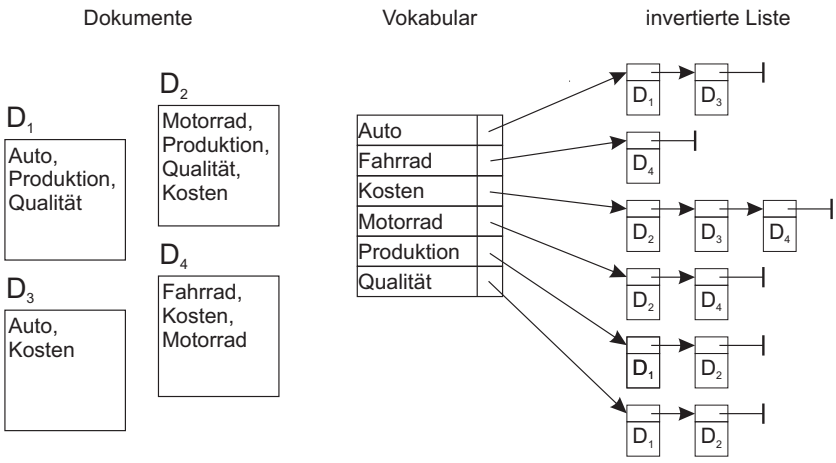


Abbildung 4.1 — Aufbau einer invertierten Liste

vorkommenden Wörter beinhaltet. Jeder Tabelleneintrag besteht aus einem Wort und einem Verweis auf eine Liste, die Referenzen auf die Dokumente beinhaltet, welche dieses Wort enthalten. Die Elemente dieser Dokumentenliste können nach bestimmten Kriterien sortiert werden, in Abbildung 4.1 nach den Dokumenten-IDs.

An dieser Stelle sei noch angemerkt, dass der Begriff »invertierte Liste« im Folgenden sowohl für die gesamte auf der rechten Seite von Abbildung 4.1 dargestellte Datenstruktur verwendet wird – also für die Tabelle mit dem Vokabular und alle Listen zusammen – als auch für eine einzelne Liste innerhalb dieser Datenstruktur. Die Bedeutung sollte aber jeweils aus dem Kontext klar sein.

Abbildung der Wortsuche auf invertierte Listen

Wir betrachten hier die drei bereits erwähnten Varianten der Wortsuche.

Suche nach allen Dokumenten, die exakt das Suchwort enthalten

Eine solche einfache Wortsuche lässt sich auf Basis von invertierten Listen sehr effizient realisieren. Man muss lediglich die invertierte Liste zum vorgegebenen Suchwort betrachten und alle in dieser Liste vorkommenden Dokument-IDs in das Ergebnis aufnehmen.

Gleicher Wortstamm

Will man nicht nur nach einem konkreten Wort, sondern nach Vorkommen des gleichen Wortstamms suchen, so muss man wie folgt vorgehen:

1. Zunächst bildet man zu dem gegebenen Suchwort den Wortstamm.
2. Nun durchläuft man die Tabelle mit den Wörtern des Vokabulars und untersucht für jedes Wort, ob sein Wortstamm identisch mit dem Wortstamm des Suchwortes ist. Ist dies der Fall, so fügt man die entsprechende invertierte Liste einer anfangs leeren temporären Listenmenge hinzu.
3. Über alle Listen in der temporären Listenmenge – dies sind alle Listen zu Wörtern, die den gleichen Wortstamm haben wie das Suchwort – bildet man nun die Vereinigungsmenge der Dokument-IDs.
4. Die Dokumente in dieser Vereinigungsmenge bilden das Ergebnis der Anfrage.

Vereinigung mehrerer Listen

Zur Bestimmung der Vereinigungsmenge über mehrere Listen ist dabei die Sortierung der Listen nach Dokument-IDs nützlich. Man durchläuft dazu die Listen parallel und nimmt jede Dokument-ID, die in mindestens einer Liste auftritt, in die Vereinigungsmenge auf. Konkret kann diese Vereinigung mehrerer Listen wie folgt beschrieben werden:

1. Initialisiere alle Listen, über die die »Vereinigungsliste« gebildet werden soll, auf ihr erstes Element.
2. Betrachte unter allen Listen in jedem Schritt nur die Listen, für die das aktuell betrachtete Element auf das Dokument mit der kleinsten noch nicht betrachteten Dokument-ID verweist.
   - Nimm dieses Dokument in die anfangs leere Ergebnis-Liste auf.
   - Betrachte in den Listen, in denen das betrachtete Element auf das Dokument mit der kleinsten Dokument-ID verweist, das nächste Element.
3. Sofern in mindestens einer Liste noch ein Element zu betrachten ist, fahre mit Schritt 2 fort.

Suche nach einer Wortmenge

Schließlich können auch Anfragen mit mehreren Suchwörtern, die alle in einem gesuchten Dokument enthalten sein müssen (UND-Verknüpfung), effizient bearbeitet werden. Dazu werden nur die Dokumente in das Ergebnis aufgenommen, die in allen den Suchwörtern zugeordneten invertierten Listen auftreten. Man bildet also die Schnittmenge über die in den Listen zu den Suchwörtern enthaltenen Dokument-IDs.

UND-Verknüpfung

Zur Schnittmengenbildung ist dabei ebenfalls die Sortierung der Listen nach Dokument-IDs nützlich. Man durchläuft die Listen parallel und nimmt immer dann eine Dokument-ID in die Schnittmenge auf, wenn sie in allen invertierten Listen enthalten ist. Konkret:

Schnittmenge mehrerer Listen

1. Initialisiere alle Listen, über die die »Schnittliste« gebildet werden soll, auf ihr erstes Element.
2. Verweist das aktuell betrachtete Element in allen Listen auf das gleiche Dokument?
   - Wenn ja: Nimm dieses Dokument in die anfangs leere Ergebnis-Liste auf und betrachte in allen Listen das nächste Element.
   - Wenn nein: Betrachte in der Liste, in der das betrachtete Element auf das Dokument mit der kleinsten Dokument-ID verweist, das nächste Element.
3. Sofern in allen Listen noch ein Element zu betrachten ist, fahre mit Schritt 2 fort.

Einleitende Überlegungen zur Implementierung invertierter Listen

Da invertierte Listen auch bei der Implementierung zahlreicher anderer IR-Modelle zum Einsatz kommen, wollen wir nun einen genaueren Blick auf diese Datenstruktur werfen. Zunächst ist dabei der zusätzliche Speicherplatz zu betrachten, den invertierte Listen erfordern.

Sehen wir uns dazu ein Beispiel aus der TREC-Kollektion an.

TREC-Kollektion

Beispiel

Die TREC-Kollektion enthält unter anderem Artikel der *Financial Times*. Der Umfang dieser Dokumentenkollektion beträgt 564 MByte. Dabei enthält die Kollektion 210.158 Artikel, die im Mittel ca. 150 unterschiedliche Wörter enthalten.

Platzbedarf

Geht man davon aus, dass eine Referenz auf ein Dokument, also eine Dokument-ID, in der invertierten Liste 4 Byte benötigt, so belegt die Speicherstruktur $210.158 \cdot 150 \cdot 4$ Byte $= 126.094.800$ Byte $= 120$MByte. Dabei ist noch nicht der Speicherplatz mit eingerechnet, den bspw. die Begriffe selbst oder weitere Verwaltungsinformationen belegen.

Wir können aus dieser groben Rechnung bereits erkennen, dass eine invertierte Liste leicht mehr als 20% des Umfangs der durch sie indexierten Dokumentenkollektion erreichen kann. Werden in den Listen noch weitere Informationen wie die Vorkommenshäufigkeit oder der Vorkommensort der einzelnen Begriffe mit gespeichert, so kann der Speicherplatzbedarf für die invertierten Listen den Speicherplatzbedarf der eigentlichen Dokumentenkollektion sogar übertreffen.

Hintergrundspeicher

Dies führt unmittelbar zu der Einsicht, dass invertierte Listen nicht vollständig im Hauptspeicher verwaltet werden können. Vielmehr müssen Sie auf dem Hintergrundspeicher – typischerweise einer Festplatte – abgelegt werden. Ein weiterer Grund für die Verwaltung auf dem Hintergrundspeicher ist die Tatsache, dass invertierte Listen persistent gespeichert werden müssen.

Zipf's Law

Zipf's Law

Ausgangspunkt für die weiteren Überlegungen wie eine geeignete Speicherstruktur für invertierte Listen aussehen könnte, ist zunächst *Zipf's Law*.

Wenn man Texte betrachtet, kann man folgende Feststellungen treffen:

- Einige Wörter kommen sehr häufig vor. Für englische Texte wurde bspw. festgestellt, dass die beiden am häufigsten vorkommenden Wörter mehr als 10% aller Vorkommen stellen können. Die sechs häufigsten Wörter können 20% und die 50 häufigsten Wörter 50% aller Vorkommen stellen. Das heißt, die Hälfte der in Texten vorkommenden Wörter besteht aus nur 50 verschiedenen Wörtern.

- Sehr viele Wörter kommen dagegen sehr selten vor.
- Zudem lässt sich feststellen, dass, wenn man die Wörter nach ihrer Vorkommenshäufigkeit sortiert, das Produkt aus der Rangzahl für die Häufigkeit und der Vorkommenszahl praktisch konstant ist!

  Für einen gedachten Textkorpus könnte dies bspw. bedeuten, dass das am häufigsten vorkommende Wort ca. 7 Millionen Mal vorkommt, während das am zweithäufigsten vorkommende Wort noch 3,5 Millionen Mal und das am dritthäufigsten vorkommende Wort nur noch 2,3 Millionen Mal vorkommt.

Das Zipf'sche Gesetz adressiert diesen dritten Aspekt:

> Für einen repräsentativen Textcorpus C bezeichne $W(C)$ die Menge der Wörter, die in C vorkommen, und $h(w)$ die Häufigkeit mit der ein Wort $w \in W(C)$ in dem Korpus vorkommt. $r(w)$ bezeichne den Rangplatz von $w \in W(C)$, wenn die Wörter nach abfallender Häufigkeit sortiert werden.
>
> Dann gilt: $r(w) \cdot h(w) \approx c = konstant$, $\forall w \in W(C)$

Ein Beispiel zu Zipf's Law

In Tabelle 4.1 sind die Ergebniswerte aus dem *Brown und Lob Textkorpus* abgebildet.

In der ersten Spalte ist für jeden Term dessen Rang, in der zweiten Spalte dessen Vorkommensanzahl im gesamten Korpus und in der dritten Spalte das Produkt aus Rangplatz und Vorkommensanzahl dividiert durch 100.000 dargestellt.

Wie wir sehen, ist das Produkt aus Rang und Vorkommensanzahl zwar nicht im mathematischen Sinne, aber zumindest im linguistischen Sinne eine Konstante. Der Wert bewegt sich zwischen einem Minimum von 1,24982 und einem Maximum von 2,55618. Der Mittelwert liegt ungefähr bei 1,7.

Schlussfolgerungen für die Verwaltung von invertierten Listen

Als Schlussfolgerung aus dem Zipf'schen Gesetz lässt sich nun für invertierte Listen ableiten, dass

Rang $r(w)$	**Anzahl** $h(w)$	$\frac{r(w) \cdot h(w)}{100.000}$	**Wort bzw. Term**
1	138.323	1,38323	the
2	72.159	1,4432	of
3	56.750	1,7025	and
4	52.941	2,1176	to
5	46.523	2,3262	a
6	42.603	2,5562	in
7	22.177	1,5524	that
...	...	...	...
2804	73	2,0476	destroy
2805	73	2,0476	determination
...	...	...	...
12032	11	1,3235	would-be
12033	11	1,3236	yachting
12034	11	1,3237	yell

Tabelle 4.1 — Ergebnisse für den *Brown und Lob-Textkorpus*

wenige lange Listen

- es zu einigen wenigen häufig vorkommenden Wörtern sehr lange Listen geben wird und

viele kurze Listen

- auf der anderen Seite zu sehr vielen eher selten vorkommenden Wörtern die Listen eher kurz sein werden.

Dabei ist natürlich zu berücksichtigen, dass es sich bei den am häufigsten vorkommenden Wörtern i.d.R. um Stoppwörter handeln wird, zu denen ohnehin keine Liste verwaltet wird. Auch werden z.B. nach den Regeln von Crouch die sehr seltenen Wörter ebenfalls nicht betrachtet (vgl. hierzu Abschnitt 3.3). Trotzdem werden aber wenigen langen Listen zahlreiche kurze Listen gegenüberstehen.

Eine weitere Erkenntnis aus dem obigen Beispiel ist, dass bei einem Vokabularumfang von 12.034 verschiedenen Wörtern – wobei nur Begriffe betrachtet sind, die mehr als zehnmal vorkommen – auch eine Suchstruktur für die Tabelle mit den Wörtern verwendet werden muss, die einen effizienten Zugriff auf die zu einem gegebenen Wort gehörende Liste erlaubt.

Wir werden uns daher im Folgenden zunächst mit der Verwaltung der invertierten Listen selbst beschäftigen und anschließend auf die Verwal-

tung der Tabelle eingehen, die das Vokabular mit den Verweisen auf die einzelnen invertierten Listen enthält (vgl. hierzu Abbildung 4.1).

Verwaltung der invertierten Listen

Wie wir bereits festgestellt haben, müssen invertierte Listen aufgrund ihres Umfangs und der erforderlichen Persistenz auf dem Hintergrundspeicher verwaltet werden. Da auf diesen Speicher technisch immer blockweise zugegriffen wird (man spricht auch von einzelnen *Seiten*), bietet es sich an, auch bei der Verwaltung der invertierten Listen einzelne Plattenblöcke bzw. Seiten zu nutzen. Eine direkte 1:1-Zuordnung zwischen Listen und Seiten ist dabei aber nicht möglich. Geht man bspw. von einem Dateisystem mit einer Seitengröße von 4 KByte aus, so hätte eine solche 1:1-Zuordnung folgende Konsequenzen:

Seiten

- Die vielen kurzen Listen lasten eine Seite nur sehr gering aus, während
- die wenigen langen Listen sich über mehrere Seiten erstrecken müssen.

Um einer Speicherplatzverschwendung bei der Verwaltung der kurzen Listen zu begegnen, werden daher zur Ablage kurzer Listen gegebenenfalls auch Teilbereiche eines Blocks verwendet, das heißt es werden mehrere kurze Listen in einem Block zusammengefasst.

Die einem Term zugeordnete Liste kann nun mehrstufig adressiert werden:

- Zunächst kann man entweder alle Listen in einer Datei, jede Liste in einer eigenen Datei oder jeweils eine gewisse Zahl von Listen gemeinsam in einer Datei verwalten. Vor allem in den beiden letzten Fällen ist zur Adressierung einer invertierten Liste die **Dateikennung** (typischerweise der Dateiname) erforderlich.

  Dateikennung

- Jede Datei wird ihrerseits als Abfolge von Blöcken interpretiert. Eine Seite oder ein Block, in dem eine invertierte Liste liegt oder beginnt, kann daher über die **Blocknummer** adressiert werden.

  Blocknummer

- Schließlich wird bei kurzen Listen, die keinen ganzen Block für sich beanspruchen können, noch die **Anfangsadresse** und die Länge des Bereichs benötigt, der für die Liste im Block reserviert ist.

  Anfangsadresse

Die Listen für hochfrequente Terme können sich dabei auch über mehrere Blöcke erstrecken. Am Ende der einzelnen Blöcke wird dabei die Nummer des Folgeblocks vermerkt.

Verwaltung von invertierten Listen mit Blöcken

Im Folgenden ist nun exemplarisch die Verwaltung einer invertierten Liste mit Hilfe von Blöcken dargestellt. Jeder Block hat in diesem Beispiel eine Größe von 1 KByte (= 1024 Byte).

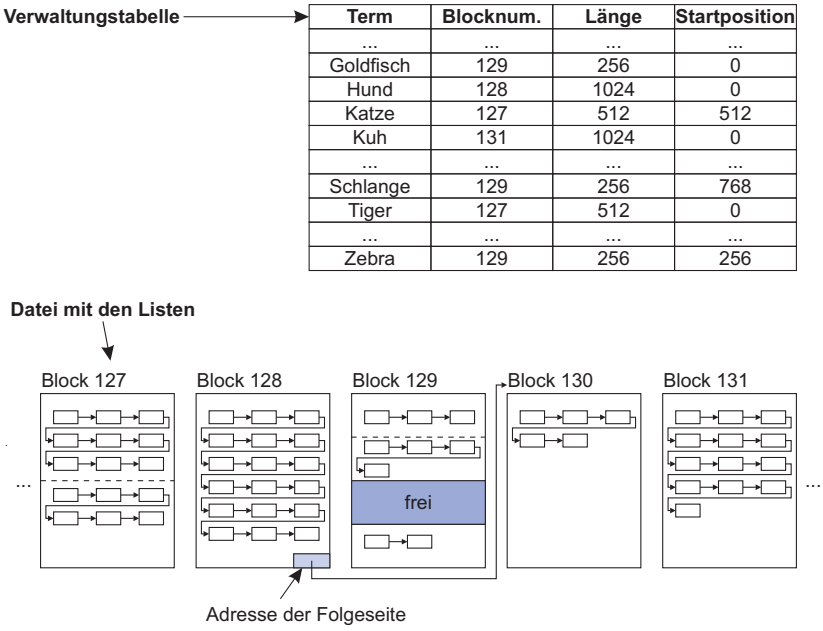


Abbildung 4.2 — Verwaltung einer invertierten Liste mit Blöcken

Im oberen Teil finden wir die Verwaltungstabelle, die zu jedem Term einen Verweis auf die Liste der Dokumente enthält, die diesen Term enthalten. In diesem Beispiel haben wir nur eine Datei für alle Dokumentlisten verwendet, so dass wir auf eine Dateikennung verzichten können. Jeder Verweis besteht aus einer Blocknummer, der Länge des für die Liste reservierten Speicherbereichs im Block sowie der Startposition der Liste im Block.

Bspw. ist die Liste für den Term »Goldfisch« im Block 129 zu finden. Sie hat eine Größe von nur 256 Byte und beginnt an der Adresse 0 des Blocks.

Der Term »Hund« verweist auf den Block 128; die Liste belegt diesen Block komplett. Da diese Liste allerdings größer ist als die Blockgröße, ist am Ende des Blocks 128 ein Verweis auf einen weiteren Block, in diesem Beispiel 130, eingefügt, in dem die Liste fortgesetzt wird.

Realisierung der Verwaltungstabelle

Die Verwaltungstabelle, in der zu jedem in den Dokumenten vorkommenden Wort ein Verweis auf die zugehörige invertierte Liste gespeichert wird, muss effizient implementiert werden, um zu einem gegebenen Suchbegriff schnell die invertierte Liste zu finden. Hierzu kann z.B. ein so genannter B-Baum (vgl. dazu [OW02] oder *Wikipedia*[1]) eingesetzt werden.

B-Baum

Zusammenfassung invertierte Listen

Wir können nun also eine invertierte Liste realisieren, indem wir für die Verwaltung der Wörter (bzw. Terme) und der zu den jeweiligen Wörtern verwalteten invertierten Liste einen B-Baum verwenden. Die invertierten Listen selbst können dann in der beschriebenen Form in einer oder mehreren Dateien abgelegt werden.

Damit ist die Suche nach den zur Bearbeitung einer Anfrage benötigten invertierten Listen ebenso effizient möglich wie das Durchlaufen der invertierten Listen.

4.2.2 Signaturen

Eine Alternative zur Implementierung der Wortsuche mit invertierten Listen bilden Signaturen. Signaturen sind dabei zunächst lediglich Bitstrings fester Länge. Einzelne Wörter oder ganze Textblöcke werden dabei auf solche Bitstrings abgebildet, weil Vergleichsoperationen auf Signaturen wesentlich effizienter implementiert werden können als auf Zeichenketten.

Signaturen

[1] *http://de.wikipedia.org/wiki/B-Baum*, letzter Abruf: 14.11.2006

Definition: Signatur

Signatur

Sei F eine natürliche Zahl. Eine *Signatur* der Länge F ist ein Bitstring $s = b_1 b_2 ... b_F$ mit $b_i \in \{0; 1\}$ für $1 \leq i \leq F$.

Hash-Funktion

Signaturen werden durch die Abbildung von Wörtern auf Bitstrings erzeugt. Diese Abbildung geschieht typischerweise mit Hilfe einer *Hash-Funktion*. Ein Beispiel für eine solche Hash-Funktion zur Berechnung der Signatur für ein l-stelliges Wort $w_1, w_2, \ldots, w_l$ ist:

```
hash := 0;
for i := 1 to l do
   hash := (hash + w_i) * 157;
end
hash := hash mod 2^F;
```

Die Berechnung des Hash-Wertes erfolgt über alle Zeichen des l-stelligen Wortes durch Summation und Multiplikation der Zwischenergebnisse mit der Primzahl 157. Die fortgesetzte Multiplikation mit 157 führt dabei dazu, dass zum einen eine hinreichend große Zahl entsteht und zum anderen eine »gute Durchmischung« erfolgt. Insbesondere sollen sich für Wörter, die sich nur geringfügig unterscheiden sicher unterschiedliche Hash-Werte ergeben. Am Ende erhalten wir durch Anwendung des Modulo-Operators einen Hash-Wert, der auf maximal F Stellen bzw. Bits begrenzt ist[2].

Eigenschaften von Signaturen

Indem man die Suchoperationen auf den Signaturen und nicht mehr auf den Texten der einzelnen Dokumente durchführt, erhält man eine Effizienzsteigerung gegenüber dem Einsatz von Pattern-Matching-Techniken. Durch den Einsatz von speziellen Datenstrukturen für die Verwaltung der Signaturen ist eine weitere Beschleunigung der Suchoperationen möglich.

Bei der Abbildung von Worten in Hash-Werte (Signaturen) entstehen allerdings i.d.R. künstliche Homonyme. Das bedeutet, dass verschiedene Wörter auf den gleichen Hash-Wert abgebildet werden und damit eine identische Signatur aufweisen.

[2] Innerhalb der for-Schleife muss dabei ggf. noch sichergestellt werden, dass es nicht zu einem Überlauf kommt. Dies ist individuell je nach verwendeter Programmiersprache sicherzustellen.

Grundlegende Idee der Signaturdateien

Signaturdateien basieren daher auf der Idee des *inexakten Filters*. Das heißt mit ihnen kann auf sehr effiziente Weise eine notwendige Bedingung überprüft werden, die die Dokumente erfüllen müssen, damit sie das gesuchte Wort enthalten können. In unserem Fall ist dies die Bedingung, dass zu dem Dokument eine Signatur ermittelt wurde, die mit der Signatur des Anfragewortes übereinstimmt.

inexakter Filter

Es kann nun vorkommen, dass einige Dokumente, welche die Testbedingung erfüllen, das zu suchende Muster trotzdem nicht enthalten. Dies ist auf den Effekt der künstlichen Homonyme zurückzuführen. Dokumente, die die Testbedingung erfüllen, tatsächlich das Suchmuster aber nicht enthalten, nennt man *false drops*. Diese false drops müssen im Nachhinein herausgefiltert werden.

false drops

Die meisten Dokumente, die das Suchmuster nicht enthalten, werden aber bereits durch die Testbedingung auf den Signaturen herausgefiltert, so dass die wesentlich aufwändigeren Pattern-Matching-Verfahren nur noch auf recht wenige Dokumente angewendet werden müssen.

Pattern-Matching

Architektur und grobe Funktionsweise eines Systems mit Signaturdateien

Abbildung 4.3 stellt grob die Architektur und Funktionsweise eines Information Retrieval Systems dar, das auf der Basis von Signaturdateien arbeitet.

In dem mit *Signaturen* bezeichneten »Topf« werden dabei grob gesprochen Paare aus Signatur und Dokument-ID verwaltet. Für jedes Wort, das in einem Dokument vorkommt, wird hier ein solches Paar verwaltet. Eine Anfrage wird nun zunächst mittels der Hash-Funktion in eine Anfragesignatur überführt. Mittels dieser Anfragesignatur findet ein Abgleich mit den in der Signaturdatei gespeicherten Signaturen statt. Dabei kann die Signaturdatei sequentiell nach passenden Signaturen durchsucht werden. Wir werden aber im Folgenden noch effizientere Verfahren hierzu kennen lernen. Ergebnis dieses Schritts ist eine Vorauswahl an Dokumenten, die eine passende Signatur enthalten. Diese Vorauswahl wird i.d.R. nur noch einen kleinen Teil der Gesamtkollektion umfassen.

Diese Vorauswahl wird nun in einem weiteren Schritt mittels Pattern-Matching daraufhin überprüft, ob die Dokumente auch wirklich das

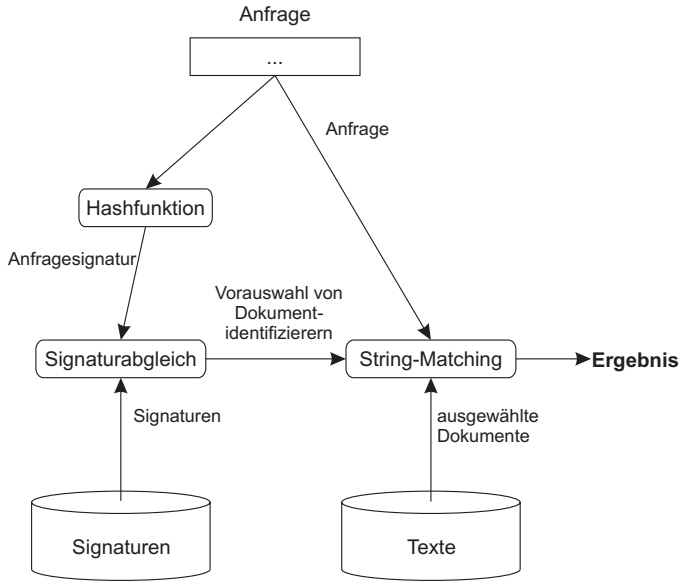


Abbildung 4.3 — Architektur eines Information Retrieval Systems auf Basis von Signaturen

Suchwort in ihrem Text enthalten. Dokumente, die das Suchwort wirklich enthalten, erscheinen im Ergebnis. Dokumente, die die Anfrage nicht enthalten, werden als *false drops* verworfen.

4.2.3 Überlagerungsfähige Signaturen

Bisher haben wir den Fall betrachtet, dass jedes einzelne Wort in einem Text auf eine eigene Signatur abgebildet wird. Dabei werden alle Wörter aus den betrachteten Dokumenten auf alle 2^F möglichen Signaturen abgebildet. Für eine Anfrage vergleicht man nun die aus der Anfrage berechnete Anfragesignatur mit den einzelnen Wortsignaturen. Man bezeichnet diese Signaturen auch als *Binärsignaturen*.

Binärsignaturen

Der Effizienzgewinn entsteht bei diesem Vorgehen dadurch, dass ein Vergleich von zwei Signaturen – Anfragesignatur und Wortsignatur – effizienter erfolgen kann als der Vergleich zweier Zeichenketten.

Innerhalb von Signaturdateien werden nun typischerweise

überlagerungsfähige Signaturen eingesetzt. Dabei wird im Gegensatz zu den Binärsignaturen nicht mehr nur ein Wort auf eine Signatur abgebildet; statt dessen werden mehrere Worte oder ganze Textblöcke in eine Signatur überführt. Diese Überführung von mehreren Wörtern in eine Signatur bezeichnet man auch als *Superimposed Coding*. Dabei werden die Signaturen im Vergleich zu den Binärsignaturen länger.

überlagerungsfähige Signaturen

Ausgangspunkt für überlagerungsfähige Signaturen ist zunächst eine Hash-Funktion, die für ein gegebenes Wort eine Signatur liefert, in der immer genau oder maximal m Bits gesetzt sind. Somit stehen also

$$\binom{F}{m}$$

verschiedene Signaturen zur Verfügung, die Wörter annehmen können. Wobei hier jetzt m als *Signaturgewicht* bezeichnet wird.

Signaturgewicht

Superimposed Coding

Die Eigenschaften des Superimposed Coding lassen sich nun wie folgt zusammenfassen:

Superimposed Coding

- Jedes zu indexierende Wort wird auf einen Bitstring der Länge F abgebildet, in dem genau m Bits gesetzt sind.
- Nehmen wir nun an, ein Textblock b besteht aus D verschiedenen Wörtern. Die Blocksignatur für b wird nun gebildet, indem die D Wortsignaturen mittels ODER verknüpft werden.

Beispiel

Es gilt: $F = 12$, $m = 4$ und $D = 2$

Wenn die verwendete Hash-Funktion für das Wort »free« die Signatur `001 000 110 010` und für das Wort »text« `000 010 111 000` liefert, dann erhalten wir als Blocksignatur `001 010 111 010` (siehe Tabelle 4.2).

Wort	Signatur
free	001 000 110 010
text	000 010 111 000
Blocksignatur	001 010 111 010

Tabelle 4.2 — Beispiel für die Bildung einer Blocksignatur

Hash-Funktionen für Superimposed Coding

Die Anforderung an die beim Superimposed Coding eingesetzte Hash-Funktion immer genau m gesetzte Bits im Ergebnis zu liefern, ist bei gängigen Hash-Funktionen nicht erfüllt. Daher verwendet man zur Bestimmung der Signatur im Superimposed Coding für ein Wort W_j nun m verschiedene Hash-Funktionen $h_1(W_j), \ldots, h_m(W_j)$ (vgl. hierzu auch Knuth [Knu73]).

Dabei bilden alle m Hash-Funktionen h_i aus der Menge der Wörter auf eine Zahl im Bereich 1 bis F ab. Das Bit b_x ist dann in der Signatur für das Wort W_j gleich 1, wenn es ein $i \in 1, \ldots, m$ mit $h_i(W_j) = x$ gibt.

Da mehrere der m Hash-Funktionen auch gleiche Werte liefern können, können in der Signatur auch weniger als m Bits gesetzt sein. Für diesen Fall gibt es folgende Lösungsmöglichkeiten:

- Wenn m deutlich kleiner als F ist, kann dieser Effekt vernachlässigt werden.
- Um jedoch genau m gesetzte Bits zu erreichen verwendet man mehr als m Hash-Funktionen und wendet diese der Reihe nach an, bis die Menge der Ergebniswerte m verschiedene Zahlen enthält. Die Bits mit den entsprechenden Indexwerten werden dann zu 1 gesetzt.

Wie erzeugt man nun m Hash-Funktionen

Zur Erzeugung von m Hash-Funktionen gibt es zahlreiche Möglichkeiten. Eine Möglichkeit ist bspw. in der oben bereits eingeführten Hash-Funktion die als Multiplikator eingesetzte Primzahl durch andere Primzahlen p_i zu ersetzen.

Für ein l-stelliges Wort $w_1, w_2, \ldots, w_l$ ergibt sich die i-te Hash-Funktion dann zu:

```
hash_i
for k := 1 to l do
   hash_i := (hash_i + w_k) * p_i;
end
hash_i := hash_i mod F;
```

Einsatz von Superimposed Coding in Signaturdateien

Zuerst definieren wir die so genannten *Entwurfsparameter* zu Signaturen. Dies sind:

- Die Signaturlänge F,
- das Signaturgewicht m, das die Anzahl der gesetzten Bits pro Wortsignatur festlegt, sowie
- der Überlagerungsfaktor D, der angibt wie viele Wortsignaturen zu einer Blocksignatur zusammengefasst werden.

In einem ersten Schritt werden die zu indexierenden Dokumente in logische Blöcke unterteilt. Ein logischer Block ist ein Teil eines Dokumentes, der genau D verschiedene zu indexierende Wörter – nach einer Stoppworteliminierung – enthält. Durch Anwendung des Superimposed Coding auf diesen logischen Block von Wörtern erhalten wir nun eine Blocksignatur.

Die Suche nach allen Dokumenten, die ein gesuchtes Wort w enthalten, funktioniert nun wie folgt:

- Zuerst wird für das gesuchte Wort die Signatur S_w berechnet.
- Daraufhin wird die Signaturdatei durchsucht, die aus den Blocksignaturen S_{bi} und zugeordneten Dokumenten-IDs besteht. Dabei wird für jede Blocksignatur eine bitweise UND-Verknüpfung ($\wedge$) mit der Signatur des Suchwortes S_w durchgeführt.
- Wenn nun $S_w = S_w \wedge S_{bi}$ gilt, dann sind alle in S_w gesetzten Bits auch in S_{bi} gesetzt, und b_i wird in die Kandidatenmenge mit aufgenommen. Diese Kandidatenmenge kann allerdings durch den Einsatz der Hash-Funktion und die Bildung der Blocksignatur noch Dokumente enthalten, die das Suchwort nicht enthalten.

- Daher wird, um diese *false drops* zu eliminieren, in den Dokumenten der Kandidatenmenge nach dem Vorkommen des Suchwortes w mittels Pattern-Matching gesucht. Die Dokumente, in denen das Suchwort w nicht gefunden wurde, werden als *false drops* verworfen.

konjunktive Anfrage

Bei einer *konjunktiven Anfrage*, das heißt, wenn nach mehreren mit UND verknüpften Suchwörtern gesucht wird, kann man ähnlich vorgehen. Hierzu bildet man durch Überlagerung (ODER-Verknüpfung) der Signaturen der einzelnen Anfragewörter eine Anfragesignatur für die vollständige Anfrage. Verwendet man diese überlagerte Anfragesignatur nun in der oben beschriebenen Weise, so wird man allerdings nur die Dokumente finden, bei denen alle Anfragewörter innerhalb eines Blocks liegen!

Eine Alternative, die mehr Aufwand verursacht, aber diese Schwäche nicht hat ist nun, für jedes Anfragewort eine eigene Kandidatenmenge zu bestimmen. Über diese Kandidatenmengen für die einzelnen Anfragewörter kann dann vor dem Pattern-Matching die Schnittmenge gebildet werden, da nur die Dokumente alle Suchwörter enthalten können, die in allen Kandidatenmengen enthalten sind. Dieser Ansatz kann analog auch für *disjunktive Anfragen* (ODER-Verknüpfung der Suchwörter) angewendet werden. Hier müssen die Kandidatenmengen natürlich vereinigt werden.

disjunktive Anfrage

False Drops

false drops

Wie bereits aufgeführt können bei Signaturdateien sogenannte *false drops* auftreten. Dies bedeutet, dass ein Block bzw. dessen Blocksignatur die notwendige Bedingung erfüllt, das zu suchende Muster aber nicht in dem Block bzw. dem dadurch repräsentierten Dokument enthalten ist.

False drops können sowohl durch den Einsatz der Hash-Funktion, wenn diese verschiedene Wörter auf identische Signaturen abbildet und damit künstliche Homonyme entstehen, als auch durch die Überlagerung der Signaturen auftreten.

Beispiel

Es gelte: $F = 12$, $m = 3$ und $D = 3$.

Betrachten wir zunächst folgende Wortsignaturen:

Wort	Signatur
text	010 010 001 000
search	010 000 100 100
method	100 100 000 001
full	010 110 000 000
knowledge	000 111 000 000
retrieval	000 000 111 000
lexicon	100 010 010 000

Eine konjunktive Suchanfrage bestehe nun aus den UND-verknüpften Suchwörtern »text« und »search«. Die Anfragesignatur ergibt sich somit durch die ODER-Verknüpfung der Signaturen dieser beiden Wörter:

Anfrage	Anfragesignatur
text search	010 010 101 100

Aus einzelnen Textblöcken, bei denen es sich z.B. um Absätze eines Dokumentes oder immer drei aufeinanderfolgende Wörter eines Textes handeln kann, werden nun zu den oben aufgeführten Wortsignaturen die folgenden Blocksignaturen durch Überlagern gebildet:

Textblöcke	Blocksignaturen
text search method	110 110 101 101
search knowledge retrieval	010 111 111 100
full text search	010 110 101 100
lexicon knowledge retrieval	100 111 111 000

Als Antwort auf die obige Suchanfrage erhalten wir nun die ersten drei Blöcke. Dabei stellt sich der zweite Block (»search knowledge retrieval«) als false drop heraus.

Minimierung von False Drops

Eine wichtige Aufgabe bei der Konstruktion von Signaturdateien besteht nun darin, die Entwurfsparameter F, m und D (vgl. Abschnitt 4.2.3) so zu wählen, dass das Verhältnis zwischen dem Speicherplatzbedarf für die Signaturen und der Anzahl der auftretenden *false drops* möglichst günstig ist.

Definition Fehlerwahrscheinlichkeit (false drop probability)

Fehlerwahrscheinlichkeit

Die *Fehlerwahrscheinlichkeit* (*false drop probability*) F_d ist die Wahrscheinlichkeit dafür, dass eine Signatur oder Blocksignatur die notwendige Bedingung erfüllt, aber das durch diesen Block repräsentierte Dokument das zu suchende Muster nicht enthält.

Formal können wir dies durch die bedingte Wahrscheinlichkeit, dass die Signatur die Bedingung erfüllt, wenn das Suchmuster nicht im Block enthalten ist, definieren:

$F_d = P(\text{Signatur erfüllt Bedingung}|\text{Suchmuster nicht im Block enthalten})$

Als Voraussetzungen für unsere weiteren Überlegungen legen wir fest, dass

- zunächst nur einfache Anfragen, die aus einem Wort bestehen, betrachtet werden und
- eine Gleichverteilung der Bits auf die Signaturen besteht.

Berechnung der Fehlerwahrscheinlichkeit

Zunächst berechnen wir die Wahrscheinlichkeit mit der ein Bit in einer **Block**signatur gesetzt ist. Dazu beginnen wir unsere Überlegungen bei den **Wort**signaturen. Da $\frac{m}{F}$ die Wahrscheinlichkeit dafür ist, dass ein Bit in einer Wortsignatur gesetzt ist, ist $1 - \frac{m}{F}$ die Wahrscheinlichkeit dafür, dass ein Bit in einer Wortsignatur nicht gesetzt ist.

Für eine Blocksignatur werden nun D Wortsignaturen überlagert. Damit beträgt die Wahrscheinlichkeit, dass ein Bit in einer Blocksignatur nicht gesetzt ist:

$$\left(1 - \frac{m}{F}\right)^D$$

Die Wahrscheinlichkeit, dass ein Bit in einer Blocksignatur gesetzt ist, beträgt somit:

$$1 - \left(1 - \frac{m}{F}\right)^D$$

Nun berechnen wir die Wahrscheinlichkeit mit der sich eine Blocksignatur für eine einfache Anfrage qualifiziert. Bei einer einfachen Anfrage sind in der Anfragesignatur m Bits gesetzt. Eine Blocksignatur qualifiziert sich für eine Anfrage genau dann, wenn alle m Bits, die in der Anfragesignatur gesetzt sind, auch in der Blocksignatur gesetzt sind. Die Wahrscheinlichkeit beträgt somit:

$$\left(1 - \left(1 - \frac{m}{F}\right)^D\right)^m$$

Geht man nun sehr pessimistisch vor und nimmt an, dass alle Blocksignaturen, die sich qualifizieren, false drops sind, dann beträgt die Fehlerwahrscheinlichkeit:

$$F_d = \left(1 - \left(1 - \frac{m}{F}\right)^D\right)^m$$

Anmerkungen

Bei steigendem F – also bei zunehmender Signaturlänge – fällt F_d monoton. Bei steigendem D – also bei zunehmendem Überlagerungsfaktor – steigt F_d monoton. Für ein steigendes m – also eine größere Anzahl an gesetzten Bits in den Wortsignaturen – kann F_d fallen oder steigen.

Aus diesen Überlegungen ergibt sich nun ein **Optimierungsproblem** für die Bestimmung der optimalen Entwurfsparameter F, m und D: Gegeben sind F und D; man bestimme m so, dass die Fehlerwahrscheinlichkeit minimiert wird.

Für große D können wir dazu die obige Formel für F_d, die auf einer Binomialverteilung basiert, durch die folgende Formel, die diese Binomialverteilung durch eine Exponentialverteilung annähert, approximieren:

$$F_d = \left(1 - e^{-\frac{m\,D}{F}}\right)^m$$

Hieraus ergibt sich durch Ableitung und Umformung der optimale Wert für m als:

$$m_{opt} = \frac{F \cdot \ln 2}{D}$$

Setzen wir dieses Ergebnis in die Formel zur Berechnung der Fehlerwahrscheinlichkeit ein, so erhalten wir:

$$F_d = \left(\frac{1}{2}\right)^{\frac{F \cdot \ln 2}{D}} = \frac{1}{2^{\frac{F \cdot \ln 2}{D}}}$$

Eine weitere Konsequenz ergibt sich durch Einsetzen von $m_{opt} = \frac{F \cdot \ln 2}{D}$ in die oben bestimmte Formel $1 - \left(1 - \frac{m}{F}\right)^D$ für die Wahrscheinlichkeit, dass ein Bit in einer Blocksignatur gesetzt ist. Die Anzahl der Einsen in einer Blocksignatur liegt bei Verwendung von m_{opt} nämlich bei ca. 50%.

Praktische Anwendung der Fehlerwahrscheinlichkeit

Mit der Bestimmung der optimalen Entwurfsparameter können wir nun aus dem Speicherplatzbedarf, den wir für die Signaturen vorsehen, die Fehlerwahrscheinlichkeit berechnen.

Dabei gehen wir von der Annahme aus, dass ein Wort im Durchschnitt aus c_w Zeichen besteht und ein Zeichen jeweils 8 Bit benötigt (wie bspw. bei ASCII). Ferner sei Y die Gesamtzahl der Wörter in den Dokumenten.

Daraus lassen sich nun folgende Werte ermitteln:

- Der Speicherplatzbedarf für die Dokumente beträgt nun $S_D = Y \cdot c_w \cdot 8$ Bit
- Der Speicherplatzbedarf für die Signaturen beträgt $S_S = Y \cdot \left(\frac{F}{D}\right)$ Bit

Wenn wir den Zusatzspeicher, den wir für die Signaturen verwenden wollen, in Relation zu S_D definieren, dann setzen wir $S_S = \alpha \cdot S_D$. Dabei können wir α bspw. auf 10% oder 20% festlegen. Aus α können wir dann die Werte für die anderen Parameter ableiten:

$$\begin{aligned} S_S &= \alpha \cdot S_D \\ Y \cdot \frac{F}{D} &= \alpha \cdot Y \cdot c_w \cdot 8 \\ \frac{F}{D} &= \alpha \cdot c_w \cdot 8 \\ \frac{F}{D \cdot c_w \cdot 8} &= \alpha \end{aligned}$$

Geben wir α vor, so erhalten wir für $\frac{F}{D}$ nun: $\frac{F}{D} = \alpha \cdot c_w \cdot 8$.

Sind wir nun bei einem Wert von $c_w = 10$ bereit, 10% zusätzlichen Speicher für die Signaturen zu verwenden, so ergibt sich $\frac{F}{D} = \alpha \cdot c_w \cdot 8 = 0,1 \cdot 10 \cdot 8 = 8$. Wir könnten also z.B. die Signaturlänge F auf 80 und den Überlagerungsfaktor D auf 10 setzen. Alternative wäre z.B. auch eine Signaturlänge F von 40 und ein Überlagerungsfaktor D von 5 denkbar; was offensichtlich zum gleichen Speicherplatzbedarf führt.

Setzen wir dies in die Formel für die Fehlerwahrscheinlichkeit ein, ergibt sich eine Fehlerwahrscheinlichkeit von:

$$F_d = \frac{1}{2^{\frac{F \cdot \ln 2}{D}}} = \frac{1}{2^{8 \cdot \ln 2}} = 2,14\%$$

Daraus folgt, dass 2,14% der Textdokumente sequentiell mit Pattern-Matching Techniken durchsucht werden müssen – und wir nehmen hier ja vereinfachend an, dass alle diese Dokumente false drops sind.

Würden wir dagegen 20% zusätzlichen Speicher für die Signaturen bereitstellen, so erhalten wir für $\frac{F}{D}$ einen Wert von 16 und somit eine Fehlerwahrscheinlichkeit von:

$$F_d = \frac{1}{2^{\frac{F \cdot \ln 2}{D}}} = \frac{1}{2^{16 \cdot \ln 2}} = 0,0459\%$$

Wir erhalten also bereits mit einem zusätzlichen Speicherplatzbedarf von 20% eine sehr geringe Fehlerwahrscheinlichkeit. Daraus sehen wir, dass wir mit vertretbarem Aufwand und Speicherplatzbedarf die Wahrscheinlichkeit der false drops auf ein akzeptables Niveau reduzieren können.

4.2.4 Speicherungsstrukturen für Signaturdateien

Speicherungsstrukturen für Signaturdateien

Analog zu den invertierten Listen muss nun auch bei den Signaturdateien eine Struktur für die physische Ablage gefunden werden. Hierzu werden in der Literatur insbesondere drei Alternativen vorgeschlagen:

- Sequentielle Signaturdateien
- Bitscheibenorganisation
- S-Baum

Sequentielle Signaturdateien

sequentielle Signaturdateien

Im Prinzip stellt eine Signaturdatei für N Blöcke eine Bitmatrix mit F Spalten und N Zeilen dar, also eine $N \times F$-Matrix. Die einfachste Möglichkeit diese Matrix zu speichern, stellt die sequentielle Speicherung der einzelnen Zeilen der Matrix dar. Die Signaturen werden dazu einfach hintereinander in eine Datei geschrieben. Liegt diese Organisation vor, so spricht man von einer sequentiellen Signaturdatei bzw. einem *Sequential Signature File* (SSF).

Neben der eigentlichen Datei mit den Signaturen, wird zudem eine Datei mit den Verweisen auf die Dokumente oder Blöcke benötigt. Diese Verweise können alternativ auch hinter jeder einzelnen Signatur mit in die Signaturdatei aufgenommen werden.

In Abbildung 4.4 ist der Aufbau eines Sequential Signature File (SSF) dargestellt.

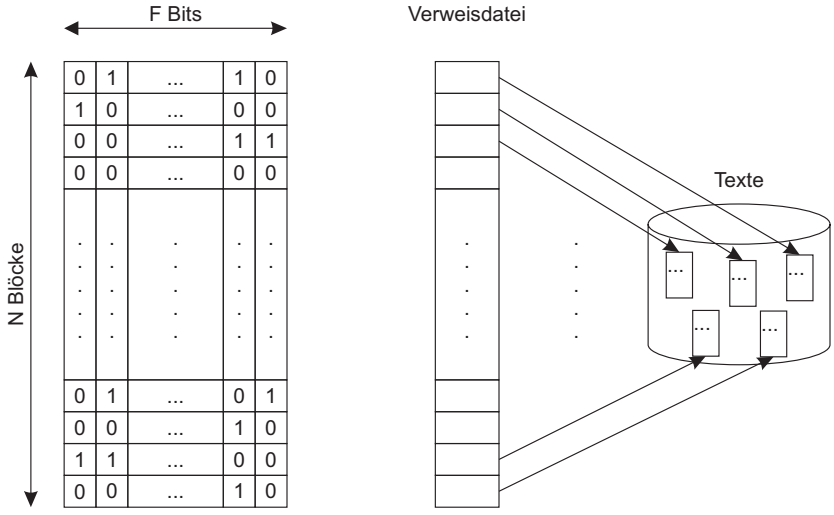


Abbildung 4.4 — Aufbau eines Sequential Signature File (SSF)

Die Auslagerung der Verweise in eine separate Datei hat den Vorteil, dass im Rahmen einer Suche nur die Verweise geladen werden müssen, deren Signaturen als Kandidaten weiter betrachtet werden müssen, während die Signaturdatei ja vollständig gelesen werden muss. Durch dieses vollständige Lesen der Signaturdatei wächst der Aufwand zur Bearbeitung einer Anfrage aber – wie beim Pattern-Matching – linear mit der

Größe der Dokumentenkollektion an. Auch die SSF Organisation gehört damit zur Komplexitätsklasse $O(n)$.

Trotz dieser Problematik kommen Verfahren, die Signaturen einsetzen, derzeit wieder in Mode. Dies liegt unter anderem daran, dass sich mit der heute verfügbaren Rechnerhardware die sequentielle Suche in vertretbarem Aufwand durchführen lässt. Zudem können bei geschickter Wahl von F, das ein Vielfaches der Wortgröße der eingesetzten Rechnerarchitektur sein sollte, sowie großem Hauptspeicher auch sehr große Dokumentenkollektionen sehr effizient durchsucht werden.

Bitscheibenorganisation

Bei sequentiellen Signaturdateien (SSF) werden die Blocksignaturen komplett gelesen. Eine nähere Betrachtung ergibt aber, dass für den Abgleich zwischen der Anfragesignatur und den einzelnen Blocksignaturen nicht alle Bits der Blocksignaturen benötigt werden. Es werden nur die Bits der Positionen in den Blocksignaturen benötigt, die in der Anfragesignatur gesetzt sind. Bei der so genannten *Bitscheibenorganisation* (*Bit-Sliced Signature Files*, BSSF) findet nun eine vertikale Partitionierung des Signaturdateien statt, das heißt die einzelnen Bits einer Signatur werden in F verschiedenen Dateien abgelegt. In Abbildung 4.5 ist der Aufbau einer Bitscheibenorganisation dargestellt.

Bitscheibenorganisation

In jeder der F Dateien werden die Bits einer bestimmten Position über alle Blocksignaturen hinweg hintereinander abgelegt. Man spricht dabei auch von *Bitscheiben*, die in den einzelnen Dateien abgelegt werden. Sie entsprechen einer Spalte in der $N \times F$-Matrix.

Bitscheiben

Bei einer Anfrage mit einem Suchwort braucht man nun nur die Einträge der m Bitscheiben zu lesen, deren Bits in der Anfragesignatur gesetzt sind. Durch eine horizontale UND-Verknüpfung der Einträge dieser Bitscheiben erhält man die Kandidatenmenge.

Beispiel eines Retrieval mit Hilfe eines BSSF

Als Ausgangspunkt seien erneut die folgenden Wortsignaturen gegeben:

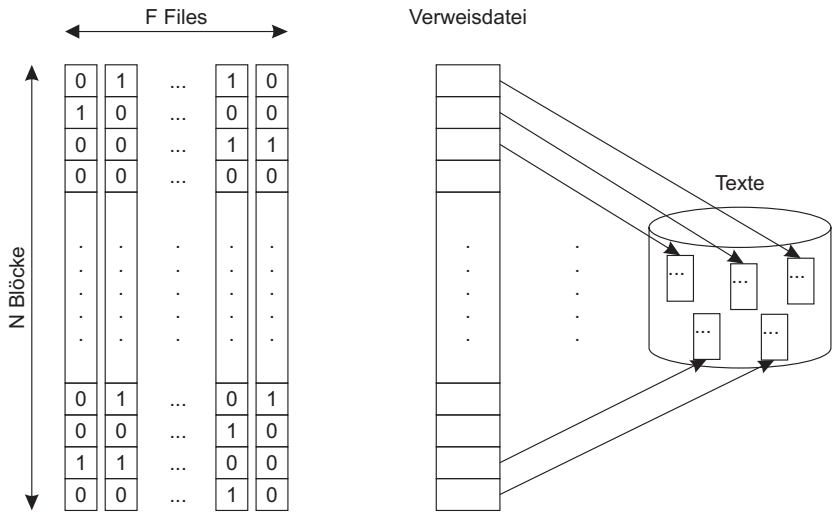


Abbildung 4.5 — Aufbau einer Bitscheibenorganisation (BSSF)

Wort	Signatur
text	010 010 001 000
search	010 000 100 100
method	100 100 000 001
full	010 110 000 000
knowledge	000 111 000 000
retrieval	000 000 111 000
lexicon	100 010 010 000

Für eine Suchanfrage, bei der alle Dokumente gesucht werden, in denen die Worte »text« und »search« innerhalb eines Blocks vorkommen, erhalten wir folgende Anfragesignatur:

Anfrage	Anfragesignatur
text search	010 010 101 100

Beim Einsatz eines BSSF müssen nun zur Bearbeitung dieser Anfrage nur die Bitscheiben 2, 5, 7, 9 und 10 gelesen werden, da nur diese Bits in der Anfrage gesetzt sind. Es muss also nur auf 5 von 12 Bitscheiben und damit nur auf 41,7% der Signaturinformationen zugegriffen werden.

Damit erhalten wir als Kandidatenmenge die in der folgenden Tabelle kursiv hervorgehobenen Blöcke. Dabei ist der zweite Block (»search knowledge retrieval«) ein false drop, der vierte Block muss nicht untersucht werden.

Bitscheibennr.	**2**	**5**	**7**	**9**	**10**
text search method	1	1	1	1	1
search knowledge retrieval	1	1	1	1	1
full text search	1	1	1	1	1
lexicon knowledge retrieval	0	1	1	1	0

Fazit

Die Bitscheibenorganisation (Bit-Sliced Signature Files, BSSF) hat gegenüber dem SSF den Vorteil, dass bei einer Suchanfrage nicht mehr alle Bits sequentiell geprüft werden müssen, sondern nur die Bits, die in der Suchanfrage gesetzt sind. Bei Signaturlängen von z.B. 128 Bit und nur 5 gesetzten Bits in der Anfrage bedeutet dies eine große Zeitersparnis.

Die Bitscheibenorganisation hat aber den Nachteil, dass bei Einfügungen und Löschungen F Plattenzugriffe notwendig sind, da alle Bitscheibendateien entsprechend geändert werden müssen. Vor allem bei heutigen Anwendungen zur Indexierung im Internet oder Intranet ist das Update sehr wichtig, während man vor einigen Jahren noch von eher statischen Datenbeständen ausging.

S-Baum

Im Gegensatz zur Bitscheibenorganisation, die eine vertikale Partitionierung der $N \times F$-Matrix mit den Signaturen vornimmt, findet beim S-Baum eine horizontale Partitionierung dieser $N \times F$-Matrix statt. *S-Baum*

Der S-Baum kann als naher Verwandter des B-Baums betrachtet werden. Wie der B-Baum soll auch der S-Baum effiziente Such- und Änderungsoperationen erlauben – in diesem Fall auf einer Menge von Signaturen.

Aufbau des S-Baums

Jeder Knoten in einem S-Baum besteht aus einer Menge von maximal k Einträgen. Jeder Eintrag besteht aus einem Tupel der Form (*Signatur*,

Adresse). Für innere Knoten des S-Baums zeigt die *Adresse* auf einen Sohn im S-Baum. Bei den Blättern des S-Baums zeigt die *Adresse* auf ein Dokument oder einen Block.

Überlagerung

Bei den inneren Knoten besteht die Signatur aus der Überlagerung (ODER-Verknüpfung) der Signaturen in dem Knoten, der über die mit der Signatur verbundene *Adresse* erreicht werden kann. Dementsprechend enthält ein Blatt nur noch die Signatur des Dokumentes oder Blocks, auf den die *Adresse* verweist.

Durch diese Überlagerung können allerdings bei ungeschickter Zusammenfassung der Signaturen sehr schnell Signaturen erreicht werden, die sehr viele bzw. nur noch Einsen enthalten. Daher sollten in einem Knoten stets nur solche Signaturen zusammengefasst werden, die bezüglich der gesetzten Einsen sehr ähnlich sind.

In Abbildung 4.6 ist exemplarisch ein S-Baum mit einer Knotenkapazität k von 4 Tupeln der Form (*Signatur*, *Adresse*) abgebildet.

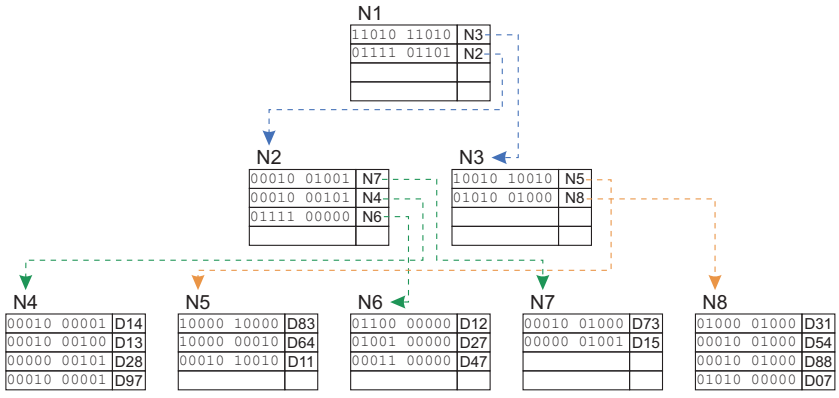


Abbildung 4.6 — Ein exemplarischer S-Baum mit $k = 4$

Es sei zu dieser Abbildung noch angemerkt, dass bspw. im Blatt $N8$ zwei Dokumente mit identischer Signatur enthalten sind, das heißt diese Dokumente können identische Inhalte aufweisen.

Suchen im S-Baum

Suche im S-Baum

Die Suche im S-Baum erfolgt von der Wurzel ausgehend. Hierbei müssen für jeden betrachteten Knoten nur die Söhne weiter untersucht werden,

für die der Signaturabgleich, also die UND-Verknüpfung mit der Anfragesignatur, im Vater erfolgreich war. Dabei kann es durchaus vorkommen, dass mehrere Pfade verfolgt werden müssen.

Würden wir im oben abgebildeten S-Baum nun Dokumente suchen, die zu einer Anfragesignatur `01000 00000` passen, so müssten wir dabei folgendermaßen vorgehen: Im Knoten $N1$ passen die beiden überlagerten Signaturen zu unserer Anfragesignatur, so dass wir die beiden Pfade zu den Knoten $N2$ und $N3$ weiterverfolgen müssen. Innerhalb des Knotens $N2$ passt nur noch die Signatur mit dem Verweis auf $N6$ und im Knoten $N3$ die Signatur mit dem Verweis auf $N8$ zur Anfragesignatur. Aus dem Blattknoten $N6$ passen nun die Signaturen der Dokumente $D12$ und $D27$ und aus dem Blattknoten $N8$ die Signaturen der Dokumente $D31$ und $D07$ zur Anfragesignatur. Somit bilden diese vier Dokumente die Kandidatenmenge zu der gegebenen Anfragesignatur.

Einfügen im S-Baum

Einfügen im S-Baum

Bei Einfügungen wird im S-Baum wie folgt vorgegangen: Von der Wurzel ausgehend wird zunächst das Blatt gesucht, in das der neue Eintrag aus Signatur und Dokument-ID eingefügt werden sollte. Hierzu wird ausgehend von der Wurzel in jedem betrachteten Knoten der Eintrag ausgewählt und weiterverfolgt, bei dem durch das Einfügen der neuen Signatur in dem unterliegenden Teilbaum möglichst wenig zusätzliche Bits in der überlagerten Signatur gesetzt werden müssen.

Tritt dann durch die Einfügung in dem gefundenen Blattknoten ein Überlauf auf – also übersteigt die Anzahl der Einträge k – so werden die Signaturen auf zwei Knoten verteilt. Die Signaturen innerhalb eines Knotens sollten dabei möglichst ähnlich sein.

In Abbildung 4.7 ist das Vorgehen beim Einfügen einer neuen Signatur dargestellt.

In diesem Beispiel wird ein neuer Datensatz mit der Signatur `11000 00000` in den S-Baum eingefügt. Beim Hinzufügen zum Teilbaum mit der Wurzel $N2$ müsste im Knoten $N1$ ein Bit hinzugefügt werden, beim Hinzufügen zum Teilbaum mit der Wurzel $N3$ jedoch kein Bit. Daher wird der Knoten $N3$ weiter betrachtet.

Innerhalb des Knotens $N3$ würde der Bit-Zuwachs in beiden Teilbäumen jeweils 1 Bit betragen. Daher wird in diesem Fall der Teilpfad bzw. das Blatt betrachtet, in dem die Einfügeoperation am leichtesten erfolgen

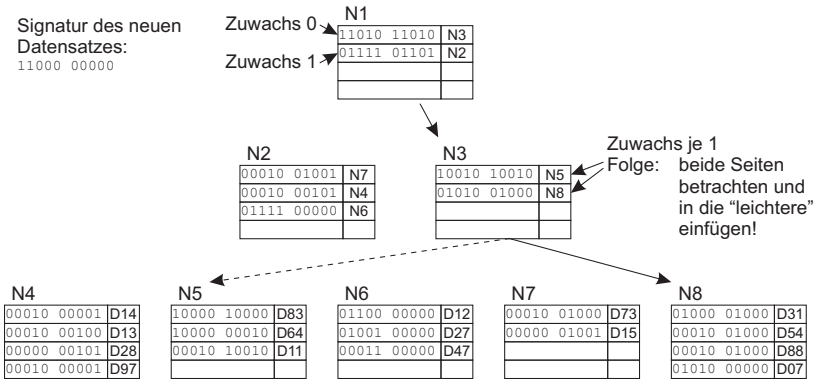


Abbildung 4.7 — Vorgehen beim Einfügen einer neuen Signatur

kann. Dies ist hier Blatt $N5$, weil $N8$ seine Maximalanzahl an Elementen bereits erreicht hat und beim Einfügen auf zwei Blätter aufgeteilt werden müsste.

Vorgehen beim Splitten

Splitten eines Knotens

Wenn ein Blattknoten bzw. eine Seite S_i in die ein neuer Eintrag eingefügt werden soll, bereits die Maximalzahl an Einträgen enthält, so müssen die Einträge dieser Seite auf zwei Seiten S_x und S_y verteilt werden.

Dazu bestimmt man zunächst zwei möglichst verschiedene »Saatsignaturen« und fügt je eine in S_x und S_y ein. Dies kann bspw. dadurch erfolgen, dass man das Paar an Signaturen wählt, das die meisten Unterschiede aufweist. Bei großen Seiten mit b Signaturen pro Seite kann diese Vorgehensweise wegen der erforderlichen b^2 Vergleiche zwischen Signaturen zum Problem werden. Man kann dann unter allen Signaturen die beiden Saatsignaturen z.B. so wählen, dass die Bits der einen Saatsignatur am weitesten links und die Bits der anderen am weitesten rechts liegen.

Anschließend sucht man unter den restlichen Signaturen der Seite S_i abwechselnd nach Signaturen, die am besten zu den bisherigen Signaturen in S_x und S_y passen und überträgt sie dorthin. Würde man dabei nicht abwechselnd vorgehen, würden die bisherigen Einträge auf die neuen Seiten ungleichmäßig verteilt.

Struktur	Profil			
	Suche	Einfügen	Löschen	Speicher
sequentiell	selten	dominant	selten	wichtig
Bitscheiben	dominant	selten	selten	weniger wichtig
S-Baum	(wenig)	viel	viel	weniger wichtig

Tabelle 4.3 — Vergleich der Speicherstrukturen für Signaturen

Der Verweis in der Vaterseite, der bisher auf S_i zeigte, wird nun durch zwei Verweise auf die neuen Seiten S_x und S_y ersetzt. Hierbei kann es zu einem Überlauf in der Vaterseite kommen, der analog durch Splitten der Vaterseite behandelt wird. Gibt es bisher keine Vaterseite, so wird eine neue Wurzelseite mit den zwei neuen Einträgen angelegt.

Fazit

Der große Vorteil des S-Baums liegt darin, dass der Suchaufwand bei einer wachsenden Größe der Dokumentenkollektion »sublinear« anwächst, d.h. die Komplexitätsklasse liegt unter $O(n)$. Allerdings kann man die tatsächliche Komplexitätsklasse des S-Baums nur sehr schwer bestimmen, da die Verteilung der einzelnen Knoten einer Heuristik folgt.

Ein Nachteil des S-Baums ergibt sich allerdings aus der Tatsache, dass die Signaturen in Knoten nahe der Wurzel unter Umständen sehr viele gesetzte Bits aufweisen und damit ihre Unterscheidungskraft für die weitere Suche in den Teilbäumen sehr schwach ist, so dass man zahlreiche Verweise betrachten muss.

Vergleich der Speicherstrukturen für Signaturen

Fuhr [Fuh96] gibt eine grobe Darstellung der Stärken und Schwächen der drei betrachteten Datenstrukturen für Signaturen (vgl. Tabelle 4.3).

Diese Tabelle versucht die sinnvollen Einsatzbedingungen für die verschiedenen Strukturen zu charakterisieren. Dazu wird jeweils im *Profil* angegeben, wie häufig bzw. wichtig die einzelnen Operationen beim Einsatz der verschiedenen Strukturen sein sollten.

So kann die Bitscheibenorganisation sinnvoll eingesetzt werden, wenn die Suche die Hauptaufgabe für die eingesetzte Speicherstruktur ist,

während die sequentiellen Signaturdateien bei zahlreichen Einfügeoperationen Vorteile aufweisen. Der S-Baum wiederum bietet Vorteile beim Löschen von einzelnen Elementen, während die sequentiellen Signaturdateien zum Einsatz kommen sollten, wenn Speicherplatz ein wichtiges Auswahlkriterium ist.

4.3 Boolesches Retrieval

Boolesches Retrieval

Während bei der Wortsuche die Anfrage aus einem Suchwort oder allenfalls einer Menge von konjunktiv verknüpften Suchwörtern bestand, erlaubt das Modell des Booleschen Retrieval beliebige Boolesche Ausdrücke über Suchwörtern als Anfrage zu nutzen. Abbildung 4.8 fasst das bereits in der Einleitung (siehe Abschnitt 1.4.1) kurz skizzierte Modell des Booleschen IR zusammen:

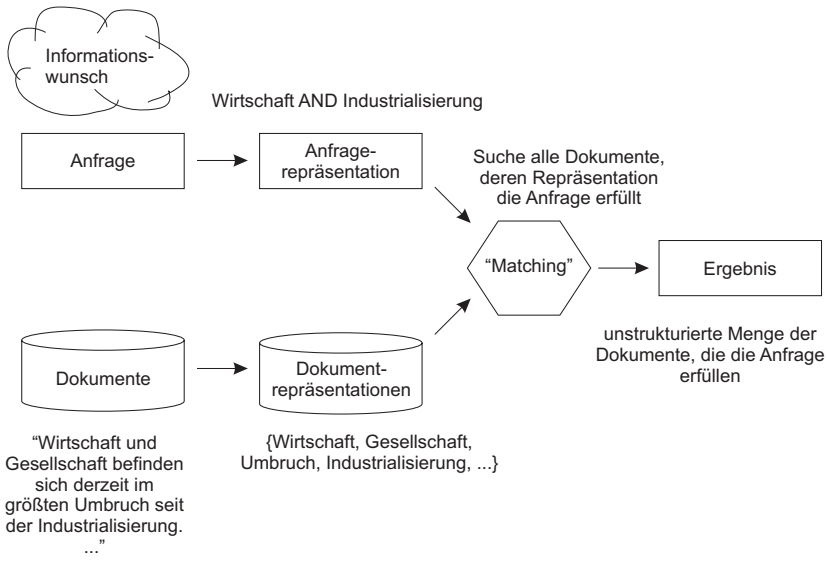


Abbildung 4.8 — Vorgehen beim Booleschen Retrieval

Dokumentenrepräsentation im Booleschen Retrieval

Die einzelnen Dokumente werden beim Booleschen Retrieval durch die Menge der in ihnen auftretenden Wörter repräsentiert. Dabei sind zahlreiche Varianten für diese Repräsentation denkbar:

- Zunächst kann statt einer Menge auch eine *Multimenge* der in einem Dokument vorkommenden Wörter als Repräsentation benutzt werden. Typischerweise wird dabei auf der Implementierungsebene zu jedem Wort seine Vorkommenshäufigkeit im Dokument gespeichert. In diesem Fall kann beim Matching auch die Vorkommenshäufigkeit von einzelnen Wörtern berücksichtigt werden.

  Vorkommenshäufigkeit

- Eine weitere Möglichkeit stellt die Repräsentation durch eine Menge aus Paaren von jeweils einem Wort und dessen *Vorkommensort* dar. Als Vorkommensort kann dabei z.B. vermerkt werden, ob ein Wort im Titel, in einer Überschrift oder »nur« im normalen Text auftritt. Daneben kann aber auch die Wortposition im Text verwaltet werden. Dazu kann sowohl eine Wortnummer, die die Position im Text angibt, als auch eine Satznummer und eine Wortnummer im Satz verwendet werden. Die Speicherung des Vorkommensortes ermöglicht Suchmaschinen bspw. Anfragen zu beantworten, in denen mittels des Schlüsselwortes NEAR definiert ist, dass in den gesuchten Dokumenten zwei Begriffe *nahe beieinander* stehen müssen. *Nahe beieinander* bedeutet dann je nach Suchmaschine entweder im gleichen Satz oder durch eine maximale Anzahl von z.B. 10 anderen Worten getrennt.

  Vorkommensort

Ferner kann man natürlich auch vor der Erstellung der Repräsentation noch eine Stoppworteliminierung und/oder eine Stammformreduktion durchführen.

Anfragerepräsentation im Booleschen Retrieval

Eine aus einem Informationswunsch abgeleitete Anfrage muss beim Booleschen Retrieval in einen Booleschen Ausdruck überführt werden. Wird als Anfrage nur eine Menge an Wörtern angegeben, so werden diese ja nach Information Retrieval System implizit entweder disjunktiv (mit ODER) oder konjunktiv (mit UND) verknüpft.

Beispiele für Anfragen sind:

- **»Adler Bär«**: sucht nach Dokumenten, die bei einer vom Information Retrieval System implizit angenommenen UND-Verknüpfung beide Worte enthalten.
- **»Adler AND (Bär OR Löwe)«**: sucht nach Dokumenten, die das Wort »Adler« und mindestens eines der Worte »Bär« oder »Löwe« enthalten.
- **»Adler AND NOT Löwe«**: sucht nach Dokumenten, die das Wort »Adler« aber nicht das Wort »Löwe« enthalten.

Möglichkeiten der Implementierung

Die Implementierung eines auf dem Booleschen Retrieval basierenden IR-Systems kann auf verschiedenen Methoden aufbauen. Eine Auswahl dieser Methoden sind:

Pattern Matching
- Pattern Matching, das heißt bei einer Suchanfrage wird die gesamte Dokumentenkollektion durchlaufen. Dies ist gerade bei größeren Dokumentenkollektionen eine sehr ineffiziente Methode.

Signatur
- Signaturen, sind insbesondere für die UND-Verknüpfung im Booleschen Retrieval interessant, da Signaturen die UND-Verknüpfung von Suchtermen effizient unterstützen, sofern sie blockweise und nicht dokumentenbezogen interpretiert wird.

invertierte Liste
- Invertierte Listen bilden die dritte und vielseitigste Möglichkeit. Mit ihnen können auch relativ komplexe Boolesche Anfragen, die bei der Bearbeitung mit Signaturen Probleme aufwerfen würden, einfach und effizient bearbeitet werden.

Im Folgenden werden wir nun ausführlicher auf die Implementierung des Booleschen Retrieval mit invertierten Listen eingehen.

Implementierung des Booleschen Retrieval mit invertierten Listen

Die einzelnen Operationen des Booleschen Retrieval lassen sich recht einfach auf invertierte Listen abbilden. Wir werden dazu im Folgenden die verschiedenen Operationen betrachten.

Einfache Anfragen

Unter einer einfachen Anfrage wollen wir hier eine Anfrage verstehen, die – analog zur Wortsuche – nur aus einem einzelnen Wort besteht. Eine solche Anfrage, die wir mit q bezeichnen wollen, lässt sich nun dadurch beantworten, das wir die dem Suchwort zugeordnete Liste als Ergebnis zurückliefern.

Betrachten wir dazu erneut unser Beispiel für eine invertierte Liste:

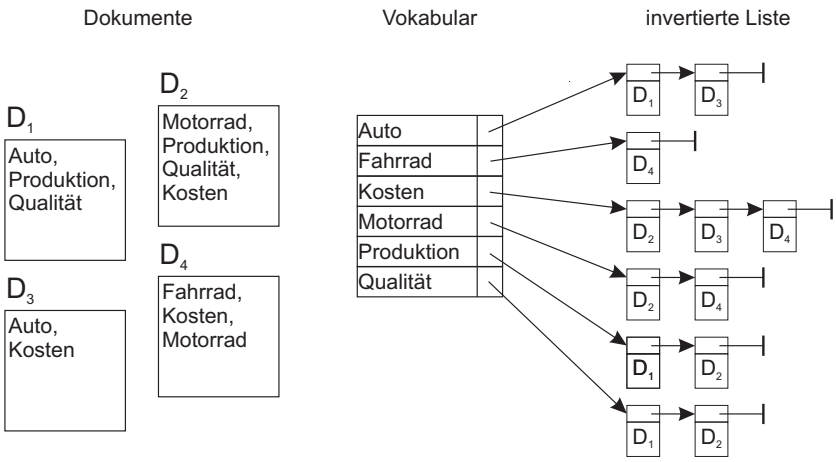


Abbildung 4.9 — Ein Beispiel einer invertierten Liste

In diesem Beispiel könnte eine Anfrage $q =$ »Produktion« mit der Rückgabe der diesem Begriff zugeordneten Liste mit den Dokumenten D_1 und D_2 beantwortet werden.

UND-verknüpfte Anfragen

Im Weiteren werden wir nun ausnutzen, dass einfache Anfragen und auch die UND- und die ODER-Verknüpfung von Teilanfragen immer Listen mit Dokument-IDs als Ergebnis liefern. Eine Anfrage, die aus zwei UND-verknüpften Teilanfragen besteht ($q = q_1$ AND q_2) wird durch die Bildung der Schnittmenge der beiden Dokumentlisten beantwortet. Die Schnittmenge beinhaltet nur die Elemente, die in beiden Dokumentlisten enthalten sind.

Schnittmenge der Listen

Ein sehr effizienter Weg zur Implementierung dieser Schnittoperation ist, die Dokumentlisten jeweils nach den Dokument-IDs sortiert zu verwalten. In diesem Fall kann die Schnittmenge effizient in einem parallelen Durchlauf der beiden Listen für q_1 und q_2 bestimmt werden (vgl. Abschnitt 4.2.1).

Eine Anfrage »Produktion AND Kosten« würde damit im obigen Beispiel als Ergebnismenge nur das Dokument D_2 zurückliefern.

ODER-verknüpfte Anfragen

Vereinigungsmenge der Listen

Eine ODER-verknüpfte Anfrage $q = q_1$ OR q_2 entspricht der Vereinigungsmenge der beiden Dokumentlisten zu q_1 und q_2.

Eine Anfrage »Produktion OR Motorrad« würde als Ergebnismenge die Dokumente D_1, D_2 und D_4 liefern.

Negation von Anfragen

Eine Anfrage in der Form $q =$ NOT q_2 würde bedeuten, dass alle Dokumente gesucht werden, die der Anfrage q_2 nicht entsprechen. Dies würde im Allgemeinen bedeuten, dass eine Liste mit allen Dokument-IDs außer denen in der Ergebnisliste zu q_2 erstellt werden müsste. Dies würde i.d.R. eine sehr lange Liste von Dokumenten ergeben.

In unserem Beispiel würde die Anfrage »NOT Auto« zwar ein durchaus überschaubares Ergebnis mit D_2 und D_4 liefern, bei einer größeren Dokumentenkollektion ist die Bearbeitung solcher Anfragen aber problematisch. Zudem erscheinen derartige Anfragen dann auch wenig sinnvoll.

Dies führt auch dazu, dass eine unäre NOT-Operation in realen IR-Systemen meist nicht zu finden ist. Viele Systeme bieten nur die Operation q_1 AND NOT q_2 an. Anfragen in dieser Form können implementiert werden, indem aus der Liste, die q_1 entspricht, die Elemente, die auch in q_2 enthalten sind, entfernt werden.

Eine Anfrage »Kosten AND NOT Motorrad« wird nun so bearbeitet, dass zuerst die Dokumentliste zum Begriff »Kosten« mit D_2, D_3 und D_4 ermittelt wird und aus dieser Menge dann die Elemente entfernt werden, die in der Dokumentliste zum Begriff *Motorrad* enthalten sind. Da dies D_2 und D_4 sind, erhalten wir als Ergebnisliste schließlich D_3.

Weitere Operationen

NEAR-Operator

Eine weitere Operation ist bspw. t_i NEAR[k] t_j. Diese Operation sucht nach allen Dokumenten, in denen ein Term t_i nicht weiter als k Worte von einem anderen Term t_j entfernt steht. Weitere denkbare Operationen suchen bspw. in bestimmten Teilen oder Abschnitten eines Dokuments. So könnte eine Operation IN_TITLE nur die Dokumente als Ergebnis liefern, die einen gegebenen Begriff im Titel enthalten.

Um solche Operationen mit Hilfe von invertierten Listen bearbeiten zu können, müssen in den einzelnen Listenelementen zusätzliche Informationen verwaltet werden.

Um z.B. eine NEAR-Operation zu implementieren müsste man zusätzlich zu jeder Dokumentenreferenz in den invertierten Listen noch eine Liste der Vorkommenspositionen des Begriffs im Dokument hinterlegen. Um dann eine Anfrage der Form t_i NEAR[k] t_j zu bearbeiten, müssen die Listen zu t_i und t_j parallel durchlaufen werden. Dabei müssen für jedes Dokument, das in beiden Listen vorkommt, die Vorkommenspositionen der Begriffe verglichen werden. Gibt es in den Listen der Vorkommenspositionen ein Paar, dessen Einzelwerte nicht weiter als k voneinander entfernt sind, so wird das Dokument in das Ergebnis übernommen.

Zusammenfassung Boolesches Retrieval

Das Boolesche Retrieval weist in seiner Grundform eine Reihe von Schwachstellen auf:

- In seiner einfachsten Form führt das Boolesche Retrieval keine Rückführung der indexierten Wörter auf ihre Grund- oder Stammform durch. Dies kann aber ergänzt werden.
- Schwerer wiegt das Problem, dass keine Gewichtung der Wörter nach dem Ort des Vorkommens oder der Vorkommenshäufigkeit stattfindet. Auch unterstützt das Boolesche Retrieval in seiner Grundform keine Zerlegung von Mehrwortgruppen.
- Als weitere Schwachstelle kann die relativ aufwändige Formulierung der Anfrage aufgeführt werden.
- Weitere Schwachstellen des Booleschen Retrieval sind die kaum vorhersehbare Größe der Ergebnismenge sowie das fehlende Ranking der Dokumente.

Einige dieser Schwachstellen können abgemildert werden. Bspw. kann durch Erweiterungen die Häufigkeit der Begriffe oder deren Vorkommensort mit ausgewertet werden, so dass sich auch beim Booleschen Retrieval ein Ranking realisieren lässt. Wir wollen hierzu aber im Folgenden zwei andere Ansätze betrachten.

4.4 Coordination-Level-Match

Coordination Level Match

Das Hauptproblem des Booleschen Retrieval besteht in der fehlenden Strukturierung der Ergebnismenge. Diesem Umstand will das *Coordination Level Match* Rechnung tragen.

Beispiel

Gegeben sei folgende Anfrage an ein Information Retrieval System: »Haus AND Garten AND Italien«.

Als Folge dieser Anfrage würden nun bei einem Booleschen Retrieval alle Dokumente, die zwei der gesuchten Begriffe enthalten, ebenso verworfen, wie Dokumente, die keinen der gesuchten Begriffe enthalten.

Finden sich in der indexierten Dokumentenkollektion allerdings keine Dokumente, die alle drei gesuchten Begriffe enthalten, dafür aber Dokumente, die zwei der gesuchten Begriffe beinhalten, so wäre es wünschenswert, diese Dokumente im Ergebnis zu erhalten.

Vorgehen des Coordination Level Match

Beim Coordination Level Match besteht nun eine Anfrage aus:

- Termen, die in den Dokumenten vorkommen sollen und
- Termen, die nicht in den Dokumenten vorkommen sollen.

Die Retrieval-Funktion berechnet nun zu jedem Dokument eine Kennzahl:

- Begriffe, die in der Anfrage gewünscht werden und auch im Dokument vorkommen, werden dabei mit einem positiven Wert, also bspw. 1, berücksichtigt.

- Für Begriffe, die laut der Anfrage nicht in den Dokumenten vorkommen sollen, wird ein negativer Wert, also bspw. -1, berücksichtigt, wenn sie doch in einem Dokument vorkommen.

Diese Werte werden nun für jedes Dokument aufaddiert.

Betrachten wir z.B. die Anfrage »Haus, Garten, NOT Frankreich«, so würde sich für das Dokument »Garten in Frankreich« ein Wert von 0 ergeben, weil »Garten« mit +1 bewertet wird und »Frankreich« mit -1.

Implementierung des Coordination Level Match

Die Implementierung des Coordination Level Match lässt sich recht einfach über invertierte Listen realisieren. Vorteilhaft ist dabei wiederum, wenn die einzelnen Listen nach den Dokumenten-IDs sortiert sind. Man kann die Listen zu den in der Anfrage vorkommenden Termen parallel durchlaufen und zu den Dokumenten nacheinander die Kennzahlen berechnen.

Beispiel

Gegeben sei eine Anfrage: »Kosten, Motorrad, NOT Produktion«

Wir durchlaufen nun die einzelnen Listen parallel und addieren die einzelnen Werte für jedes Dokument auf. Dies ist in Abbildung 4.10 visualisiert.

Dokumente, die in keiner der Listen vorkommen, werden nun ebenso wie Dokumente, die einen Wert kleiner oder gleich 0 erzielen, nicht in die Ergebnismenge mit aufgenommen.

Die Dokumente in der Ergebnismenge werden absteigend nach ihrem Ergebniswert sortiert. Somit erhalten wir als Ergebnis die Dokumente: D_4 im ersten Rang und D_3 sowie D_2 im zweiten Rang.

Der Aufwand für das Durchlaufen der Listen ist recht günstig. Wir erhalten eine Komplexität von ca. $O(n \cdot m)$. Wobei n die Anzahl der Dokumente und m die Anzahl der parallel zu durchlaufenden Listen darstellt. In der Praxis wird n allerdings deutlich kleiner als die Menge aller indexierten Dokumente sein, da nicht alle Terme in allen Dokumenten enthalten sind.

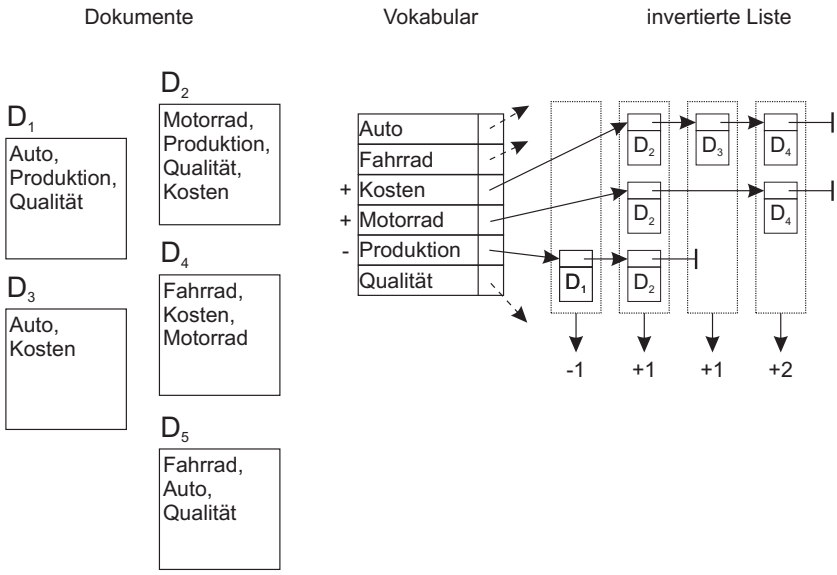


Abbildung 4.10 — Coordination Level Match mit Hilfe invertierter Listen

Des Weiteren ist festzustellen, dass sich die Anfrageformulierung beim Coordination Level Match im Vergleich zum Booleschen Retrieval einfacher gestaltet.

4.5 Fuzzy-Set Modell

Fuzzy-Set Modell

Das Fuzzy-Set Modell ist der Versuch Fuzzy-Sets mit Information Retrieval zu kombinieren.

Insbesondere beim Booleschen Retrieval wird davon ausgegangen, dass die Zugehörigkeit eines Elements zu einer Menge immer eindeutig entscheidbar ist. So wird einem Term t_i letztlich eine Menge von Dokumenten zugeordnet, die er repräsentiert. Diese Menge entspricht den Dokumenten, die in der invertierten Liste zu t_i enthalten sind, und damit der Menge der Dokumente, in denen t_i vorkommt.

Dies trägt aber in keiner Weise der Unschärfe in der Begriffsverwendung Rechnung. So kann ein Begriff z.B. auch dann ein guter Repräsentant für ein Dokument sein, wenn er nicht in diesem Dokument vorkommt.

Dokumente, die z.B. den Begriff »SQL« enthalten, beziehen sich durchaus auch auf den Einsatz von Datenbanken, selbst wenn sie das Wort »Datenbank« nicht enthalten.

Letztlich erscheint es sinnvoller, jedem Dokument einen Grad der Zugehörigkeit zu der Menge T_i der Dokumente, die durch t_i repräsentiert werden, zuzuordnen, der zwischen 0 und 1 liegen kann. Das heißt, wir geben für jedes Dokument D_j und jeden Term t_i aus dem Vokabular einen Wert $\mu_{i,j} \in [0;1]$ an, der angibt, wie repräsentativ t_i für D_j ist. Damit betrachten wir die Menge der Dokumente, die durch einen Term t_i repräsentiert werden, als Fuzzy-Set.

Definition: Fuzzy-Set

Fuzzy-Set

Ein Fuzzy-Set S über einer Grundgesamtheit G wird durch eine Funktion $\mu_S : G \rightarrow [0;1]$ beschrieben, die für jedes Element aus G den Grad der Zugehörigkeit zu S angibt.

Basierend auf dieser Definition werden die Negation, die ODER-Verknüpfung und die UND-Verknüpfung auf Fuzzy-Sets dann üblicherweise wie folgt definiert:

- Negation: $\mu_{\neg S}(u) = 1 - \mu_S(u)$
- ODER-Verknüpfung: $\mu_{A \cup B}(u) = max(\mu_A(u), \mu_B(u))$
- UND-Verknüpfung: $\mu_{A \cap B}(u) = min(\mu_A(u), \mu_B(u))$

Bei der ODER-Verknüpfung verwenden wir also das Maximum der Zugehörigkeitswerte und bei der UND-Verknüpfung das Minimum.

Es soll an dieser Stelle nicht verschwiegen werden, dass es in der Literatur zahlreiche weitere Ansätze für die logische Verknüpfung einzelner Fuzzy-Sets gibt, die teilweise andere Berechnungen zugrunde legen. Die oben angeführte Interpretation ist aber sehr häufig anzutreffen.

Repräsentation der Anfragen und Dokumente

Anfragen können damit weiter im Stile des Booleschen Retrieval gestellt werden. Ein Dokument D_j wird nun aber durch die Werte $\mu_{i,j}$ für alle Terme des Vokabulars repräsentiert, auf deren Bestimmung wir später eingehen werden.

Beispiel

Das Vokabular besteht aus vier Termen und wir betrachten fünf Dokumente.

In der folgenden Tabelle sind die $\mu_{i,j}$ für diese Konstellation angegeben. In diesem Beispiel sind die Werte »vom Himmel gefallen«. Im Weiteren werden wir aber noch auf ihre Berechnung eingehen.

Die Anfrage lautet nun:

Haus AND (Italien OR Frankreich) AND NOT Garten

Wenn wir nun Haus als t_1, Garten als t_2, Italien als t_3 und Frankreich als t_4 betrachten, dann erhalten wir für diese Anfrage die Retrievalfunktion:

$$min(min(\mu_{1,j}, max(\mu_{3,j}, \mu_{4,j})), 1 - \mu_{2,j})$$

Aus den einzelnen Zugehörigkeitswerten $\mu_{i,j}$ zwischen den Termen und den Dokumenten lassen sich nun die Zugehörigkeitswerte für die Anfrage berechnen:

	Dokumente D_j				
Term	D_1	D_2	D_3	D_4	D_5
Haus	0,5	0,0	0,7	0,3	0,9
Garten	0,9	1,0	0,1	0,0	0,0
Italien	1,0	0,0	0,0	1,0	0,0
Frankreich	1,0	1,0	0,4	1,0	0,5
Anfrage	0,1	0,0	0,4	0,3	0,5

Bestimmung der Zugehörigkeitswerte

Term-Term-Matrix

Die Bestimmung der einzelnen Zugehörigkeitswerte $\mu_{i,j}$ kann nach [OMK91] über eine so genannte Term-Term-Matrix erfolgen. In einem ersten Schritt bauen wir hierzu die Term-Term-Matrix auf. In dieser Matrix tragen wir die Korrelationen zwischen den einzelnen Termen ein.

Sei n_i die Anzahl der Dokumente, die Term t_i enthalten und n_l die Anzahl der Dokumente, die Term t_l enthalten. Ferner sei $n_{i,l}$ die Anzahl der Dokumente, die sowohl Term t_i als auch Term t_l enthalten.

Dann berechnen wir die Elemente $c_{i,l}$ der Term-Term-Matrix – also die Korrelationen – durch:

$$c_{i,l} = \frac{n_{i,l}}{n_i + n_l - n_{i,l}}$$

Man beachte, dass $c_{i,l} = 1$ für $i = l$ gilt.

Beispiel

Gegeben sind zwei Terme »Haus« und »Dach«. Der Term »Haus« kommt in 7 und »Dach« in 6 und beide zusammen kommen in 5 Dokumenten vor.

Damit erhalten wir: $n_i = 7$, $n_l = 6$ und $n_{i,l} = 5$. Als Ergebnis erhalten wir damit $c_{i,l} = \frac{5}{8}$.

Die höchste Korrelation mit Wert 1 zwischen zwei Begriffen erhalten wir, wenn diese beiden Begriffe in den selben Dokumenten enthalten sind. Die niedrigste Korrelation zweier Begriffe mit Wert 0 erhalten wir, wenn diese nicht in den selben Dokumenten enthalten sind.

Wir können nun den Grad der Zugehörigkeit eines Dokumentes D_j zur Menge der Terme T_i berechnen als:

$$\mu_{i,j} = 1 - \prod_{t_l \in D_j} (1 - c_{i,l})$$

Man kann dies wie folgt interpretieren:

- $c_{i,l}$ ist die Wahrscheinlichkeit der Zugehörigkeit zu T_i wenn t_l im Dokument vorkommt.
- $1 - c_{i,l}$ ist die Wahrscheinlichkeit der Nicht-Zugehörigkeit zu T_i wenn t_l im Dokument vorkommt.
- $\prod_{t_l \in D_j}(1 - c_{i,l})$ ist die Wahrscheinlichkeit der Nicht-Zugehörigkeit zu T_i für D_j ermittelt über alle in D_j vorkommenden Wörter.
- $1 - \prod_{t_l \in D_j}(1 - c_{i,l})$ ist schließlich die Wahrscheinlichkeit der Zugehörigkeit zu T_i für D_j.

Implementierung

Für eine Implementierung des Fuzzy-Set Modells können nun wieder invertierte Listen zum Einsatz kommen. Dabei muss die Liste zu einem Term t_i alle Dokumente enthalten, deren $\mu_{i,j} > 0$ ist.

Probleme bereitet bei der Implementierung allerdings die Negation.

Kritisch ist auch, dass insbesondere bei längeren Dokumenten die Listen sehr lang werden, weil es sehr viele $\mu_{i,j} > 0$ gibt. In diesem Fall kann man einen sinnvollen Grenzwert definieren, bis zu dem die $c_{i,l}$ berücksichtigt werden sollen.

Einordnung des Fuzzy-Set Modells

Das Fuzzy-Set Modell weist an einigen Stellen in die richtige Richtung. So ergibt sich als Ergebnis in diesem Modell eine Ordnung der Dokumente – ein Ranking. Ferner werden bei dem beschriebenen Modell die Korrelationen zwischen den Termen berücksichtigt. Dennoch weist auch das Fuzzy-Set Modell eine Reihe von Nachteilen auf:

- Durch die Maximum- und Minimumbildung wird eine extreme Position vertreten. Ein Beispiel mag dies verdeutlichen, wir betrachten dazu die Anfrage »Haus AND Italien«:

Term	D_1	D_2
Haus	0,3	0,9
Italien	0,3	0,2
Anfrage	0,3	0,2

  Das System hält also D_1 für relevanter als D_2! Der Grund ist, dass durch die Verwendung des Minimums der wesentlich bessere Wert für »Haus« bei D_2 nicht berücksichtigt wird.

- Das Problem der relativ komplexen Booleschen Anfrageformulierung bleibt weiter erhalten. Es muss weiterhin ein Boolescher Anfrageterm durch den Benutzer formuliert werden.

- Bei der Implementierung ergeben sich gegenüber dem Booleschen Retrieval Performancenachteile, da die einzelnen Listen durch die Berücksichtigung der Korrelationen deutlich länger werden. Dabei kann man von einem Faktor zwischen 5 und 10 ausgehen.

- Dynamische Datenbestände bereiten Probleme, weil die Term-Term-Matrix eigentlich jedes Mal geändert werden müsste wenn ein neues Dokument eingefügt oder ein altes gelöscht wird. Damit müssten aber auch alle Einträge in den invertierten Listen aktualisiert werden. Daher sollte man gegebenenfalls die Matrix nur für einen repräsentativen Datenbestand berechnen.

Zusammenfassend kann man feststellen, dass das Fuzzy-Set Modell der Unschärfe der natürlichen Sprache im Information Retrieval Rechnung trägt, indem es über Term-Term-Matrizen die Korrelationen zwischen den verwendeten Begriffen berücksichtigt. Die auf einer Maximums- und Minimumsbetrachtung basierende Retrieval-Funktion erscheint aber problematisch.

Die Ermittlung der Zugehörigkeitswerte zu den Mengen der von den einzelnen Termen repräsentierten Dokumente muss dabei nicht zwangsweise über eine Term-Term-Matrix erfolgen. Es ist z.B. auch möglich, die Korrelation der einzelnen Terme untereinander aus einem Thesaurus zu entnehmen.

Andererseits kann man die aus einer Term-Term-Matrix ermittelten Korrelationen zwischen den Begriffen auch zur automatischen Erstellung eines Synonymwörterbuches benutzen. Dabei handelt es sich aber nicht um Synonyme in einem semantischen Sinn, sondern lediglich um Wörter, die häufig in den gleichen Dokumenten vorkommen.

4.6 Zusammenfassung

In diesem Kapitel haben wir vier relativ einfache Modelle des Information Retrieval betrachtet:

- die Wortsuche,
- das Boolesche Retrieval,
- das Coordination Level Match und
- das Fuzzy Set Modell.

Zur Implementierung dieser Modelle wurden

- invertierte Listen und
- Signaturen

vorgestellt.

Sowohl die Modelle als auch insbesondere die Implementierungstechniken sind dabei in der Praxis durchaus weit verbreitet. Zudem stellen Sie eine wichtige Ausgangsbasis für weiterführende Modelle dar. Als weiterführende Lektüre zu den Implementierungstechniken sei dabei dem interessierten Kursteilnehmer das Buch von Frakes und Baeza-Yates [FBY92] empfohlen.

5 Das Vektorraummodell

Vektorraummodell

Eines der am häufigsten eingesetzten Modelle im Information Retrieval ist das Ende der 1960er Jahre vorgeschlagene Vektorraummodell (vgl. [SL68], [Sal71] oder [SWY75]).

Das Vektorraummodell repräsentiert sowohl die Dokumente als auch die Anfragen durch t-dimensionale Vektoren, wobei t die Anzahl der Terme im benutzten Indexierungsvokabular ist. Die Komponenten dieser t-dimensionalen Vektoren stellen die Gewichtung der entsprechenden Terme im Hinblick auf das jeweilige Dokument oder die Anfrage dar. Ein hohes Gewicht bedeutet, dass der jeweilige Term das Dokument bzw. die Anfrage gut charakterisiert.

Boolesches Retrieval

Die so genannten Termgewichte sind dabei im Gegensatz zum Booleschen Retrieval nicht binär – also 0 oder 1 – sondern zunächst beliebige nichtnegative Zahlen. Dies gilt im Gegensatz zum Fuzzy-Retrieval auch für die Anfragetermgewichte, so dass in der Anfrage eine Gewichtung der Terme vorgenommen werden kann.

5.1 Die Grundidee des Vektorraummodells

Definitionen

Zur genaueren Betrachtung des Vektorraummodells benötigen wir einige Definitionen:

- N sei die Anzahl der Dokumente in unserer Dokumentenkollektion.
- n_k sei die Anzahl der Dokumente in unserer Dokumentenkollektion, die den Begriff bzw. Term k enthalten (offensichtlich gilt $0 \leq n_k \leq N$ für $k \in 1, \ldots, t$).
- tf_{dk} sei die Vorkommenshäufigkeit von Begriff k in Dokument D ($k \in 1, \ldots, t$).

Repräsentation mittels Vektoren

Ein Dokument D wird nun durch einen Vektor $\mathcal{D} = (w_{d1}; w_{d2}; \ldots; w_{dt})$ und eine Anfrage Q – analog zu den Dokumenten – durch einen Vektor $\mathcal{Q} = (w_{q1}; w_{q2}; \ldots; w_{qt})$ repräsentiert. Dabei entsprechen die w_{dk} der Relevanz des Dokumentes D für den Begriff k und die w_{qk} entsprechen der Bedeutung des Begriffs k für die Anfrage Q.

Skalarprodukt

Die Ähnlichkeit zwischen einer Anfrage Q und einem Dokument D lässt sich nun bspw. durch das Skalarprodukt der beiden Vektoren $\mathcal{D}$ und $\mathcal{Q}$ ausdrücken:

$$similarity(\mathcal{Q}, \mathcal{D}) = \sum_{k=1}^{t} w_{qk} \cdot w_{dk} \tag{5.1}$$

Man schreibt hierfür auch $\mathcal{D} \bullet \mathcal{Q}$.

Offensichtlich entspricht dieses Ähnlichkeitsmaß der mit den Anfragetermgewichten w_{qk} gewichteten Summe der Termgewichte des betrachteten Dokuments w_{dk}. Die Ähnlichkeit ist umso größer, je höher die Termgewichte des betrachteten Dokuments zu den für die Anfrage relevanten Termen sind.

Ein einführendes Beispiel

Es seien folgende Dokumente D_1 bis D_5 und die Anfrage Q gegeben:

- D_1 = »Häuser in Italien«
- D_2 = »Häuser in Italien und um Italien«
- D_3 = »Gärten und Häuser in Italien«
- D_4 = »Gärten in Italien«
- D_5 = »Gärten und Häuser in Frankreich«
- Q = »Häuser in Italien«

Damit besteht das Indexierungsvokabular nach der Entfernung der Stoppwörter »in«, »um«, sowie »und« aus den Worten: *Häuser*, *Italien*, *Gärten* und *Frankreich*. Das Vokabular hat damit die Dimension $t = 4$. *Indexierungsvokabular*

Für die Beschreibungsvektoren legen wir eine Reihenfolge zu den Termen im Indexierungsvokabular fest: *Häuser* = Komponente 1, *Italien* = Komponente 2, *Gärten* = Komponente 3 und *Frankreich* = Komponente 4.

Wenn wir nun in einem ersten sehr groben Ansatz die tf_{dk} Werte – also die Vorkommenshäufigkeiten der Begriffe in den Dokumenten – unmittelbar als Komponenten w_{dk} in den Vektor-Repräsentationen verwenden, dann können wir die obigen Dokumente und die Anfrage durch folgende Vektoren repräsentieren: $\mathcal{D}_1 = (1;1;0;0)$, $\mathcal{D}_2 = (1;2;0;0)$, $\mathcal{D}_3 = (1;1;1;0)$, $\mathcal{D}_4 = (0;1;1;0)$, $\mathcal{D}_5 = (1;0;1;1)$ und $\mathcal{Q} = (1;1;0;0)$.

Wenden wir nun das oben definierte Ähnlichkeitsmaß $similarity(\mathcal{Q}, \mathcal{D})$ auf diese Vektoren an, so erhalten wir folgende Werte: $similarity(\mathcal{Q}, \mathcal{D}_1) = 2$, $similarity(\mathcal{Q}, \mathcal{D}_2) = 3$, $similarity(\mathcal{Q}, \mathcal{D}_3) = 2$, $similarity(\mathcal{Q}, \mathcal{D}_4) = 1$ und $similarity(\mathcal{Q}, \mathcal{D}_5) = 1$.

Die Werte für $similarity(\mathcal{Q}, \mathcal{D})$ erlauben uns nun ein Ranking der Dokumente im Hinblick auf die Anfrage Q vorzunehmen. Dokument D_2 belegt Rang 1, die Dokumente D_1 und D_3 belegen gemeinsam den zweiten Rang und die Dokumente D_4 und D_5 belegen ebenfalls gemeinsam den dritten Rang.

Das Ergebnis ist in diesem Beispiel offensichtlich noch sehr grob. Insbesondere die direkte Nutzung der tf_{dk} Werte für die Vektorkomponenten w_{dk} erscheint verbesserungsbedürftig. Wir werden daher im Folgenden feinere Ansätze zur Bestimmung der Vektorkomponenten betrachten, die neben der Vorkommenshäufigkeit der Terme in einem Dokument weitere Faktoren wie z.B. die Länge des Dokumentes berücksichtigen, um zu schlüssigeren Ergebnissen zu kommen.

5.2 Die Basisformeln zum Vektorraummodell

Berechnung der Termgewichte w_{dk}

In der sehr umfangreichen Literatur zum Vektorraummodell wurden zahlreiche oft nur in Details abweichende Formeln zur Bestimmung der Werte für die Termgewichte w_{dk} vorgeschlagen (vgl. hierzu z.B.

[SB88]). Im Folgenden wollen wir das gebräuchlichste dieser Modelle betrachten. Das Vektorraummodell geht dabei, wie wir oben bereits gesehen haben, davon aus, dass ein Dokument durch einen Vektor $\mathcal{D} = (w_{d1}; w_{d2}; \ldots; w_{dt})$ repräsentiert wird. Jede Komponente w_{dk} sollte nun angeben, wie gut der Term k das Dokument D beschreibt.

Man nimmt nun an, dass zwei Faktoren hierauf Einfluss haben:

1. Die Vorkommenshäufigkeit des Terms im Dokument

Vorkommenshäufigkeit

Je höher die **Vorkommenshäufigkeit** eines Terms in einem Dokument ist, desto besser beschreibt der Term das Dokument.

Man kann diese Häufigkeit durch den Ausdruck tf_{dk} berücksichtigen. tf steht hier für die Termfrequenz, d identifiziert das entsprechende Dokument und k identifiziert das Wort aus dem Vokabular.

Allerdings liegt der absolute Wert der Vorkommenshäufigkeit für einzelne Terme bei langen Dokumenten deutlich höher als bei kurzen. Das heißt, lange Dokumente würden gegenüber kurzen Dokumenten bevorzugt, wenn man w_{dk} einfach gleich tf_{dk} setzt.

Beispiel

Wir gehen von einem Indexierungsvokabular mit den Termen *Haus*, *Garten*, *Italien*, *Frankreich* aus.

Ein Dokument D_1 ist recht lang und enthält 50-mal den Begriff »Haus« und 5-mal den Begriff »Garten«. Damit erhalten wir als Vektor $\mathcal{D}_1 = (50; 5; 0; 0)$.

Ein zweites Dokument D_2 ist dagegen recht kurz und enthält 2-mal den Begriff »Garten« und 2-mal den Begriff »Italien«. Damit erhalten wir als Vektor $\mathcal{D}_2 = (0; 2; 2; 0)$.

Eine Anfrage enthält ungewichtet die Begriffe »Garten« und »Italien«. Der entsprechende Anfragevektor lautet damit $\mathcal{Q} = (0; 1; 1; 0)$.

Das Skalarprodukt ist nun für D_1 gleich 5 und für D_2 gleich 4, obwohl D_2 sicher einschlägiger ist!

Normierung

Einen Ausweg aus dieser Problematik bildet die **Normierung der Vek-**

toren. Dazu wird jeder Wert durch die Wurzel aus der Summe der Quadrate über alle Werte dividiert:

$$w_{dk} = \frac{tf_{dk}}{\sqrt{\sum_{i=1}^{t} tf_{di}^2}} \tag{5.2}$$

Man beachte, dass der Nenner des obigen Bruchs $(\sum_{i\in 1,\ldots,t} tf_{di}^2)^{0,5}$ der Länge des Vektors entspricht. Eine Division aller Vektorkomponenten durch diese Länge führt dazu, dass alle Beschreibungsvektoren die Länge 1 haben.

Beispiel

Kommen wir zu unserem Beispiel mit den Dokumenten bzw. Beschreibungsvektoren $\mathcal{D}_1 = (50; 5; 0; 0)$ und $\mathcal{D}_2 = (0; 2; 2; 0)$ sowie dem Anfragevektor $\mathcal{Q} = (0; 1; 1; 0)$ zurück.

Als Normierungsfaktor für den ersten Vektor erhalten wir $(50^2 + 5^2 + 0^2 + 0^2)^{0,5} = (2500 + 25 + 0 + 0)^{0,5} = 2525^{0,5} \approx 50,25$. Wir müssen also in $\mathcal{D}_1$ alle Komponenten durch $50,25$ dividieren und erhalten als neuen Beschreibungsvektor $\mathcal{D}_1 = (0,995; 0,0995; 0; 0)$.

Als Normierungsfaktor für den zweiten Vektor erhalten wir $(0^2 + 2^2 + 2^2 + 0^2)^{0,5} = (0 + 4 + 4 + 0)^{0,5} = 8^{0,5} \approx 2,83$. Wir müssen also in $\mathcal{D}_2$ alle Komponenten durch $2,83$ dividieren und erhalten als neuen Beschreibungsvektor $\mathcal{D}_2 = (0; 0,707; 0,707; 0)$.

Das Skalarprodukt mit dem Anfragevektor $\mathcal{Q} = (0; 1; 1; 0)$ ist nun für D_1 gleich $0,0995$ und für D_2 gleich $1,41$. Damit erscheint D_2 nun wesentlich relevanter.

Das Ziel der Normierung ist damit, den Einfluss der Länge der Dokumente auf die Werte für $similarity(\mathcal{Q}, \mathcal{D})$ zu eliminieren. Hierzu werden bildlich gesprochen alle Beschreibungsvektoren auf die Oberfläche der t-dimensionalen Einheitskugel (Radius = 1) reduziert, so dass die Länge aller Beschreibungsvektoren einheitlich gleich 1 ist. Ein Nebeneffekt, der bei der Implementierung des Vektorraummodells ausgenutzt werden kann, ist, dass durch die Normierung sicher für alle Komponenten der Beschreibungsvektoren $0 \leq w_{dk} \leq 1$ gilt.

2. Die Trennschärfe des Terms bezogen auf die Dokumentenkollektion

Bisher basiert die Berechnung der w_{dk} Werte für ein Dokument D nur auf den Vorkommenshäufigkeiten der Begriffe in diesem Dokument. Nun kann aber ein Begriff sicher um so besser zur Charakterisierung eines Dokumentes beitragen, je seltener er in anderen Dokumenten auftritt. Er ist dann spezifischer für die Dokumente, in denen er auftritt und sollte entsprechend höher gewichtet werden. Man spricht hier auch von der **Trennschärfe** der Begriffe.

Trennschärfe

Man könnte als Kennzahl für die Trennschärfe des Terms i nun den Quotienten $\frac{N}{n_i}$ verwenden.

Beispiel

Sei $N = 100000$, $n_i = 5$ und $n_j = 50$. Das heißt, der Term i kommt in 5 und der Term j in 50 Dokumenten vor. In diesem Fall wäre der Korrekturfaktor für Term i gleich $\frac{N}{n_i} = 20000$ und für Term j gleich $\frac{N}{n_j} = 2000$. Dadurch würden bei Verwendung von

Korrekturfaktor

$$w_{dk} = \frac{tf_{dk} \cdot \frac{N}{n_k}}{\sqrt{\sum_{i=1}^{t} \left(tf_{di} \cdot \frac{N}{n_i}\right)^2}} \tag{5.3}$$

die sehr seltenen Terme die Vektoren völlig dominieren. Zur Verdeutlichung dieses Effektes zeigt die Abbildung 5.1 die Werte von $\frac{N}{n_k}$ in Abhängigkeit von n_k.

Einen Ausweg bildet nun die Verwendung des Logarithmus, die den skizzierten Effekt abmildert. Man verwendet also $\log(\frac{N}{n_k})$ als Korrekturfaktor zur Berücksichtigung der Trennschärfe der Begriffe. »log« verwenden wir dabei wie in technischen Disziplinen üblich für den dekadischen Logarithmus zur Basis 10.[1]

Damit ist im obigen Beispiel der Korrekturfaktor für Term i gleich $\log(20000) = 4,30$ und für Term j gleich $\log(2000) = 3,30$.

[1] Wie bereits auf Seite 49 angemerkt, könnte man ebenso den natürlichen Logarithmus (zur Basis $e = 2,71828\ldots$) verwenden.

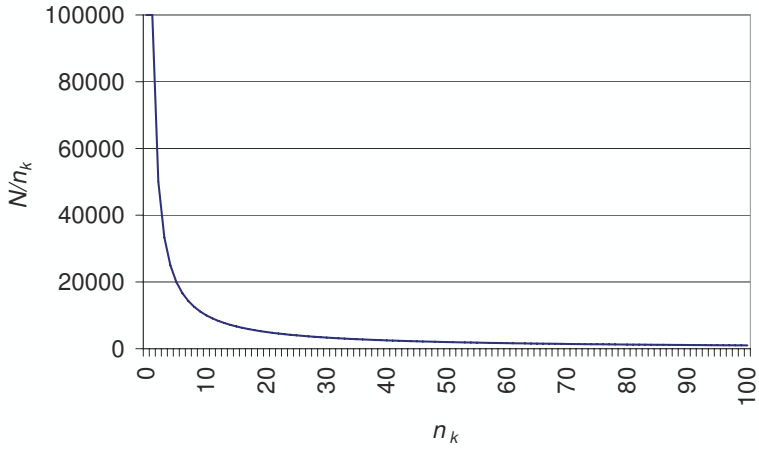


Abbildung 5.1 — Werte von $\frac{N}{n_k}$ in Abhängigkeit von n_k ($N = 100000$)

Die Abbildung 5.2 zeigt die Werte von $\log(\frac{N}{n_k})$ in Abhängigkeit von n_k. Im Vergleich zu Abbildung 5.1 werden dabei die deutlich geringeren Unterschiede zwischen den Werten sichtbar.

Durch Verwendung von $\log(\frac{N}{n_k})$ ergibt sich somit ein plausibel erscheinender Korrekturfaktor zur Berücksichtigung der Trennschärfe der Begriffe.

Die *tf·idf*-Formel

Wir erhalten damit als vorläufige Endfassung der Formel zur Berechnung der w_{dk} Werte:

$$w_{dk} = \frac{tf_{dk} \cdot \log \frac{N}{n_k}}{\sqrt{\sum_{i=1}^{t} \left(tf_{di} \cdot \log \frac{N}{n_i} \right)^2}} \tag{5.4}$$

Diese Formel wird in der Literatur häufig als *tf·idf*-Formel bezeichnet: *tf·idf-Formel*

- *tf* steht dabei für *term frequency* bzw. Termfrequenz und

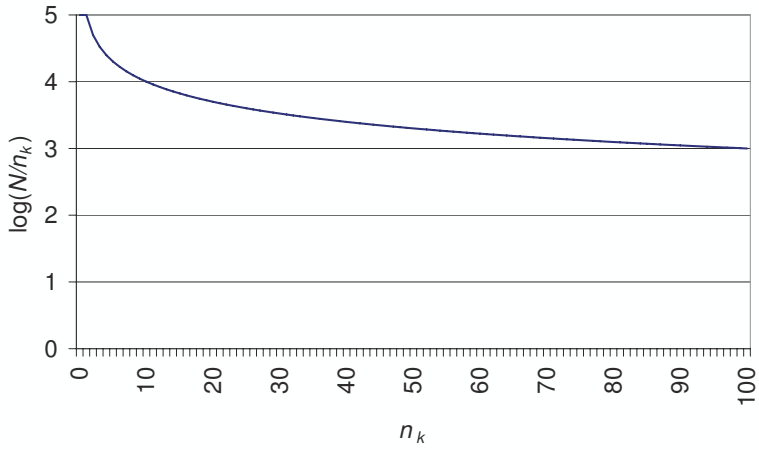


Abbildung 5.2 — Werte von $\log(\frac{N}{n_k})$ in Abhängigkeit von n_k ($N = 100000$)

- *idf* steht für *inverse document frequency*, das heißt, je seltener ein Begriff in den Dokumenten vorkommt, desto höher ist sein *idf* Wert.

Die *tf*·*idf*-Formel ist dabei als heuristische Formel zu verstehen. Eine strenge formale Begründung, warum gerade diese Formel gut arbeitet, kann nicht gegeben werden. Sie hat sich aber in dieser und ähnlichen Formen in zahlreichen empirischen Untersuchungen bewährt.

Die *tf*·*idf*-Formel stellt auch den Ausgangspunkt für zahlreiche weiterführende Ansätze im Information Retrieval dar. In die Formel können z.B. noch weitere Faktoren mit aufgenommen werden. So lassen sich bspw. Wörter, die in Überschriften vorkommen sowie Wörter, die kursiv oder fett dargestellt sind, höher gewichten als die restlichen Wörter eines Dokumentes. Dies könnte bereits bei der Bestimmung modifizierter tf_{dk} Werte berücksichtigt werden, indem Termen in Überschriften das Gewicht 5, fett gedruckten Termen das Gewicht 3, kursiv gedruckten Termen das Gewicht 2 und sonstigen Termen das Gewicht 1 zugeordnet wird.

Eine konkrete Variante der *tf*·*idf*-Formel werden wir in Abschnitt 5.3 exemplarisch betrachten. Zuvor wollen wir uns aber der Bestimmung der Anfragevektoren zuwenden.

Repräsentation der Anfragen

Eine Anfrage Q wird ebenfalls durch einen Vektor repräsentiert: $\mathcal{Q} = (w_{q1}; w_{q2}; \ldots; w_{qt})$

Wenn die Anfrage als Text gegeben ist – wovon man beim Vektorraummodell i.d.R. ausgeht – könnten wir zur Gewinnung der w_{qk} auf die *tf·idf*-Formel zurückgreifen.

Dies erscheint aber aus mehreren Gründen nicht sinnvoll:

1. Sofern ein Term in der Anfrage vorkommt, sollte man diesem immer ein gewisses Grundgewicht beimessen.
2. Eine Normierung der Vektoren ist hier nicht erforderlich, weil ja für alle Dokumente der gleiche Anfragevektor verwendet wird.

Salton und Buckley [SB88] schlagen daher die folgende Formel vor:

$$w_{qk} = \begin{cases} \left(0.5 + \frac{0.5 \cdot tf_{qk}}{\max\limits_{1 \le i \le t} tf_{qi}}\right) \cdot \log \frac{N}{n_k} & \text{falls} \quad tf_{qk} > 0 \\ 0 & \text{falls} \quad tf_{qk} = 0 \end{cases} \tag{5.5}$$

Durch diese Berechnung werden alle Begriffe, die in der Anfrage vorkommen, mit einem Grundgewicht von 0,5 versehen, weil man davon ausgeht, dass Terme, die in der Anfrage auftreten, grundsätzlich wichtig für diese Anfrage sind. Ein weiteres Gewicht von maximal 0,5 ergibt sich durch die gewichtete Betrachtung der Termfrequenz in der obigen Formel. Schließlich wird auch in der Anfrage noch eine Gewichtung mit $\log(\frac{N}{n_k})$ vorgenommen, um die Trennschärfe der Terme zu berücksichtigen.

Ähnlichkeit zwischen Dokumenten und Anfrage

Die Ähnlichkeit zwischen einer Anfrage und einem Dokument wird wie bereits erwähnt üblicherweise durch das Skalarprodukt (Formel 5.1) beschrieben:

Skalarprodukt

$$similarity(\mathcal{Q}, \mathcal{D}) = \sum_{k=1}^{t} w_{qk} \cdot w_{dk}$$

Grob gesprochen werden hier die w_{dk} Werte der Dokumente mit den w_{qk} Werten der Anfrage gewichtet.

Cosinusmaß

Für die Interpretation dieses »Ähnlichkeitsmaßes«, das beim Vektorraummodell häufig als Retrievalfunktion verwendet wird, ist die Beziehung zum so genannten *Cosinusmaß* wichtig. Das Cosinusmaß bestimmt den Winkel zwischen zwei Vektoren – oder präziser dessen Cosinuswert.

Nehmen wir hierzu zunächst an, dass wir bei den Beschreibungsvektoren für die Dokumente auf die Normierung verzichten und statt dessen $w'_{dk} = tf_{dk} \cdot \log(\frac{N}{n_k})$ verwenden.

Ähnlichkeitsmaß

Als Ähnlichkeitsmaß kann dann das Cosinusmaß verwendet werden, das den Winkel zwischen dem Dokumentvektor und dem Anfragevektor berechnet:

$$sim_{\cos}(\mathcal{Q}, \mathcal{D}) = \frac{\sum_{k=1}^{t} w_{qk} \cdot w'_{dk}}{\sqrt{\sum_{i=1}^{t} w_{qi}^2} \cdot \sqrt{\sum_{j=1}^{t} w_{dj}'^2}} \tag{5.6}$$

Im Zähler des Bruchs sehen wir dabei das normale Skalarprodukt. Im Nenner werden bei der Berechnung des Cosinusmaßes für den Anfrage- und den Dokumentvektor Normierungen durchgeführt. Durch diese Normierungen ist das Cosinusmaß unabhängig von der Länge der Vektoren. Die Normierung wird hier von der Erstellung der Vektoren auf die Berechnung der Ähnlichkeit verlagert und auch für den Anfragevektor durchgeführt.

Da bei der Ähnlichkeitsberechnung der Normierungsfaktor für die Anfrage $(\sum_{i \in 1,\ldots,t} w_{qi}^2)^{0,5}$ für alle Dokumente gleich ist, ergibt sich nun beim Cosinusmaß die gleiche Rangordnung unter den Dokumenten wie bei normierten Dokumentvektoren und Verwendung des Skalarproduktes.

Die Rangordnung der Dokumente wird somit bei Verwendung des Skalarproduktes nach der Größe des Winkels zwischen dem Anfragevektor und den Dokumentvektoren vorgenommen.

5.3 Eine Variante der tf·idf-Formel

Variante der tf·idf-Formel

Wie bereits mehrfach erwähnt, gibt es in der Literatur eine Vielzahl von Detailvorschlägen zur Verbesserung der $tf \cdot idf$-Formel . Wir wollen hier exemplarisch einen dieser Vorschläge herausgreifen, der sich mit der Normierung der Beschreibungsvektoren beschäftigt [SBM96].

Wir haben die Normierung der Beschreibungsvektoren in der $tf \cdot idf$-Formel ja damit begründet, dass kurze Dokumente ohne diese Normierung benachteiligt wären, weil in langen Dokumenten die einzelnen Begriffe naturgemäß häufiger auftreten. Nun enthalten kurze Dokumente aber im Allgemeinen recht wenige Terme und ihre Beschreibungsvektoren damit nur recht wenige Komponenten mit $w_{dk} > 0$. Diese Komponenten werden nun durch die Normierung weniger stark betroffen als die Komponenten in den Beschreibungsvektoren langer Dokumente.

Etwas vereinfacht dargestellt ergibt sich nach der Normierung dadurch, dass alle Beschreibungsvektoren gleich lang sind, folgende Situation:

- Die Beschreibungsvektoren für kurze Dokumente enthalten nur wenige Komponenten mit $w_{dk} > 0$. Diese Komponenten haben dafür aber relativ große Werte, weil die normierte Länge von 1 nur zwischen diesen wenigen Komponenten aufgeteilt werden muss.
- Die Beschreibungsvektoren für lange Dokumente enthalten hingegen mehr Komponenten mit $w_{dk} > 0$. Diese Komponenten haben aber relativ kleine Werte, weil die normierte Länge von 1 hier zwischen einer größeren Zahl von Komponenten aufgeteilt werden muss.

Werden in einer Anfrage nun nur wenige Terme vorgegeben, so werden die Dokumente zuerst ermittelt, die für diese Terme hohe w_{dk} Werte haben. Das werden aber eher kurze als lange Dokumente sein.

Damit stellt sich die Frage ob eine vollständige Normierung der Beschreibungsvektoren nicht »zu viel des Guten« ist.

Beispiel

TREC-Kollektion

Sehen wir uns dazu ein Beispiel an. Als Testkollektion kommen dabei die CDs 1 und 2 der TREC-Kollektion mit insgesamt 741856 Dokumenten

und die Topics 151 bis 200 zum Einsatz.

Die Abbildung 5.3 stellt nun in Abhängigkeit von der Dokumentlänge die Wahrscheinlichkeit dafür, dass ein Dokument relevant ist, und die Wahrscheinlichkeit dafür, dass ein Dokument bei Verwendung der klassischen $tf \cdot idf$-Formel für relevant gehalten wird, gegenüber.

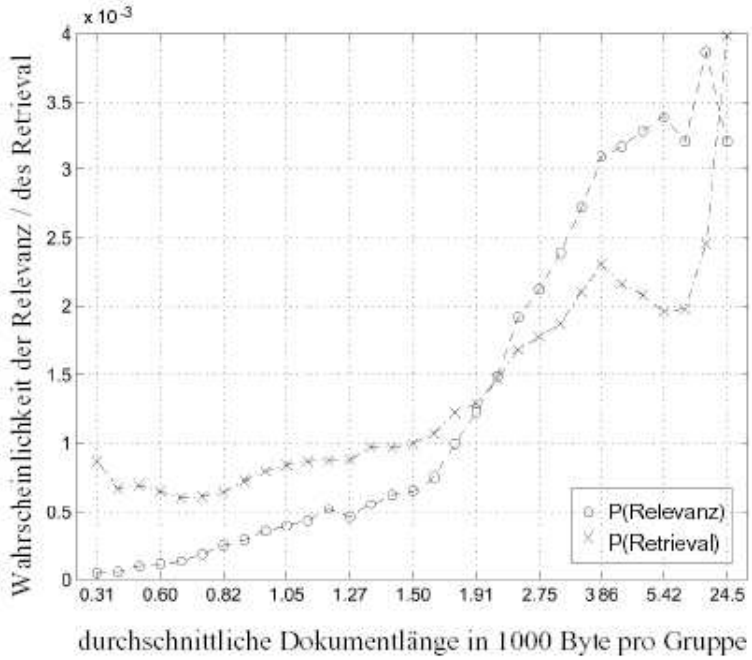


Abbildung 5.3 — Wahrscheinlichkeit für Relevanz und Wahrscheinlichkeit dafür, dass ein Dokument mit der klassischen $tf \cdot idf$-Formel gefunden wird, im Verhältnis zur Größe des Dokumentes [SBM96]

Wir können nun folgenden Effekt beobachten:

- Für kurze Dokumente ist die Wahrscheinlichkeit, dass sie im Ergebnis einer Anfrage enthalten sind, höher als die Wahrscheinlichkeit, dass sie für eine Anfrage relevant sind.
- Für lange Dokumente ist es genau umgekehrt.

Dies kann als Indiz dafür betrachtet werden, dass die Normierung der Beschreibungsvektoren zu weit geht. Es sollte also eine Abschwächung der Normierung vorgenommen werden.

Anpassung der Normierung

Anpassung der Normierung

Man sollte also den Normierungsfaktor

$$\sqrt{\sum_{i=1}^{t} \left(tf_{di} \cdot \log \frac{N}{n_i} \right)^2} \tag{5.7}$$

- für lange Dokumente etwas abschwächen und damit die w_{dk} Werte tendenziell etwas anheben und
- die Normierung für kurze Dokumente etwas verschärfen und damit die w_{dk} Werte tendenziell etwas senken.

Dazu wird nun Formel 5.8 verwendet, in der der in Formel 5.7 angegebene alte Normierungsfaktor als *old normalization* bezeichnet wird. Zusätzlich wird mit *average old normalization* der Durchschnitt der Normierungsfaktoren über alle Dokumente hinweg bezeichnet. Der Parameter *slope* ist dabei geeignet zu wählen.

$$kf_d = (1,0 - \text{slope}) + \text{slope} \cdot \frac{\text{old normalization}}{\text{average old normalization}} \tag{5.8}$$

Statt der herkömmlichen w_{dk} Werte werden im Beschreibungsvektor für ein Dokument D nun die Werte $kf_d \cdot w_{dk}$ verwendet. Oder anders ausgedrückt, wird der bisherige Normierungsfaktor durch kf_d dividiert.

Beispiel

Folgende Tabelle verdeutlicht den Effekt für $slope = 0,75$ und *average old normalization* $= 40$. Wir betrachten für diese Konstellation drei Beispieldokumente:

Dokument	old normalization	kf_d	new normalization	Länge des Vektors
D_1	20	0,65	30,77	0,65
D_2	40	1,0	40	1,0
D_3	80	1,7	47,06	1,7

Die Komponenten des relativ kurzen Dokumentes D_1 werden also mit dem Korrekturfaktor 0,65 multipliziert. Dadurch ergibt sich für solche kurzen Dokumente eine stärkere Normierung – im Beispiel werden die $w_{dk} \cdot w_{qk}$ Werte nun durch 30,77 statt bisher durch 20 geteilt.

Die Komponenten des relativ langen Dokumentes D_3 werden dagegen mit dem Korrekturfaktor 1,7 multipliziert. Dadurch ergibt sich für lange Dokumente eine schwächere Normierung von 47,06 im Verhältnis zu vorher 80. Lange Dokumente haben damit wieder längere Beschreibungsvektoren als kurze.

Bestimmung des *slope*-Wertes

slope-Wert

Der beste Wert für *slope* muss nun empirisch bestimmt werden.

Für die CDs 1 und 2 von TREC mit 741856 Dokumenten ergab sich eine *average old normalization* von 13,36.

In der folgenden Tabelle sind experimentelle Ergebnisse angegeben, die sich mit den Topics 151 bis 200 ergeben:

	tf·idf-Formel	modifizierter Ansatz				
slope	–	**0,60**	**0,65**	**0,70**	**0,75**	**0,80**
gefunden	6526	6342	6458	6574	6629	6671
Precision	0,2840	0,3024	0,3097	0,3144	0,3171	0,3162
Verbesserung	–	+6,5%	+9,0%	+10,7%	+11,7%	+11,3%

Die Zeilen der Tabelle sind wie folgt zu interpretieren:

- Die Werte in der Zeile **gefunden** stehen für die Anzahl der über alle Topics hinweg gefundenen relevanten Dokumente (von 9805 relevanten Dokumenten insgesamt).
- Die Werte in der Zeile **Precision** stehen für die durchschnittliche Precision.
- Die Werte in der Zeile **Verbesserung** stehen für die Verbesserung der durchschnittlichen Precision.

Man sieht, dass sich mit der modifizierten Methode deutlich bessere Precision-Werte erzielen lassen. Für die gegebenen Dokumente ist der Wert $slope = 0,75$ am günstigsten.

Zusammenfassung

Das beschriebene Verfahren versucht durch eine modifizierte Normierung die Wahrscheinlichkeit der Relevanz und die Wahrscheinlichkeit des Retrieval anzugleichen. Experimente mit der TREC-Kollektion deuten an, dass dies die Precision erhöhen kann.

Dabei sind aber folgende Aspekte zu beachten:

- Durch die modifizierte Normierung gilt nicht mehr sicher $w_{dk} \leq 1$. Alle Zugriffstrukturen und Algorithmen, die diese Eigenschaft der klassischen $tf \cdot idf$-Formel ausnutzen, müssen entsprechend angepasst werden.
- Der Wert für *slope* muss empirisch ermittelt werden. Ob es eine allgemein sinnvolle Belegung für *slope* gibt, ist offen.

5.4 Relevance Feedback

Das Vektorraummodell in der bisher betrachteten Form geht – wie auch die zuvor betrachteten einfachen IR-Modelle – davon aus, dass ein Benutzer seinen Informationswunsch als Anfrage an ein entsprechendes IR-System formuliert. Diese Anfrage ist aber nur eine unvollkommene Beschreibung bzw. Abbildung des Informationswunsches. Eine Idee ist nun, weitere Informationsquellen über diesen Informationswunsch zur Verbesserung der Ergebnisse zu nutzen.

Informationswunsch

Eine mögliche Quelle für die Verbesserung der Ergebnisse stellt die Beurteilung der von dem IR-System gelieferten Ergebnisse durch den Benutzer dar. Dabei stellt der Benutzer – wie bisher auch – zuerst eine Anfrage an das System. Die von dem System in einem ersten Durchlauf gelieferten Ergebnisse kann der Benutzer dann im Hinblick auf ihre Relevanz für seinen eigentlichen Informationswunsch bewerten. Aus dieser Bewertung reformuliert das System die Anfrage und liefert dem Benutzer ein neues Ergebnis.

Die Beurteilung des ersten Ergebnisses im Hinblick auf die Relevanz für den eigentlichen Informationswunsch erfolgt dabei üblicherweise dadurch, dass der Benutzer Elemente dieses Ergebnisses gezielt als relevant oder auch als nicht relevant kennzeichnen kann. Der Anfragevektor $\mathcal{Q}$ wird dann zur besseren Repräsentation des Informationswunsches von den Beschreibungsvektoren der gefundenen nicht relevanten Dokumente weg und in Richtung der gefundenen relevanten Dokumente verschoben.

Relevance Feedback

Zum besseren Verständnis der Verfahren zum Relevance Feedback ist dabei Abbildung 5.4 der Vektoraddition hilfreich.

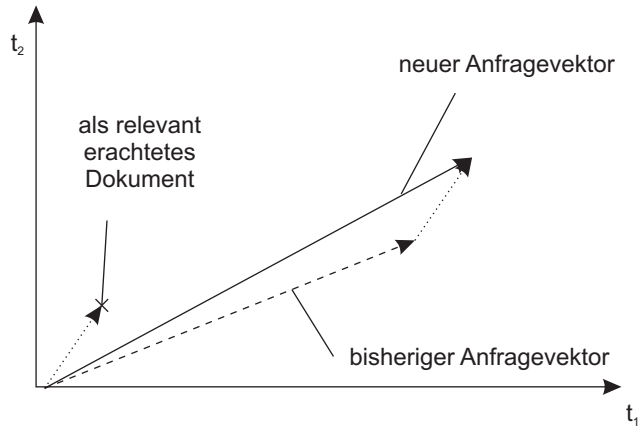


Abbildung 5.4 — Addition eines Dokumentvektors zum Anfragevektor.

Abbildung 5.4 geht von einem Indexierungsvokabular mit 2 Termen aus ($t = 2$). In der Abbildung ist dabei zunächst gestrichelt der alte Anfragevektor und gepunktet der Beschreibungsvektor eines als relevant erachteten Dokuments eingezeichnet. Addiert man nun den Vektor dieses Dokuments zum alten Anfragevektor hinzu, so erhält man den neuen Anfragevektor. Bedenkt man nun, dass wir uns bei der Ähnlichkeitsbe-

trachtung auf den Winkel zwischen den Vektoren beziehen und die Länge des Anfragevektors vernachlässigen, so wird deutlich, dass die Addition eines Vektors $\mathcal{X}$ zu einem Vektor $\mathcal{Y}$ immer ein Verschieben von $\mathcal{Y}$ in Richtung $\mathcal{X}$ bedeutet.

Methoden zur Anpassung der Anfragevektoren

In der Literatur wurden zahlreiche Formeln zur Umsetzung der Idee des Relevance Feedback vorgestellt, von denen wir hier exemplarisch die in [SB90] verglichenen darstellen wollen.

Wir gehen dabei davon aus, dass die bisher vom Anfragesteller betrachteten Dokumente (die so genannte *Feedbackmenge*) in zwei Mengen F^+ und F^- zerfallen. F^+ enthält die als relevant eingestuften Dokumente und F^- die als nicht relevant eingestuften. Gegebenenfalls können auch Dokumente in eine dritte Menge $F^?$ eingeordnet werden; dies sind dann Dokumente, die weder als relevant noch als irrelevant betrachtet werden.

Im Folgenden betrachten wir nun mehrere Methoden wie der Anfragevektor auf Basis dieser beiden Mengen angepasst werden kann.

Ide (dec hi)

Ide (dec hi)

Die Beschreibungsvektoren der Dokumente aus F^+ werden hier zum Anfragevektor hinzuaddiert. Ferner wird der Beschreibungsvektor $\mathcal{D}_1^{\overline{rel}}$ des von der bisherigen Anfrage am höchsten eingeschätzten Dokumentes aus F^- von dem entstehenden Anfragevektor abgezogen. Damit wird der Anfragevektor von diesem nicht relevanten Dokument weg und zu den relevanten Dokumenten hin verschoben.

$$\mathcal{Q}_{neu} = \mathcal{Q}_{alt} + \left(\sum_{\mathcal{D} \in F^+} \mathcal{D} \right) - \mathcal{D}_1^{\overline{rel}} \tag{5.9}$$

Ide (regular)

Ide (regular)

Hier werden im Gegensatz zu Ide (dec hi) alle Dokumente aus F^- berücksichtigt.

$$\mathcal{Q}_{neu} = \mathcal{Q}_{alt} + \left(\sum_{\mathcal{D} \in F^+} \mathcal{D} \right) - \left(\sum_{\mathcal{D} \in F^-} \mathcal{D} \right) \tag{5.10}$$

Rocchio

Rocchio

An der als Ide (regular) vorgestellten Formel werden hier einige Verfeinerungen vorgenommen.

Ein erster Schritt ist die Gewichtung der in F^+ und F^- enthaltenen Dokumente (bzw. ihrer Dokumentvektoren) mit den Kardinalitäten von F^+ und F^-.

Darüberhinaus gewichtet man den Einfluss der als relevant erachteten Dokumente mit β und den der nicht relevant erachteten Dokumente mit α. Dabei wird meist gefordert, dass $\beta + \alpha = 1$ gelten muss. Typische Belegungen sind bspw. $\alpha = 0,25$ und $\beta = 0,75$, das heißt die relevanten Dokumente werden stärker gewichtet als die nicht relevanten.

$$\mathcal{Q}_{neu} = \mathcal{Q}_{alt} + \beta \cdot \left(\sum_{\mathcal{D} \in F^+} \frac{\mathcal{D}}{|F^+|} \right) - \alpha \cdot \left(\sum_{\mathcal{D} \in F^-} \frac{\mathcal{D}}{|F^-|} \right) \tag{5.11}$$

Experimentelle Ergebnisse

Wir betrachten nun die von Salton in [SB90] beschriebenen experimentellen Ergebnisse für die oben skizzierten Verfahren des Relevance Feedback.

Die in Tabelle 5.1 aufgeführten Precision-Werte sind Mittelwerte über die Recall-Punkte 0,75, 0,5 und 0,25.

Zur Durchführung des Relevance Feedbacks wurden die 15 Dokumente verwendet, die von der initialen Suche mittels des klassischen Vektorraummodells als »relevanteste« Dokumente ermittelt wurden. Diese 15 Dokumente wurden weder bei der initialen noch bei den nachfolgenden Suchen bei der Berechnung von Recall und Precision berücksichtigt, um einen fairen Vergleich zu ermöglichen.

Die Ergebnisse unter der Angabe *ausgewählte Terme* berücksichtigen beim Relevance Feedback jeweils nur die Terme, die in vielen relevanten

eingesetzte Methode		CACM 3204 Dok. 64 Anfr.	CISI 1460 Dok. 112 Anfr.	CRAN 1397 Dok. 225 Anfr.	INSPEC 12684 Dok. 84 Anfr.	MED 1033 Dok. 30 Anfr.	Durchschnitt
initiale Anfrage							
	Precision	0,1459	0,1184	0,1156	0,1368	0,3346	
Ide (dec hi)							
mit allen Termen	Precision	0,2704	0,1742	0,3011	0,2140	0,6305	
	Verbesserung	+86%	+47%	+160%	+56%	+88%	+87%
ausgewählte Terme	Precision	0,2479	0,1924	0,2498	0,1976	0,6218	
	Verbesserung	+70%	+63%	+116%	+44%	+86%	+76%
Ide (regular)							
mit allen Termen	Precision	0,2241	0,1550	0,2508	0,1936	0,6228	
	Verbesserung	+66%	+31%	+117%	+42%	+86%	+68%
ausgewählte Terme	Precision	0,2179	0,1704	0,2217	0,1808	0,5980	
	Verbesserung	+49%	+44%	+92%	+32%	+79%	+59%
Rocchio (standard $\beta = 0,75$***;*** $\alpha = 0,25$***)***							
mit allen Termen	Precision	0,2552	0,1404	0,2955	0,1821	0,5630	
	Verbesserung	+75%	+19%	+156%	+33%	+68%	+70%
ausgewählte Terme	Precision	0,2491	0,1623	0,2534	0,1861	0,5297	
	Verbesserung	+71%	+37%	+119%	+36%	+55%	+64%

Tabelle 5.1 — Experimentelle Ergebnisse zum Relevance Feedback

Dokumenten vorkommen. Ein pragmatischer Vorteil dieses Vorgehens ist, dass dies die Bearbeitung der Anfragen mit invertierten Listen beschleunigt (vgl. Abschnitt 5.5).

Tabelle 5.1 zeigt die Ergebnisse für die verschiedenen Testkollektionen.

Aus Tabelle 5.1 wird deutlich, dass sich durch den Einsatz von Relevance Feedback die Precision stark steigern lässt. Im Mittel bewegt sich die Verbesserung zwischen +59% und +87%. Damit ergeben sich insbesondere auch deutlich signifikantere Verbesserungen als bei einfachen Variationen der $tf \cdot idf$-Formel.

Ein Vergleich zwischen den Verfahren zeigt, dass Ide (dec hi) in diesem Experiment am besten abschneidet. Der Grund hierfür ist, dass die zu einer Anfrage als nicht relevant markierten Dokumente i.d.R. sehr heterogen sein werden. Zieht man die Beschreibungsvektoren dieser Dokumente nun vom ursprünglichen Anfragevektor ab, so ergibt sich keine homogene Verschiebung in eine Richtung sondern eine gleichzeitige Verschiebung in viele verschiedene Richtungen, die wenig zielführend ist. Die als relevant markierten Dokumente sind dagegen i.d.R. eher homogen, so

dass sich hier eine zielgerichtete Verschiebung des Anfragevektors ergibt. Es ist deshalb sehr umstritten, ob als nicht relevant markierte Dokumente überhaupt beim Relevance Feedback berücksichtigt werden sollten.

Schließlich wird aus den Ergebnissen auch deutlich, dass die Beschränkung des Relevance Feedback auf ausgewählte Terme nur eine geringfügig schlechtere Precision mit sich bringt.

5.5 Implementierung des Vektorraummodells

Implementierung des Vektorraummodell

Die Implementierung des Vektorraummodells kann auf mehrere Arten erfolgen:

- Die gebräuchliche Art ist die Implementierung durch invertierte Listen.
- Eine weitere Möglichkeit ist die Realisierung durch mehrdimensionale Zugriffsstrukturen [Hen96], die aber aufgrund der hohen Dimensionszahl beim Vektorraummodell problematisch ist.
- Zudem ist auch der Einsatz von Signaturen denkbar [SP93].

Wir wollen uns im Folgenden auf die übliche Realisierung mit invertierten Listen konzentrieren.

5.5.1 Basisalgorithmus mit invertierten Listen

Da das Vektorraummodell ein Ranking der Dokumente im Hinblick auf die Anfrage erstellt, geht man bei der Anfragebearbeitung üblicherweise davon aus, dass der Anfragende nur die γ relevantesten Dokumente sehen möchte. γ ist dabei ein frei zu wählender Parameter.

Basisalgorithmus

Der Basisalgorithmus zur Bestimmung der γ relevantesten Dokumente mit Hilfe invertierter Listen arbeitet wie folgt:

- Betrachte die invertierten Listen zu den Anfragetermen in der Reihenfolge fallender Fragetermgewichte w_{qk}.

- Für jedes Paar (D, w_{dk}) in der gerade betrachteten invertierten Liste addiere $w_{dk} \cdot w_{qk}$ zum bisherigen Gewicht von D. Dabei werden zu allen Dokumenten die bisher aufgelaufenen Retrievalgewichte festgehalten. Diese sind initial null.
- Jeweils nachdem eine invertierte Liste vollständig betrachtet worden ist, prüfe, ob es noch möglich ist, dass das Dokument mit dem aktuellen Rang $\gamma+1$ Rang γ erreichen kann. Wenn dies nicht mehr möglich ist, können wir an dieser Stelle abbrechen!

Beispiel

Gegeben seien das Vokabular {*Auto*, *Fahrrad*, *Kosten*, *Motorrad*, *Produktion*, *Qualität*} und der Anfragevektor (0; 0; 8; 1; 10; 0).

Die $\gamma+1$ Dokumente mit den höchsten bisher aufgelaufenen Retrievalgewichten werden in einem Array *TopDocs* nach dem bisher aufgelaufenen Retrievalgewicht sortiert verwaltet.

TopDocs

Zusätzlich wird zu jedem Dokument, für das ein Listeneintrag in einer invertierten Liste gelesen wurde, das bisher aufgelaufene Retrievalgewicht in einer Hilfsdatenstruktur verwaltet. Diese Hilfsdatenstruktur verwaltet im Gegensatz zu *TopDocs* alle bisher betrachteten Dokumente und ermöglicht einen schnellen Zugriff über die Dokument-ID während *TopDocs* nur maximal $\gamma + 1$ Dokumente enthält und diese nach der Gewichtung absteigend sortiert.

Hilfsdatenstruktur

Betrachten wir nun den Ablauf des Algorithmus auf den in Abbildung 5.5 dargestellten invertierten Listen für $\gamma = 2$.

Zuerst wird die Liste für den Term 5 = »Produktion« durchlaufen, da dessen Gewicht $w_{q;5} = 10$ im Anfragevektor am höchsten ist. Diese Liste ist in Abbildung 5.5 durch eine Umrahmung hervorgehoben. Im oberen Teil der Abbildung sind die invertierten Listen dargestellt und im unteren Teil die Entwicklung von *TopDocs* und der Hilfsdatenstruktur während der Bearbeitung der Liste.

Zunächst sind das Array *TopDocs* und die Hilfsdatenstruktur leer. Der erste Eintrag in der invertierten Liste zum Term »Produktion« beinhaltet nun das Paar $(D_1; 0,4)$, wobei die $0,4$ dem Wert für $w_{1;5}$ entspricht. Die Multiplikation $w_{1;5} \cdot w_{q;5}$ mit dem Anfragetermgewicht $w_{q;5}$ von 10 ergibt nun $4,0$. Dieser Wert wird zusammen mit der Dokument-ID in *TopDocs* eingetragen und stellt, da bisher nur ein Wert enthalten ist,

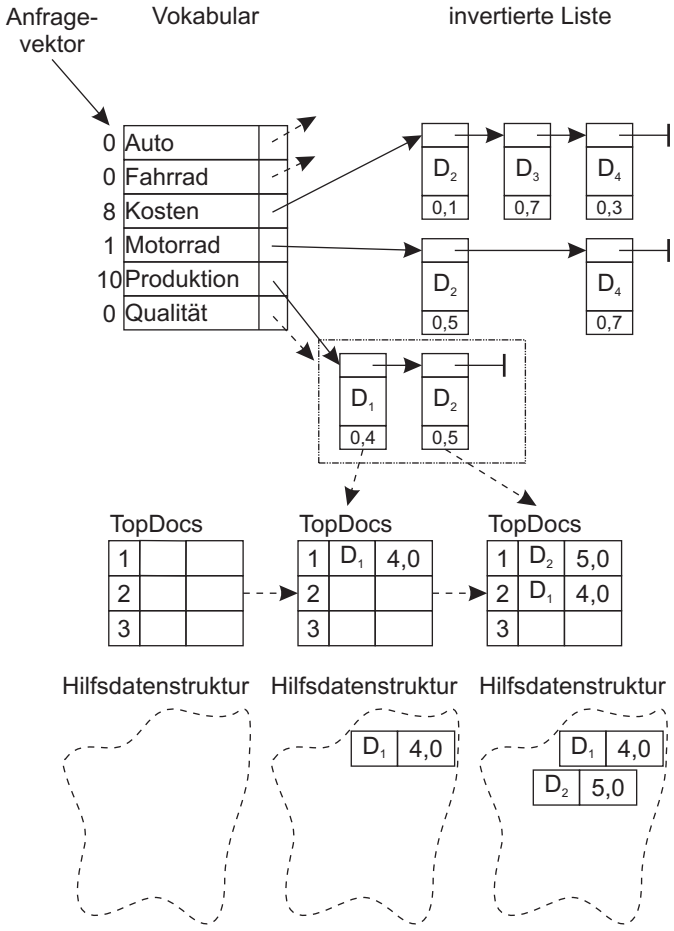


Abbildung 5.5 — Basisalgorithmus mit invertierten Listen – Schritt 1

den höchsten Eintrag dar. Zusätzlich wird in der Hilfsdatenstruktur ein Paar aus Dokument-ID und bisher aufgelaufenem Skalarprodukt, also in diesem Fall aus D_1 und $4,0$, vermerkt.

Nun betrachten wir den zweiten Listeneintrag für den Term »Produktion«, das Paar $(D_2; 0,5)$. Als Wert für $w_{2;5} \cdot w_{q;5}$ erhalten wir hier 5,0. Dieser Wert wird zusammen mit der Dokument-ID in *TopDocs* einsortiert, wo er noch vor dem Eintrag von D_1 den höchsten Rang annimmt. Gleichzeitig vermerken wir für D_2 das Paar aus Dokument-ID und bisher aufgelaufenem Retrievalgewicht in der Hilfsdatenstruktur.

Wir haben nun die invertierte Liste zum Term »Produktion« abgearbeitet und werfen einen Blick auf die durch die noch zu betrachtenden Anfrageterme maximal erreichbaren Gewichte. Dabei können wir ausnutzen, dass bei Verwendung der klassischen *tf*·*idf*-Formel durch die Normierung der Beschreibungsvektoren $w_{dk} \leq 1$ gilt.

Im Anfragevektor stehen nun noch die Gewichte 8 für den Term »Kosten« und 1 für den Term »Motorrad«. Dies bedeutet, dass weitere Dokumente durch die Betrachtung der invertierten Listen zu diesen beiden Termen durchaus noch ein Maximalgewicht von 9 erreichen könnten. Da das bisherige Höchstgewicht in *TopDocs* nur 5,0 beträgt, könnten weitere Dokumente somit durchaus noch einen vorderen Rang in *TopDocs* einnehmen. Wir können den Algorithmus an dieser Stelle also noch nicht abbrechen.

Wir betrachten deshalb die invertierte Liste zu Term 3 = »Kosten« (vgl. Abbildung 5.6).

Das erste Dokument in der invertierten Liste für den Term »Kosten« ist Dokument D_2. Wir berechnen für dieses Dokument nun $w_{2;3} \cdot w_{q;3}$ mit der Anfragetermgewichtung 8 und der Dokumenttermgewichtung 0,1 und erhalten 0,8. Da wir dieses Dokument bereits in unserer Hilfsdatenstruktur vorfinden, erhöhen wir die Gewichtung dort auf 5,8. Analog erhöhen wir die Gewichtung für dieses Dokument in *TopDocs* und aktualisieren die Sortierung in *TopDocs* – was in diesem Fall keine Veränderungen erbringt.

Als nächstes Dokument in der invertierten Liste zum Term »Kosten« finden wir das Dokument D_3. Wir berechnen auch für dieses Dokument den Beitrag $w_{3;3} \cdot w_{q;3}$ zum Skalarprodukt und fügen D_3 mit der resultierenden Gewichtung von $5,6$ sowohl in *TopDocs* als auch in die Hilfsdatenstruktur neu ein.

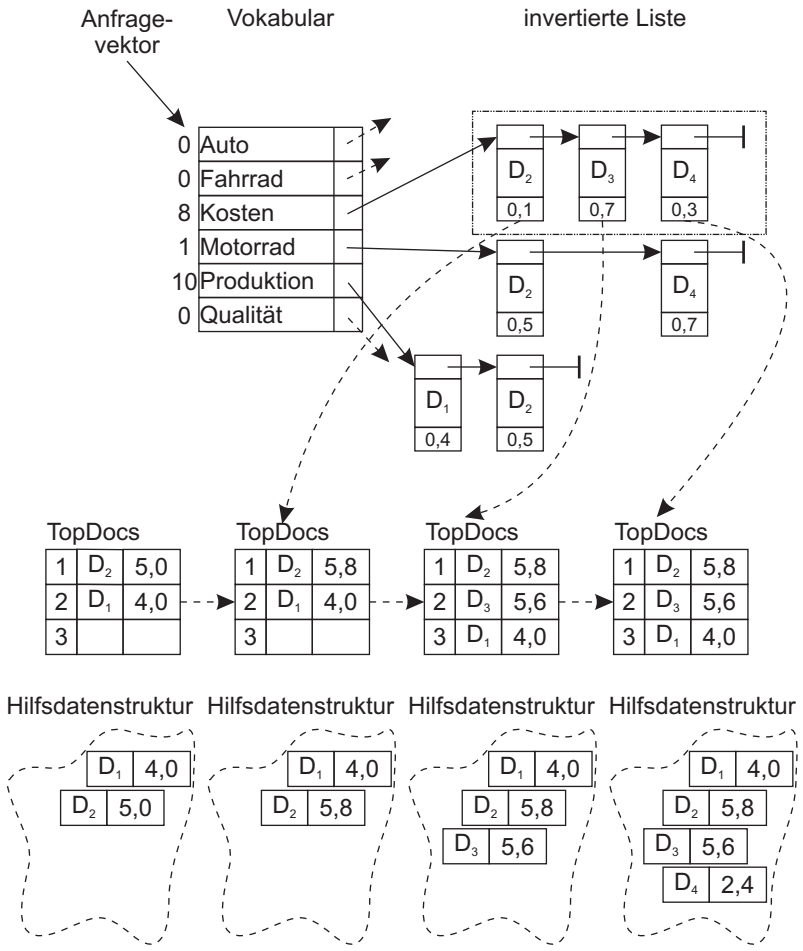


Abbildung 5.6 — Basisalgorithmus mit invertierten Listen – Schritt 2

Als letztes Dokument finden wir nun D_4 in der betrachteten invertierten Liste und berechnen auch hier den Beitrag $w_{4;3} \cdot w_{q;3}$ zum Skalarprodukt. Die resultierende Gewichtung von $2,4$ ist allerdings zu gering, als dass dieses Dokument in *TopDocs* aufgenommen werden könnte. Trotzdem wird es in die Hilfsdatenstruktur aufgenommen. Sein Gewicht könnte sich ja bei der Betrachtung weiterer Listen noch so erhöhen, dass es dennoch in *TopDocs* aufrücken müsste.

Wir haben nun auch das Ende der Liste für den Term »Kosten« erreicht. An dieser Stelle fragen wir uns wieder, ob wir die Liste zum jetzt noch ausstehenden Term »Motorrad« noch betrachten müssen. Da der Anfrageterm »Motorrad« eine Gewichtung von 1 hat, kann zu allen Einträgen in *TopDocs* maximal noch ein Gewicht von 1 hinzukommen. Da die Differenz zwischen dem vorletzten (Platz γ) und dem letzten (Platz $\gamma + 1$) Element in *TopDocs* aber $1,6$ ist, wissen wir, dass *TopDocs* sicher die γ besten Dokumente enthält. Es könnte allerdings sein, dass sich die Reihenfolge der γ besten Dokumente in *TopDocs* noch ändern würde, wenn bspw. D_3 deutlich mehr Gewicht gegenüber D_2 bekommen würde. Wollten wir diesbezüglich Sicherheit haben, müssten wir auch die invertierte Liste zum Term »Motorrad« betrachten. Buckley und Lewit [BL85], die den beschriebenen Algorithmus vorgeschlagen haben, plädieren aber dafür an dieser Stelle bereits abzubrechen, da nun bereits sicher die γ besten Dokumente gefunden sind und geringfügige Verschiebungen unter diesen Dokumenten den zusätzlichen Aufwand nicht rechtfertigen würden.

Implementierung des Algorithmus

Für die Implementierung des im Beispiel dargestellten Algorithmus benötigen wir einige Datenstrukturen und Operationen, die wir im Folgenden einführen.

Realisierung der Hilfsdatenstruktur

Realisierung der Hilfsdatenstruktur

Für die Implementierung des Algorithmus benötigen wir eine Hilfsdatenstruktur , die Paare (D, g) verwaltet. Diese bestehen aus einer Dokument-ID D und dem bisher aufgelaufenen Retrievalgewicht g.

Auf dieser Struktur seien folgende Operationen realisiert:

- $IsInDS(D)$ prüft für eine gegebene Dokument-ID D, ob ein Paar zu diesem Dokument in der Datenstruktur enthalten ist.

- $InsertIntoDS(D, g)$ fügt ein neues Paar in die Datenstruktur ein. Zuvor sollte kein Paar für D in der Datenstruktur enthalten sein.

- $AddToDSEntry(D, \delta)$ fügt zu dem bisher in der Datenstruktur verwalteten Gewicht für D den Wert δ hinzu.

- $GetWeightFromDS(D)$ liefert das derzeit für D in der Datenstruktur abgelegte Gewicht.

Notwendige Operationen auf der invertierten Liste

Wir nehmen an, dass auf der invertierten Liste zu einem Term k folgende Operationen zur Verfügung stehen:

- $InitList(k)$ initialisiert die Liste für einen sequentiellen Durchlauf und setzt den Durchlaufzeiger auf das erste Listenelement.

- $GetNextElemOfList(k)$ liefert das aktuelle Paar (D, w_{dk}) aus der Liste im Rahmen des mit $InitList(k)$ gestarteten Durchlaufs und setzt den Durchlaufzeiger um ein Listenelement weiter.

- $NotAtEndOfList(k)$ liefert TRUE, sofern der Durchlaufzeiger auf einem Listenelement steht, und FALSE, wenn der Durchlaufzeiger hinter dem letzten Listenelement steht.

Realisierung des Array $TopDocs$

Realisierung des Arrays TopDocs

Das Array $TopDocs$ enthält $\gamma + 1$ Dokumente. Es werden zu jedem Zeitpunkt genau die Dokumente verwaltet, die bezogen auf die bisher betrachteten Listeneinträge die höchsten kumulierten Retrievalgewichte haben. Bei der Aktualisierung im Algorithmus wird dabei das betrachtete Dokument jeweils an der richtigen Stelle neu einsortiert.

```
for (alle Terme im Anfragevektor mit w_qk > 0) do
    Sei k' der bisher noch nicht betrachtete Term mit dem
        höchsten Wert w_qk
    InitList(k');
    while (NotAtEndOfList(k')) do /* durchlaufe die Liste */
        (D, w_dk') = GetNextElemOfList(k');
        if IsInDS(D) then
            AddToDSEntry(D, w_qk' · w_dk');
        else
            InsertIntoDS(D, w_qk' · w_dk');
        end;
        Sortiere D im Array TopDocs gemäß dem Gewicht
            GetWeightFromDS(D) ein;
    end;
    if (TopDocs[γ] > TopDocs[γ+1] + MaxRemainingWeight())
        then FINISH;
end;
```

Abbildung 5.7 — Der Algorithmus nach Buckley & Lewit

Die Operation $MaxRemainingWeight()$

Schließlich wird zur Formulierung des Algorithmus von Buckley und Lewit noch eine Funktion zur Berechnung des maximalen Gewichtszuwachses benötigt, der sich aus der Betrachtung der verbleibenden Anfrageterme ergeben könnte. Wenn wir davon ausgehen, dass die $w_{dk} \leq 1$ sind, erhalten wir Formel 5.12:

$$MaxRemainingWeight() = \sum_{\text{alle verbleibenden Anfrageterme } k'} w_{qk'} \quad (5.12)$$

Algorithmus nach Buckley & Lewit

Algorithmus nach Buckley & Lewit

Wir können damit den Algorithmus zur Suche nach den γ relevantesten Dokumenten mit Hilfe des Vektorraummodells nach Buckley & Lewit [BL85] wie in Abbildung 5.7 erfolgt formulieren:

Der beschriebene Algorithmus stellt sicher, dass die gefundenen γ Dokumente die γ Dokumente mit den höchsten Retrievalgewichten sind. Er stellt aber *nicht* sicher, dass diese γ Dokumente in der richtigen Reihenfolge im Ergebnis erscheinen! Dazu müsste man fordern, dass der

Unterschied zwischen allen benachbarten Einträgen in $TopDocs$ mindestens $MaxRemainingWeight()$ beträgt. Dies würde aber die Laufzeit negativ beeinflussen.

5.5.2 Steigerung der Performance

Der im letzten Abschnitt vorgestellte Algorithmus benutzt bereits ein Abbruchkriterium, das bewusst inexakte Ergebnisse in Kauf nimmt.

In der vorgestellten Form arbeitet der Algorithmus bis die γ Dokumente mit den höchsten Retrievalgewichten sicher erkannt sind. Die Rangordnung unter diesen γ Dokumenten ist möglicherweise noch nicht exakt. Eine solche bewusste Inkaufnahme nicht ganz korrekter Ergebnisse ist in IR-Systemen unter Umständen vertretbar, weil die berechneten Retrievalgewichte ohnehin nur eine vage Näherung der tatsächlichen Relevanzwahrscheinlichkeiten darstellen.

Man kann deshalb auf die Idee kommen, den Algorithmus noch früher zu beenden und durch Recall/Precision-Betrachtungen die dadurch entstehende Verschlechterung der Qualität des Ergebnisses messen. Hierzu brechen wir den Algorithmus gemäß einem Vorschlag von Buckley und Lewit [BL85] nun bereits ab, wenn n der gewünschten γ Dokumente sicher erkannt sind ($n \leq \gamma$).

In dem Algorithmus aus Abbildung 5.7 verwendet man dazu die folgende Abbruchbedingung:

```
if (TopDocs[n] > TopDocs[γ+1] + MaxRemainingWeight()) then
    FINISH;
```

Unter den γ Dokumenten, die dann in den ersten γ Zellen von $TopDocs$ stehen, gehören dadurch nur noch n sicher zu den γ rechnerisch relevantesten. Der Vorteil ist aber, dass man den Algorithmus früher abbrechen kann – das heißt, den erhaltenen Laufzeitgewinn erkaufen wir durch rechnerische Ungenauigkeit.

Experimentelle Ergebnisse für die INSPEC-Kollektion

INSPEC-Kollektion

Mit dem in seiner Laufzeit verbesserten Algorithmus von Buckley und Lewit wurden nun Messungen anhand der INSPEC-Kollektion durchgeführt (vgl. Tabelle 5.2). Dabei wurde der Recall für $\gamma = 10$ Dokumente bei jeweils verschiedenen Werten für n gemessen. Gleichzeitig

wurde die relativ zum ursprünglichen Algorithmus benötigte CPU-Zeit in Sekunden und die Anzahl der gelesenen Blöcke für die invertierten Listen aufgezeichnet. (Im Hinblick auf die CPU-Zeiten ist dabei zu beachten, dass die Messergebnisse von 1985 stammen!)

n	CPU-Zeit	# gelesene Blöcke	Recall
1	61.3	2833	0.1588
2	68.2	3058	0.1566
3	75.9	3255	0.1555
4	83.1	3451	0.1538
5	90.3	3630	0.1548
6	93.6	3721	0.1543
7	99.7	3863	0.1501
8	104.5	3980	0.1505
9	107.0	4033	0.1508
10	109.2	4111	0.1508

Tabelle 5.2 — Experimentelle Ergebnisse von Buckley und Lewit für die INSPEC-Kollektion

Als – auf den ersten Blick frappierendes – Ergebnis lässt sich feststellen, dass wir, je schlechter wir rechnen, also je kleiner n ist, desto bessere Ergebnisse im Recall erhalten.

Die Gründe für dieses überraschende Ergebnis können wie folgt beschrieben werden:

Der Algorithmus betrachtet zunächst die Anfrageterme mit einem hohen Anfragegewicht w_{qk}. Diese Terme sind in der Regel sehr trennscharf im Hinblick auf die Anfrage. Häufig führen dagegen bei längeren Anfragen die Anfrageterme mit einem geringeren Gewicht eher zu einer »Verwässerung« der Anfrage und senken so die Qualität des Ergebnisses.

Als Schlussfolgerung kann man sich in diesem Zusammenhang aber natürlich auch fragen, ob die Gewichtung der Anfrageterme nicht überdacht werden sollte.

Wichtig ist für uns an dieser Stelle aber primär die Erkenntnis, dass im Information Retrieval eine zu exakte Berechnung nicht unbedingt zu besseren Ergebnissen führen muss.

5.6 Zusammenfassung

In diesem Kapitel haben wir das Vektorraummodell betrachtet. Dieses Modell stellt eines der am häufigsten eingesetzten Modelle im Information Retrieval dar. Es repräsentiert sowohl Dokumente als auch Anfragen durch t-dimensionale Vektoren, wobei t die Anzahl der Terme im benutzten Indexierungsvokabular ist. Die Elemente dieser t-dimensionalen Vektoren stellen die Gewichtung der entsprechenden Terme in Bezug auf ein Dokument oder eine Anfrage dar.

Die Vorteile des Vektorraummodells lassen sich wie folgt zusammenfassen:

- Das Vektorraummodell stellt ein einfaches und verständliches Modell dar. Es erlaubt eine einfache Anfrageformulierung durch natürlichsprachliche Anfragen.
- Im Gegensatz zu Modellen des probabilistischen Information Retrieval, die wir im folgenden Kapitel betrachten werden, ist das Vektorraummodell unmittelbar auf neue Dokumentkollektionen anwendbar.
- Das Vektorraummodell hat sich seit seiner Veröffentlichung durch Salton et al. durch seine sehr gute empirische Retrievalqualität bewährt.
- Das Vektorraummodell kann mit Hilfe invertierter Listen effizient implementiert werden.
- Durch die Nutzung von Relevance Feedback lässt sich die Ergebnisqualität deutlich steigern.

Dem stehen folgende Nachteile gegenüber:

- Das Vektorraummodell enthält zahlreiche problematische Annahmen. So wird durch die Verwendung des Skalarproduktes (bzw. des Cosinusmaßes) implizit eine Unabhängigkeit zwischen den einzelnen Dimensionen der Dokument- und Anfragevektoren unterstellt. Zudem fehlt es an einer theoretischen Fundierung des Modells.
- Eine Anpassung an neue Anforderungen, wie bspw. strukturierte Dokumente mit Titel und Abstract, ist nur auf heuristischem Wege möglich.

Zusammenfassend bleibt aber festzuhalten, dass das Vektorraummodell heute eines der wichtigsten Modelle des Information Retrieval ist. Systeme die auf verschiedenen Variationen der *tf·idf*-Formel basieren sind heute vielfach im Einsatz.

Schließlich kann das Vektorraummodell pragmatisch recht einfach erweitert werden. Die Verwendung einer leistungsfähigen Stammformreduktion ist dabei ebenso möglich wie eine umfassende Betrachtung von Mehrwortgruppen. Um die Korrelationen zwischen den Termen zu berücksichtigen könnten die w_{dk} Werte auch analog zu unseren Überlegungen zum Fuzzy-Set Retrieval über eine $Term \times Term$-Matrix bestimmt werden (vgl. Abschnitt 4.5). Ferner wurde um den Problemen mit Stammformen und Mehrwortgruppen zu entgehen relativ erfolgreich der Einsatz von n-Grams im Vektorraummodell propagiert. Dabei werden statt der Terme 5-Grams als Dimensionen des Vektorraums genutzt [MM97].

6 Alternativen zur globalen Suche

Alternativen zur globalen Suche

In den bisherigen Kapiteln sind wir davon ausgegangen, dass der Benutzer seinen Informationswunsch in Form einer Anfrage formuliert und als Antwort auf diese Anfrage eine Menge von Dokumenten oder ein Ranking von Dokumenten erhält, das die nach Ansicht des Systems zur Anfrage relevanten Dokumente enthält. Im vorliegenden Kapitel wollen wir Alternativen und ergänzende Techniken zu diesem Vorgehen betrachten.

Alternativen zur globalen Suche

Zunächst ist in diesem Zusammenhang die Verwendung von *Klassifikationen* zu nennen. Dabei werden die Dokumente in ein vorgegebenes Ordnungssystem eingeordnet. Derartige Systeme stammen historisch aus dem Bibliotheksbereich, wo es u.a. darum geht, eine Systematik für die Aufstellung von Büchern in Regalen zu finden, die gewährleistet, dass thematisch ähnliche Bücher nah beieinander aufgestellt werden. Heute finden sich solche Ordnungssysteme z.B. auch bei Katalogen im Internet.

Klassifikationen

Einen zweiten Bereich bilden *Cluster-Ansätze*, die im Gegensatz zu Klassifikationen nicht auf einer manuell erstellten inhaltlichen Aufteilung der Themengebiete, sondern auf einer automatischen Bildung von Clustern inhaltlich ähnlicher Dokumente basieren. Dadurch kann der erforderliche manuelle Aufwand deutlich gesenkt werden. Andererseits ist eine geschlossene inhaltliche Umschreibung der Dokumente in einem Cluster durch einen oder mehrere Begriffe oft schwer zu geben.

Cluster-Ansätze

Schließlich stellt das *Browsing* eine weitere Herangehensweise an die Informationssuche dar. Die Möglichkeiten reichen vom einfachen Verfolgen von Links oder Verweisen in Dokumenten im Internet bis zu Systemen, die durch eine graphische Darstellung der inhaltlichen Bezüge von Dokumenten ein Navigieren im Dokumentraum ermöglichen. Hier kommen Aspekte der Visualisierung zum Tragen, um die inhaltlichen Zusammenhänge und Cluster möglichst überschaubar zu machen.

Browsing

Im Einzelnen gliedert sich das vorliegende Kapitel, das die drei oben angesprochenen Bereiche adressiert, wie folgt:

6.1 Klassifikationen
6.2 Cluster-Ansätze
6.3 Browsing
6.4 Zusammenfassung

6.1 Klassifikationen

Klassifikation

In Situationen, in denen man mit einer schwer zu überschauenden Menge von Objekten konfrontiert ist, aus denen z.B. eine Auswahl getroffen werden muss, erscheint es grundsätzlich sinnvoll, zunächst eine Klassifikation der Objekte vorzunehmen. Dies gilt bei der Auswahl aus mehreren Lösungsmöglichkeiten ebenso wie bei der Betrachtung von Forschungsansätzen oder eben auch bei Dokumenten. Eine Klassifikation versucht dabei die Menge der Objekte nach bestimmten Kriterien zu strukturieren und damit überschaubarer zu machen.

Folgende allgemeine – nicht auf das Information Retrieval bezogene – Definition zum Begriff der *Klassifizierung* findet sich in Gablers Wirtschaftsinformatik-Lexikon [SGR97]:

> **Klassifizierung**, *Klassifikation*, Abstraktionskonzept, das zu einer Einteilung von Dingen und Begriffen führt, die durch gemeinsame Merkmale miteinander verbunden sind. Sowohl aus wissenschaftlicher als auch aus praktischer Sicht dient die Klassifizierung zur Erhöhung der Überschaubarkeit eines beliebigen Arbeitsbereichs.
>
> Betrachten wir folgende Menge von Studenten *Franzose im Grundstudium*, *Deutscher im Grundstudium*, *Italiener im Hauptstudium*, *Österreicher im Hauptstudium*, so lassen sich diese Studenten nach dem Merkmal »Muttersprache« in die Klassen »deutschsprachige Studenten« und »fremdsprachige Studenten« und nach dem Merkmal »Studiumsabschnitt« in die Klassen »Studenten im Hauptstudium« und »Studenten im Grundstudium« einteilen.

> Dieses Beispiel zeigt, dass die Klassenbildung abhängig ist vom Anwendungs-/Problemzusammenhang und dass ein konkretes Realobjekt sehr wohl zu unterschiedlichen Klassen gehören kann.

Es geht also darum, die Elemente einer Grundgesamtheit nach ihren *Merkmalen* zu organisieren. Ein Problem stellt dabei unmittelbar die Tatsache dar, dass es in der Regel mehrere Merkmale gibt, aufgrund derer eine Einteilung möglich wäre. So könnte man Dokumente nach ihrer *Art* in »Monographien«, »Sammelbände«, »Zeitschriftenartikel«, »Konferenzbeiträge«, »technische Berichte«, ... einteilen. Daneben wäre auch eine Einteilung nach den *Autoren*, der *Seitenzahl*, dem *Erscheinungsjahr* oder dem *behandelten Thema* denkbar.

Im Hinblick auf das Information Retrieval erscheint eine Klassifikation nach dem behandelten Thema am sinnvollsten. Aufgrund der thematischen Vielfalt der Dokumente erscheint hierzu in der Regel ein mehrstufiges Vorgehen nützlich, bei dem Themenhierarchien gebildet werden, deren Ebenen die Themen oder Dokumente unterschiedlich detailliert unterscheiden. Streng hierarchische Systeme können als Themenbäume dargestellt werden. In der Wurzel werden alle Themen oder Dokumente zusammengefasst. Je weiter man sich von der Wurzel entfernt, umso spezifischer werden die Themen. Ein konkreter Knoten in der Hierarchie kann dabei durch den Pfad von der Wurzel bis zu diesem Knoten charakterisiert werden.

Klassifikationsschemata sind z.B. in Bibliotheken hilfreich, in denen sie die physische Anordnung der Bücher in den Regalen regeln. Sie sollen gewährleisten, dass thematisch verwandte Werke physisch nah beieinander stehen. Für die Suche z.B. nach dem Titel eines Buches oder nach einem Autorennamen müssen dann natürlich ergänzende Hilfsmittel wie z.B. nach Autorennamen oder Titelstichwörtern sortierte Kataloge zur Verfügung gestellt werden. Ein weiteres Beispiel für ergänzende Hilfsmittel bilden Semesterapparate, in denen die zu einzelnen Lehrveranstaltungen empfohlenen Bücher zusammengefasst werden.

Das wichtigste Kriterium bei der physischen Anordnung der Bücher in einer Bibliothek ist die Minimierung der Wege, die ein an einer bestimmten Thematik interessierter Leser zurückzulegen hat. Insofern ist eine an der Thematik der Bücher orientierte Vorgehensweise nahe liegend. Die Semesterapparate, die eine Ausnahme von der rein themenbezogenen Anordnung bilden, tragen dabei der Tatsache Rechnung, dass aufgrund

der aktuellen Lehrveranstaltungen eine gewisse ggf. auch themenübergreifende Anordnung der Bücher nach den Lehrveranstaltungen, in denen sie empfohlen werden, dem aktuellen Bedarf der Bibliotheksnutzer entspricht.

Bevor wir nun im Weiteren drei typische Klassifikationen betrachten, sei an dieser Stelle noch angemerkt, dass Klassifikationen zur »Ordnung durch Klassenbildung« auch in anderen Bereichen sehr verbreitet sind. Einige Beispiele mögen dies verdeutlichen (nach [Mei97]):

- Stoffgruppen in der Chemie
- Taxonomien in der Biologie
- Stoffgliederungen in Enzyklopädien

Für eine umfassendere Auseinandersetzung mit dem Thema *Klassifikationen* sei insbesondere [Buc89] empfohlen.

6.1.1 Die Internationale Patentklassifikation

Internationale Patentklassifikation

Ein erstes Beispiel für eine Klassifikation, die wir etwas ausführlicher betrachten wollen, stellt die Internationale Patentklassifikation (IPC) dar, mit deren Erarbeitung bereits 1954 begonnen wurde. Jedes eingereichte und akzeptierte Patent wird nach der IPC klassifiziert. Die Klassifikation von Patenten stellt eine sehr wichtige Aufgabe dar, da sichergestellt werden muss, dass zu neu eingereichten Patenten, die akzeptiert werden sollen, bisher keine ähnlichen oder identischen Patente existieren.

Die Internationale Patentklassifikation ist als hierarchisches System aufgebaut. Die einzelnen Ebenen dieses Systems bestehen aus:

- Sektionen
- Klassen
- Unterklassen
- Gruppen
- Untergruppen

Sektion	Inhalt
A	Täglicher Lebensbedarf
B	Arbeitsverfahren; Transportieren
C	Chemie; Hüttenwesen
D	Textilien; Papier
E	Bauwesen; Erdbohren; Bergbau
F	Maschinenbau; Beleuchtung; Heizung; Waffen; Sprengen
G	Physik
H	Elektrotechnik

Tabelle 6.1 — Die Sektionen der IPC

Tabelle 6.1 zeigt die Sektionen der IPC (vgl. hierzu auch die Web-Seiten des Deutschen Patent- und Markenamtes DPMA, http://www.depatisnet.de/ipc/index.html, letzter Zugriff 20.11.2006):

In einem konkreten Beispiel erfolgt die Klassifikation anhand eines Klassifikationsschlüssels, in dem die einzelnen Ebenen enthalten sind (Tabelle 6.2).

Schlüssel	Ebene	Bezeichnung
G	Sektion	Physik
G01	Klasse	Messen; Prüfen
G01R	Unterklasse	Messen elektrischer Größen; Messen magnetischer Größen
G01R 23	Gruppe	Anordnungen zum Messen von Frequenzen; Anordnungen zum Analysieren von Frequenzspektren
G01R 23/16	Untergruppe	Spektralanalyse; Fourier-Analyse

Tabelle 6.2 — Beispiel für einen Pfad in der Internationalen Patentklassifikation

In den Dokumenten zur IPC sind dabei ausführliche Anmerkungen abgedruckt, die eine Klasse definieren und ihren Inhalt begrenzen.

Revisionierung der IPC

Im Gebiet der Patente haben es Klassifikationen insofern schwer, als gerade Patente Neuerungen enthalten oder sich in Bereichen bewegen, die noch nicht in der Klassifikation berücksichtigt wurden.

Revisionierung

Aus diesem Grund wird die Internationale Patentklassifikation alle fünf Jahre einer Revision unterzogen. Die bisher klassifizierten Patente werden jedoch – auch auf Grund ihrer Menge – nicht nach der neuen Version der Klassifikation eingeordnet, sondern bleiben nach der alten Version zugeordnet. Dadurch müssen bei der Recherche die Veränderungen an der Klassifikation durch deren zwischenzeitliche Revisionen berücksichtigt werden. Dies erfolgt durch Nachschlagen in einer Konkordanz sowie in den älteren Ausgaben der IPC.

Die IPC wird durch ein neunbändiges Handbuch erschlossen. Neben einem Einführungsband ist für jede der acht Sektionen jeweils ein eigener Band vorgesehen. Zudem sind ein Schlagwortverzeichnis sowie eine CD-ROM Datenbank verfügbar.

Aufbau der Klassifikation von Patentschriften

Patente erhalten eine Hauptnotation, welche die Erfindung im Kern kennzeichnet sowie gegebenenfalls Nebennotationen, die weitere Aspekte bezeichnen.

Dementsprechend unterscheiden Patentdatenbanken zwischen der Haupt-IPC bzw. Main IPC im Feld ICM für die Hauptnotation und der Neben-IPC bzw. Secondary IPC im Feld ICS.

Probleme von Klassifikationen

Die Betrachtung der Internationalen Patentklassifikation macht einige der Probleme beim Einsatz von Klassifikationen deutlich.

Ein Problembereich ergibt sich dadurch, dass Klassifikationen innovative Themen und Gebiete häufig (noch) nicht berücksichtigen. Insbesondere wissenschaftliche Arbeiten oder eben auch Patente beinhalten aber von ihrer Natur aus Themen, die neuartig und damit in einer anzuwendenden Klassifikation oft noch nicht berücksichtigt sind.

Der Aufbau und die Pflege einer Klassifikation stellen damit einen hohen Aufwand dar, der dazu führt, dass Klassifikationen – wenn überhaupt – nur in bestimmten, meist einige Jahre auseinander liegenden Revisionen überarbeitet werden können.

Zudem behandeln bzw. berücksichtigen bspw. wissenschaftliche Arbeiten gerne interdisziplinäre Themen. Ein Beispiel hierfür wäre das Thema einer Arbeit: »Datenbanken und Information Retrieval – zwei getrennte

Welten?«. Eine eindeutige Zuordnung dieses Themas zu einer Klassifikationsgruppe ist nur schwer möglich. Die IPC berücksichtigt dies durch die Möglichkeit der Vergabe einer Neben-IPC.

6.1.2 Die Dezimalklassifikation

Ein weiteres Beispiel für eine Klassifikation ist die *Universal Decimal Classification* (UDC).

Dezimalklassifikation

Um die Systematik der Klassifikation im Bibliothekswesen zu vereinheitlichen, hat der amerikanische Bibliothekar Malvil Dewey (1851-1931) 1876 diese Dezimalklassifikation geschaffen, die heute in über 100.000 Bibliotheken verwendet wird.

Die Dezimalklassifikation kennzeichnet jede Grundwissenschaft mit einer einstelligen Zahl von 0 bis 9. So stehen bspw.:

- 0 für Allgemeines,
- 1 für Philosophie und Psychologie,
- 2 für Religion usw.

Diese Zahl kann nun durch Hinzufügen einer weiteren Zahl zwischen 0 und 9 in Untergruppen gegliedert werden. Die Untergruppen können ihrerseits wieder in gleicher Weise unterteilt werden.

Die angegebene Abfolge zeigt beispielhaft die Einordnung des Gebietes *Kohlebergbau* in die Klassifikationsgruppe 622.33:

- 6 Technologie
- 62 Technik
- 622 Bergbautechnik
- 622.3 Einzelne Bergbauzweige
- 622.33 Kohlebergbau

Nach jeweils drei Stellen fügt man zur besseren Lesbarkeit meist einen Punkt ein.

Der Aufbau der Hauptklassen (Main classes) der Dezimalklassifikation gliedert sich wie folgt in den so genannten Haupttafeln:

Klasse	Inhalt
000	Generalities
100	Philosophy and related disciplines
200	Religion
300	The social sciences
400	Language
500	Pure sciences
600	Technology (applied sciences)
700	The arts
800	Literature
900	General geography and history

Die Hauptklasse 6 *Technology* untergliedert sich nun z.B. weiter in:

Klasse	Inhalt
600	Technology (applied sciences)
610	Medical sciences
620	Engineering and allied operations
630	Agriculture and related technologies
640	Home economics and family living
650	Management and auxiliary services
660	Chemical and related technologies
670	Manufactures
680	Manufacture or specific uses
690	Buildings

Die Unterklasse 62 *Engineering* unterteilt sich wiederum in:

Klasse	Inhalt
620	Engineering and allied operations
621	Applied physics
622	Mining and related operations
623	Military and nautical engineering
624	Civil engineering
625	Railroads, roads, highways
626	[not used]
627	Hydraulic engineering
628	Sanitary and municipal engineering
629	Other branches of engineering

Anzumerken ist, dass unter den allgemeinen Gruppen, wie bspw. 600 *Technology* (*applied sciences*) wirklich nur allgemeine Dokumente klassifiziert werden und nicht sonstige Dokumente, für die bspw. keine eindeutige Klassifikationsgruppe gefunden wurde.

So genannte [not used]-Einträge rühren daher, dass im Laufe der Zeit – immerhin ist die Dezimalklassifikation über 120 Jahre alt – zahlreiche Begriffe aus dem allgemeinen Sprachgebrauch verschwunden sind.

Facettierung

Neben den oben dargestellten Haupttafeln zur systematischen Einteilung der Klassen gibt es in der UDC so genannte *Hilfstafeln*. Diese Hilfstafeln definieren weitere Eigenschaften und dienen der *Facettierung*, d.h. bspw. der Einteilung der Dokumente nach Ort, Zeit, Form, Sprache oder auch zur feineren Stoffgliederung. Diese Facettierung wird in Klammern an die Hauptklassifikationsangabe angeschlossen. Wichtig ist dabei, dass sich die Merkmale wie z.B. der *Ort* nicht auf den Entstehungsort des Dokuments, sondern auf die Thematik beziehen. Ein Beispiel wäre ein Dokument, das sich mit »Kohlebergbau in England« beschäftigt.

Facettierung

Wir wollen die vielfältigen Möglichkeiten, die sich durch die Facettierung ergeben, anhand einiger Beispiele andeuten. Eine ausführliche Darstellung würde den Rahmen dieses Kurses aber sprengen. Der interessierte Leser sei z.B. auf [Fil81] verwiesen.

Beispiel

So steht nun z.B. (430) für »Deutschland« und dementsprechend steht 622(430) für »Bergbautechnik in Deutschland«.

Darüber hinaus ist auch die Darstellung von Beziehungen zwischen zwei Themen möglich. So klassifiziert 621.3:622 die »Elektrotechnik im Bergbau«.

Weitere Beispiele für die Möglichkeiten der Dezimalklassifikation sind:

- 53(038) klassifiziert »Dictionary of physics«

  Die 53 steht hier für Physik. (038) steht für »Wörterbuch« (Dictionary). Man beachte, dass die Notation (0..) die Form des Dokuments festlegt.

- 622+629 klassifiziert »Mining and metallurgy«

  Das »+« wird verwendet, um die Verbindung zwischen zwei Themengebieten zu kennzeichnen.

- [23/28:294.3](540) klassifiziert »Christianity in relation of Buddism in India«

  Hier bezeichnet 23/28 die christlichen Religionen. Der Schrägstrich »/« wird dabei verwendet, um ein Intervall von Themenbereichen anzugeben. Der Doppelpunkt »:« bezeichnet die symmetrische Beziehung zwischen zwei Sachverhalten. 294.3 steht für den Buddhismus. »[...]« steht für eine Gruppierung von Themen. (540) steht schließlich für Indien. Man beachte zudem, dass Ortsangaben immer in Klammern stehen.

Zur Facettierung dienen in der Dezimalklassifikation somit Anhängezahlen, die durch spezielle Zeichen eingeleitet bzw. umschlossen werden. Es gibt einerseits allgemeine Anhängezahlen, die überall in der Dezimalklassifikation verwendet werden dürfen. Andererseits gibt es auch spezielle Anhängezahlen, die nur für bestimmte Klassen innerhalb der Dezimalklassifikation zulässig sind.

Für die allgemeinen Anhängezahlen sind Konstrukte mit den folgenden, vorangestellten bzw. umschließenden Zeichen zulässig:

- = Sprache
- (0...) Form
- (...) Ort
- (=...) Rassen und Völker
- »...« Zeit
- .00 Gesichtspunkt
- -05 Person

Verknüpfung von Dezimalklassifikationen

Verknüpfung einzelner Klassen

Zur Verknüpfung von einzelnen Klassen aus der Dezimalklassifikation existieren als syntaktische Elemente folgende Sonderzeichen:

- + Aufzählung mehrerer Sachverhalte,
- : symmetrische Beziehung zwischen zwei Sachverhalten,
- :: asymmetrische Beziehung zwischen zwei Sachverhalten,
- / Erstreckungszeichen zur Zusammenfassung mehrerer nebeneinander stehender Dezimalklassifikations-Zahlen,
- ' Zusammenfassungszeichen zur Bildung neuer Sachverhalte aus der Kombination einzelner Dezimalklassifikations-Komponenten.

Die UDC ist damit durch die Ergänzung der Hauptklassen um Facettierungselemente und Verknüpfungsmöglichkeiten sehr mächtig. Sie ist insbesondere für geschulte Bibliothekare ein sehr hilfreiches Werkzeug.

6.1.3 Kataloge im Internet

Zur Suche im Internet existieren grundsätzlich zwei verschiedene Ansätze:

1. Zum einen existieren Suchmaschinen wie *Google*[1] oder *Altavista*[2], die eine Anfrage des Benutzers – z.B. in Form einer Wortmenge oder eines Booleschen Ausdrucks – entgegennehmen und zu dieser Anfrage ein individuelles Ergebnis berechnen. Diese Suchmaschinen basieren auf den in diesem Kurs beschriebenen IR-Modellen und sollen im Kapitel 9 näher betrachtet werden.

2. Neben diesen Suchmaschinen existieren aber auch Kataloge wie *Yahoo!*[3], *Web.de*[4] oder *Dino Online*[5], die die Inhalte des Webs im Sinne einer Klassifikation organisieren und so den Zugang zu interessanten Dokumenten unterstützen.

Kataloge im Internet

Dabei ist anzumerken, dass heute fast alle Suchmaschinen für das Internet parallel beide Zugriffsmöglichkeiten unterstützen.

[1] http://www.google.de, letzter Zugriff 20.11.2006
[2] http://de.altavista.com, letzter Zugriff 20.11.2006
[3] http://de.yahoo.com, letzter Zugriff 20.11.2006
[4] http://www.web.de, letzter Zugriff 20.11.2006
[5] http://www.dino-online.de, letzter Zugriff 20.11.2006

Wir wollen im Folgenden nun exemplarisch anhand von Yahoo! zeigen, wie ein solcher Internet-Katalog die Ideen von Klassifikationen umsetzt. Abbildung 6.1 zeigt die obersten beiden Ebenen der Klassifikation von Yahoo!. Die Klassifikation unterscheidet sich dabei inhaltlich natürlich deutlich von der Dezimalklassifikation, da Yahoo! wie alle anderen Katalogbetreiber im Internet versucht, die Themenschwerpunkte im WWW zu repräsentieren.

Abbildung 6.1 — Die ersten beiden Stufen der von Yahoo! angewendeten Klassifikation für Internetseiten

Abbildung 6.2 zeigt den Inhalt der Seite die man erreicht, wenn man auf der Hauptseite in der Kategorie *Reisen und Freizeit* die Unterkategorie

Autos wählt. Hier finden sich – neben Verweisen auf die entsprechenden Seiten für Österreich und die Schweiz – einerseits die Unterkategorien auf der nächsttieferen Ebene und andererseits Verweise auf recht allgemeine Webseiten, die keiner spezielleren Unterkategorie zugeordnet werden konnten.

Bei einigen Verweisen auf Unterkategorien in Abbildung 6.2 fällt auf, dass diese mit einem »@« enden. Diese Verweise führen zu Unterkategorien, die primär in einem anderen Pfad der hierarchischen Klassifikation eingeordnet sind und hier quasi zusätzlich angeboten werden. Abbildung 6.3 verdeutlich dies durch die Seite, die über den Verweis *Fernsehsendungen@* erreicht wird.

Kataloge im Internet stellen damit ganz ähnliche Klassifikationssysteme zur Verfügung, wie sie im Bibliotheksbereich bereits seit langem im Einsatz sind. Einen gewissen Vorteil haben Internet-Kataloge durch die Tatsache, dass sie keine räumliche Anordnung der Dokumente zum Ziel haben. So bereitet hier das Einordnen eines Dokuments in mehrere Pfade ebenso wenig ein Problem wie Querverweise in der Hierarchie.

Auch bei Internet-Katalogen bedeutet aber die Pflege der Klassifikation und das Einordnen neuer Dokumente in das Klassifikationsschema einen hohen manuellen Aufwand. Eine gewisse Unterstützung können hierbei die im Abschnitt 6.2 beschriebenen Ansätze zum automatischen Clustern bieten.

Einordnung hierarchischer Klassifikationen

Einordnung hierarchischer Klassifikationen

Hierarchische Klassifikationen zeichnen sich dadurch aus, dass sie eine stufenweise Spezifizierung der Klassen vornehmen. Grundsätzlich muss jedes Element einer untergeordneten Klasse alle Merkmale der übergeordneten Klasse enthalten. Zusätzlich muss die Unterklasse weitere Merkmale enthalten.

Die enumerative, also aufzählende Klassifikation wurde und wird auch heute als ordnungsschaffende Methode verwendet.

Aufgrund der manuellen Erarbeitung veralten Klassifikationen bereits im Augenblick ihrer Entstehung. Die rasche Entwicklung in Wissenschaft, Industrie und Wirtschaft macht daher ständige Überarbeitungen nötig. Durch diese ständigen Änderungen sind Inhomogenitäten oder Brüche unvermeidlich.

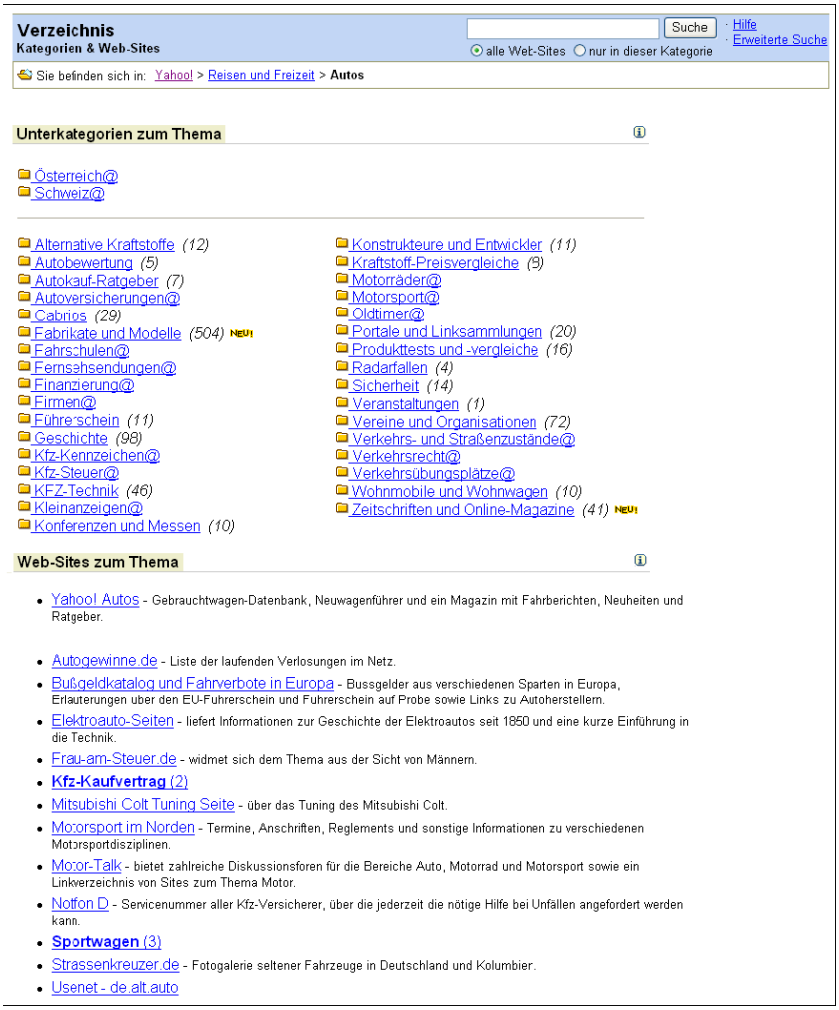


Abbildung 6.2 — Unterkategorien und Webseiten zum Thema *Autos* in Yahoo!

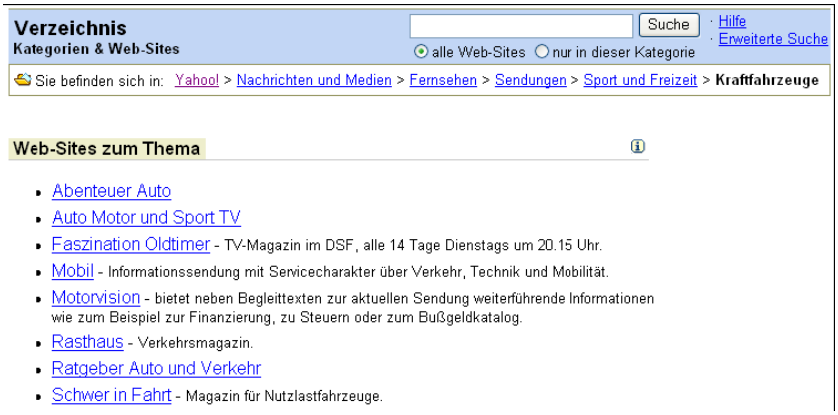


Abbildung 6.3 — Ein Beispiel für einen Querverweis: diese Seite erreicht man sowohl über den Pfad *Yahoo!* → *Reisen und Freizeit* → *Autos* → *Fernsehsendungen@* als auch über den Pfad *Yahoo!* → *Nachrichten und Medien* → *Fernsehen* → *Sendungen* → *Sport und Freizeit* → *Kraftfahrzeuge*

Hinzu kommt die nicht zu verhindernde Inkonsistenz in der Notationsvergabe und der fehlerträchtige Umgang mit den verwendeten Zahlenketten.

Einsatzgebiete von Klassifikationssystemen

Einsatzgebiete von Klassifikationen

Klassifikationen selbst müssen in unterschiedlichen Zusammenhängen und verschiedener Bedeutung gesehen werden.

So können Klassifikationen zur Wissensdarstellung und Wissensvermittlung – also wissens- und erkenntnisorientiert – eingesetzt werden. Das heißt es wird zunächst klassifiziert, was es alles gibt oder geben könnte. Beispiele hierfür wären eine Klassifikation von Sortieralgorithmen oder von Kraftfahrzeugen.

Klassifikationen können aber auch als Organisationsmittel zum geordneten Ablegen und Aufbewahren eingesetzt werden. Ein Beispiel hierfür sind so genannte Aktenpläne von Behörden und Verwaltungen, die eine entsprechende Ablage und Archivierung von Dokumenten und Akten ermöglichen.

Schließlich können Klassifikationen auch als inhaltsbeschreibendes Element – das heißt pragmatisch-dokumentarisch orientiert – zum Einsatz

kommen.

Klassifikationssysteme sind also mehr oder weniger komplexe Systeme von Themenklassen mit Ordnungscharakter. Eine Themenklasse ist eine Zusammenfassung von gleichen Sachverhalten unter bestimmten Voraussetzungen.

Mono- und polyhierarchische Klassifikationssysteme

monohierarchische Klassifkationssysteme

polyhierarchische Klassifkationssysteme

Unter einem monohierarchischen Klassifikationssystem versteht man, dass eine Klasse nur einer anderen Klasse untergeordnet werden kann, während in polyhierarchischen Systemen eine Klasse mehreren anderen Klassen untergeordnet werden kann (wie z.B. bei Yahoo!). Graphisch lässt sich somit ein monohierarchisches Klassifikationssystem als Baum und ein polyhierarchisches Klassifikationssystem als Graph darstellen.

Monohierarchische Klassifikationssysteme sind zwar in vielen Fällen inadäquat, dennoch sollte man das Ziel nicht aus den Augen verlieren, dass eine Klassifikation möglichst eindeutig sein sollte. Das heißt die Möglichkeit eines polyhierarchischen Klassifikationssystems, einzelne Dokumente mehreren Klassen zuzuordnen, sollte nicht dazu führen, dass eine Klassifikation dadurch aufgeweicht und unscharf wird.

6.2 Cluster-Ansätze

Cluster-Analyse

Die Cluster-Analyse stammt aus dem Bereich der Statistik. Es geht vereinfacht ausgedrückt darum, eine Menge so in Teilmengen (*Cluster*) einzuteilen, dass

- die Elemente innerhalb jeder Teilmenge ein hohes Maß an Übereinstimmung aufweisen, während
- die Elemente in unterschiedlichen Teilmengen ein geringes Maß an Übereinstimmung aufweisen.

Man bezeichnet Ansätze der Cluster-Analyse gelegentlich auch als *automatische Klassifikation*. Dies ist aber nicht ganz korrekt, weil einerseits das Klassifikationsschema nicht a priori bekannt ist, sondern implizit durch die Daten selbst bestimmt wird und andererseits eine saubere inhaltliche Beschreibung der empirisch ermittelten Cluster kaum möglich ist.

Der Wert der Cluster-Analyse liegt allgemein darin, dass sie in der Lage ist, große Mengen zu strukturieren und zuvor unentdeckte Beziehungen zwischen den Elementen aufzuzeigen.

Im Information Retrieval sind insbesondere die folgenden drei Anwendungen der Cluster-Analyse gebräuchlich (wir orientieren uns dabei, wie im weiteren Verlauf dieses Kapitels, an [Ras92]):

- Zunächst können Dokumente aufgrund der in ihnen vorkommenden Terme – oder allgemeiner aufgrund ihrer Repräsentationen – zu Clustern gruppiert werden. Dieser Ansatz kann einerseits auf eine gesamte Dokumentenkollektion angewendet werden, daneben ist aber auch der Einsatz bei einer Ergebnismenge, die z.B. von einer Booleschen Anfrage geliefert wird, möglich, um das Ergebnis zu strukturieren. Schließlich kann ein solcher Ansatz auch bei verteilten Information Retrieval Systemen nützlich sein, bei denen die Dokumente thematisch auf die einzelnen Knoten eines verteilten Systems aufgeteilt werden müssen.

- Die Zuordnung der Dokumente zu den Clustern muss sich natürlich nicht zwingend an den in den Dokumenten vorkommenden Termen orientieren. Alternativ könnte man bei wissenschaftlichen Publikationen z.B. eine Clusterbildung aufgrund der in den Dokumenten vorkommenden Zitate oder anderer Kriterien vornehmen.

- Schließlich können Cluster-Ansätze nicht nur zum Clustern von Dokumenten, sondern z.B. auch zum Clustern von Termen verwendet werden. Dieser Ansatz wird u.a. eingesetzt, um automatisch Terme zu ermitteln, die in ähnlicher Weise verwendet werden und somit Synonyme sein könnten. Damit kann dann automatisch ein Thesaurusersatz gewonnen werden.

Wir wollen nun im Weiteren zunächst mögliche Vorgehensweisen zur Bildung von Clustern betrachten und anschließend auf typische Anwendungen dieser Ansätze eingehen.

6.2.1 Ähnlichkeitsmaße und die Ähnlichkeitsmatrix

Die Ausgangsbasis um eine inhaltliche Clusterung der Dokumente vornehmen zu können, bildet wie schon beim Vektorraummodell ein Maß, *Ähnlichkeitsmaße*

das die Ähnlichkeit zwischen zwei Dokumenten bzw. ihren Abstand bestimmt. Dieses Maß sollte nach Möglichkeit normiert sein, um den Einfluss der Dokumentlänge zu eliminieren.

Euklidischer Abstand

Neben dem Euklidischen Abstand werden hierfür im Zusammenhang mit Cluster-Verfahren häufig der Dice-Koeffizient, der Jaccard-Koeffizient oder das bereits bekannte Cosinus-Maß verwendet. Wenn wir für die Dokumente erneut von t-dimensionalen Beschreibungsvektoren mit den Komponenten w_{ik} ausgehen, dann werden diese Maße durch die folgenden Formeln beschrieben:

Dice-Koeffizent

- Dice-Koeffizient :

$$sim_{D_i,D_j} = \frac{2 \cdot \sum_{k=1}^{t} (w_{ik} \cdot w_{jk})}{\sum_{k=1}^{t} w_{ik}^2 + \sum_{k=1}^{t} w_{jk}^2} \tag{6.1}$$

Jaccard-Koeffizent

- Jaccard-Koeffizient :

$$sim_{D_i,D_j} = \frac{\sum_{k=1}^{t} (w_{ik} \cdot w_{jk})}{\sum_{k=1}^{t} w_{ik}^2 + \sum_{k=1}^{t} w_{jk}^2 - \sum_{k=1}^{t} (w_{ik} \cdot w_{jk})} \tag{6.2}$$

Cosinus-Maß

- Cosinus-Maß :

$$sim_{D_i,D_j} = \frac{\sum_{k=1}^{t} (w_{ik} \cdot w_{jk})}{\sqrt{\sum_{k=1}^{t} w_{ik}^2 \cdot \sum_{k=1}^{t} w_{jk}^2}} \tag{6.3}$$

Ähnlichkeitsmatrix

Unter Verwendung eines dieser Maße kann dann eine Ähnlichkeitsmatrix bestimmt werden, die die paarweisen Ähnlichkeiten zwischen den Dokumenten angibt. Für die obigen symmetrischen Maße reicht es dabei

aus, die untere Dreiecksmatrix zu bestimmen, da die obere Dreiecksmatrix symmetrisch ist. Die Werte auf der Diagonalen werden dabei ausgeblendet, weil die Ähnlichkeit eines Dokuments zu sich selbst für die Cluster-Analyse ohne Belang ist. Wir erhalten damit:

$$SIM = \begin{pmatrix} sim_{D_2,D_1} & & & & \\ sim_{D_3,D_1} & sim_{D_3,D_2} & & & \\ sim_{D_4,D_1} & sim_{D_4,D_2} & sim_{D_4,D_3} & & \\ \vdots & \vdots & \vdots & \ddots & \\ sim_{D_N,D_1} & sim_{D_N,D_2} & sim_{D_N,D_3} & \ldots & sim_{D_N,D_N-1} \end{pmatrix}$$

Diese Matrix, in der die paarweisen Ähnlichkeiten zwischen den Dokumenten abgelegt sind, kann z.B. auf der Basis von invertierten Listen berechnet werden. Sie kann dann während der eigentlichen Clusterbestimmung genutzt werden, um Mehrfachberechnungen der Ähnlichkeiten zu vermeiden.

Man kann die Vorgehensweisen bei der Bildung von Clustern nun in hierarchische und nichthierarchische Verfahren unterteilen. Die hierarchischen Verfahren differenzieren sich weiter in agglomerative und teilende Verfahren:

hierarchische Verfahren der Clusterbildung

- Bei den agglomerativen Verfahren geht man bei einer Gesamtheit von N Dokumenten zunächst von N einelementigen Clustern aus, die sukzessive zu immer umfangreicheren Clustern verschmolzen werden.

  agglomerative Verfahren der Clusterbildung

- Bei den teilenden Verfahren geht man dagegen von einem einzigen N-elementigen Cluster aus, der sukzessive in kleinere Cluster zerlegt wird.

  teilende Verfahren der Clusterbildung

Die nichthierarchischen Verfahren arbeiten nicht in dieser Weise zusammenführend oder zerlegend, sondern verwenden andere Kriterien zur Clusterbildung.

nichthierarchische Verfahren der Clusterbildung

In den folgenden Abschnitten wollen wir uns nun zunächst mit nichthierarchischen und anschließend mit hierarchischen Verfahren beschäftigen.

6.2.2 Nichthierarchische Ansätze zur Clusterbildung

Erzeugung und Aktualisierung von Clustern

Bei den nichthierarchischen Verfahren zur Clusterbildung unterscheidet man Verfahren zur Erzeugung von Clustern und Verfahren zur Aktualisierung von Clustern.

Clique-Technik

Ein erstes Beispiel für ein erzeugendes, nichthierarchisches Verfahren bildet die *Clique-Technik*. Jedes Element eines Clusters muss dabei zu *allen* anderen Elementen des Clusters eine Ähnlichkeit oberhalb eines festgelegten Schwellenwertes haben. Dabei ist man natürlich an Clustern maximaler Größe interessiert. Dies führt dazu, dass ein Element mehreren Clustern angehören kann. Andererseits kann es aber auch Cluster geben, die nur aus einem Element bestehen.

Die Berechnung der Cluster nach der Clique-Technik ist allerdings recht aufwändig. Man muss im Prinzip alle möglichen Cluster ausprobieren und kann dabei lediglich einige Kombinationen mit fehlender Ähnlichkeit ausschließen, um den Aufwand zu begrenzen.

Star-Technik

Ein wesentlich einfacher zu berechnendes Verfahren stellt die *Star-Technik* dar. Hier bildet jedes Element ein Cluster mit den Elementen, die ihm – gemäß einem vorgegebenen Schwellenwert für die Ähnlichkeit – ähnlich sind. Ein Element kann dabei natürlich wieder mehreren Clustern angehören.

Zur Bestimmung der Cluster müssen hier lediglich zu jedem Dokument alle übrigen Dokumente betrachtet werden. Der Aufwand wächst damit mit wachsendem N proportional zu N^2.

Reallocation-Technik

Den letzten nichthierarchischen Ansatz, den wir hier betrachten wollen, bildet die *Reallocation-Technik*. Dieser Ansatz ist im Grunde den Verfahren zur Aktualisierung von Clustern zuzuordnen. Er kann aber auch zur initialen Erzeugung von Clustern genutzt werden. Bei der aktualisierenden Variante gehen wir davon aus, dass bereits Cluster gebildet wurden. Die Aufgabe besteht nun darin, ein neues Dokument dem am besten passenden Cluster zuzuordnen. Dazu geht die Reallocation-Technik wie folgt vor:

- Zu jedem Cluster wird der Zentroid (das fiktive »Durchschnittsdokument«) verwaltet.
- Für das neue Dokument wird nun die Ähnlichkeit zu allen Zentroiden berechnet.

- Das Dokument wird dem Cluster zugeordnet, dessen Zentroid es am ähnlichsten ist.
- Der Zentroid dieses Clusters wird neu berechnet.
- Nun können alle Dokumente in allen Clustern erneut betrachtet und den Clustern zugeordnet werden, für die sie dem Zentroid am ähnlichsten sind.
- Nun müssen die Zentroide aller veränderten Cluster aktualisiert und das Verfahren solange wiederholt werden, bis sich eine Stabilisierung ergibt.

Durch die nachfolgende Betrachtung aller Dokumente trägt das Verfahren der Tatsache Rechnung, dass das Einfügen eines neuen Elements in einen Cluster dessen Zentroid so verändert haben könnte, dass sich für andere Dokumente nun eine neue Zuordnung anbietet. Das daraus resultierende iterative Verfahren kann durchaus aufwändig sein.

Das Verfahren kann nun z.B. in der Form zum Erzeugen von Clustern eingesetzt werden, dass der im Verfahren beschriebene Iterationsprozess auf Basis der Ähnlichkeiten zu den Zentroiden z.B. im Anschluss an eine initiale Anwendung der Star-Technik eingesetzt wird.

Festlegung des Schwellenwertes

Ein praktisches Problem bei der Clique- und der Star-Technik bildet allerdings die Festlegung des Schwellenwertes. Ist er zu hoch, erhält man fast nur einelementige Cluster. Ist er zu niedrig, erhält man sehr viele sehr umfangreiche Cluster. Die im Folgenden betrachteten hierarchischen Verfahren weisen diesen Nachteil nicht auf.

6.2.3 Hierarchische Ansätze zur Clusterbildung

agglomerative Verfahren der Clusterbildung

teilende Verfahren der Clusterbildung

Die hierarchischen Verfahren unterteilen sich wie bereits erwähnt in agglomerative und teilende Verfahren: Bei den agglomerativen Verfahren geht man bei einer Gesamtheit von N Dokumenten zunächst von N einelementigen Clustern aus, die sukzessive zu immer umfangreicheren Clustern verschmolzen werden. Bei den teilenden Verfahren geht man dagegen von einem einzigen N-elementigen Cluster aus, der sukzessive in kleinere Cluster zerlegt wird.

Eine wesentliche Stärke der hierarchischen Verfahren – z.B. im Vergleich zur Clique-Technik – ist der relativ geringe Rechenaufwand. Hierzu trägt

insbesondere bei, dass mit jeder Bildung bzw. Aufteilung eines Clusters die Anzahl der zu berücksichtigenden Möglichkeiten reduziert wird. Der Preis, der für diesen Vorteil gezahlt werden muss, ist allerdings, dass die in einer frühen Phase des Verfahrens getroffenen Entscheidungen nicht mehr revidiert werden können.

Eine wesentliche Frage bei den agglomerativen Verfahren ist nun, welche beiden Cluster als nächstes verschmolzen werden sollen. Hierfür gibt es im Wesentlichen fünf Ansätze:

- **Single Link**

Single Link

Der *Single Link* Ansatz bildet den ersten Ansatz der sogenannten *Linkage Methoden*. Dabei werden jeweils die beiden Cluster verschmolzen, die das Paar mit größter Ähnlichkeit enthalten. Die Ähnlichkeit zwischen zwei Clustern wird somit über die maximale Ähnlichkeit zwischen zwei Elementen der Cluster definiert.

Formal ausgedrückt ergibt sich die Ähnlichkeit zweier Cluster A und B als $MAX_{x \in A; y \in B} sim(x, y)$.

Ein Nachteil dieses Verfahrens ist, dass große Cluster hier die Tendenz haben sich weiter zu vergrößern, während kleine Cluster klein bleiben. Die Ursache ist, dass ja nur ein Element aus dem großen Cluster zu dem anderen Cluster passen muss. Daher auch der Name des Verfahrens, weil ein einziger »Link« zwischen zwei Clustern genügt, um ihre Verschmelzung herbeizuführen.

- **Complete Link**

Complete Link

Das andere Extrem zum *Single Link* Ansatz bildet der *Complete Link* Ansatz. Die Ähnlichkeit zwischen zwei Clustern wird hier über die minimale Ähnlichkeit zwischen zwei Elementen der Cluster definiert. Formal ergibt sich die Ähnlichkeit zweier Cluster A und B hier als $MIN_{x \in A; y \in B} sim(x, y)$.

Hier wird also darauf geachtet, dass alle Elemente der Cluster, die verschmolzen werden sollen, paarweise gut zueinander passen. Im Gegensatz zum *Single Link* Ansatz werden dadurch kleinere Cluster bei der Verschmelzung bevorzugt und es entstehen von der Größe her sehr homogene Cluster.

- **Average Link**

Average Link

Die dritte nahe liegende Möglichkeit ist nun, statt des Maximums oder des Minimums über die Ähnlichkeiten zwischen den Elementen aus den Clustern deren Durchschnitt zu betrachten. For-

mal ergibt sich die Ähnlichkeit zweier Cluster A und B dann zu $\frac{1}{|A \times B|} \cdot \sum_{x \in A; y \in B} sim(x, y)$, wobei $|A \times B|$ für die Anzahl der Elemente im kartesischen Produkt der Clustermengen steht.

Von den Charakteristika der entstehenden Cluster her nimmt dieser Ansatz eine Zwischenstellung zwischen *Single Link* und *Complete Link* ein.

- **Zentroid Methode**

  Zentroid Methode

  Im Gegensatz zum *Average Link* Ansatz wird hier nicht der Durchschnitt der Ähnlichkeiten gebildet. Stattdessen wird zu jedem Cluster der Zentroid (also der »Clusterschwerpunkt«) berechnet. Verschmolzen werden dann die Cluster mit dem ähnlichsten Zentroid. Je nach eingesetztem Ähnlichkeitsmaß kann dieses Verfahren auch zu anderen Ergebnissen als der *Average Link* Ansatz führen.

- **Ward's Methode**

  Ward's Methode

  Von Ward stammt der Vorschlag die zu verschmelzenden Cluster so zu wählen, dass die Fehlersumme der Quadrate der Euklidischen Distanz zwischen den Elementen minimiert wird. Dieser Ansatz führt zu homogenen Clustern und einer recht symmetrischen Zerlegungshierarchie. Er ist aber etwas aufwändiger in der Berechnung als die anderen Ansätze.

  Anzumerken ist hier, dass Ward's Methode auch teilend (divisiv) angewendet werden kann. Man beginnt dazu mit der Gesamtheit aller Dokumente als Ausgangscluster und teilt diesen so in zwei Teilcluster, dass die Fehlersumme der Quadrate der Euklidischen Distanz zwischen den Elementen minimiert wird. Dies kann dann rekursiv fortgesetzt werden.

Abbildung 6.4 verdeutlicht zusammenfassend das Vorgehen bei agglomerativen hierarchischen Cluster-Verfahren. Man bezeichnet diese Darstellungsform auch als *Dendogramm*. Auf die denkbaren Einsatzszenarien werden wir in den folgenden Abschnitten noch näher eingehen.

Dendogramm

Allgemeiner Algorithmus für agglomerative Verfahren

Die im vergangenen Abschnitt beschriebenen agglomerativen Verfahren (Single Link, Complete Link und Average Link) können sehr einfach auf Basis der in Abschnitt 6.2.1 vorgestellten Ähnlichkeitsmatrix implementiert werden. Wir wollen hier den Ablauf des Algorithmus skizzieren:

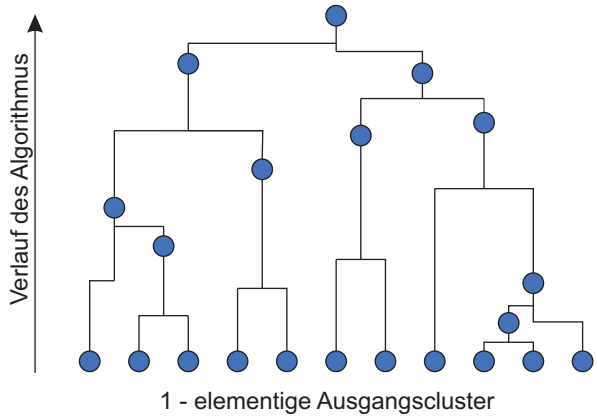


Abbildung 6.4 — Ablauf beim agglomerativen hierarchischen Clustering

1. Der Algorithmus startet mit N Clustern, die jeweils aus genau einem Element bestehen. Die Cluster seien mit 1 bis N nummeriert.
2. Durchsuche die Ähnlichkeitsmatrix nach dem Paar (x, y) mit dem höchsten Ähnlichkeitswert $sim(x, y)$ in der ganzen Matrix.
3. Verschmelze die Cluster x und y durch folgende Schritte:
   - Verschmelze die beiden Zeilen und Spalten in der Ähnlichkeitsmatrix, die für die Cluster x und y stehen, jeweils zu einer Zeile bzw. Spalte. In der neuen Zeile bzw. Spalte stehen dabei für die *Single Link* Methode die Maxima der beiden bisherigen Werte, für die *Complete Link* Methode die Minima der beiden bisherigen Werte und für die *Average Link* Methode die mit der Mächtigkeit der Cluster x und y gewichteten Mittelwerte.
   - Damit reduziert sich die Anzahl der Zeilen und Spalten in der Ähnlichkeitsmatrix ebenso wie die Anzahl der Cluster um 1. Der entstehende Cluster erhält die Nummer des kleineren der beiden Ursprungscluster.
4. Die Schritte 2. und 3. werden insgesamt $N - 1$ mal wiederholt, bis nur noch ein Cluster existiert.

Beispiel Das folgende Beispiel mit sechs Dokumenten soll den Ablauf des Al-

gorithmus verdeutlichen. Wir wenden dabei die *Average Link* Methode an.

Gegeben sei folgende initiale Ähnlichkeitsmatrix:

		Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6
		$\{D_1\}$	$\{D_2\}$	$\{D_3\}$	$\{D_4\}$	$\{D_5\}$	$\{D_6\}$
Cluster 1	$\{D_1\}$	∞	4	0	13	8	16
Cluster 2	$\{D_2\}$	4	∞	3	2	9	3
Cluster 3	$\{D_3\}$	0	3	∞	0	4	5
Cluster 4	$\{D_4\}$	13	2	0	∞	8	**20**
Cluster 5	$\{D_5\}$	8	9	4	8	∞	3
Cluster 6	$\{D_6\}$	16	3	5	**20**	3	∞

Die maximale Ähnlichkeit von 20 ergibt sich in dieser Matrix für die Cluster 4 und 6. Wir verschmelzen also die Zeilen und die Spalten dieser Cluster, indem wir jeweils den Durchschnitt der Ähnlichkeitswerte verwenden. Den entstehenden Cluster nennen wir Cluster 4. Cluster 6 wird aus der Ähnlichkeitsmatrix eliminiert:

		Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
		$\{D_1\}$	$\{D_2\}$	$\{D_3\}$	$\{D_4, D_6\}$	$\{D_5\}$
Cluster 1	$\{D_1\}$	∞	4	0	**14,5**	8
Cluster 2	$\{D_2\}$	4	∞	3	2,5	9
Cluster 3	$\{D_3\}$	0	3	∞	2,5	4
Cluster 4	$\{D_4, D_6\}$	**14,5**	2,5	2,5	∞	5,5
Cluster 5	$\{D_5\}$	8	9	4	5,5	∞

Der höchste Ähnlichkeitswert von 14,5 ergibt sich nun zwischen den Clustern 1 und 4. Wir verschmelzen also wieder die Zeilen und die Spalten dieser Cluster, indem wir jeweils den Durchschnitt der Ähnlichkeitswerte verwenden. Dabei ist nun aber zu beachten, dass in Cluster 4 zwei Elemente enthalten sind, während in Cluster 1 nur ein Element enthalten ist. So ergibt sich der Ähnlichkeitswert des neuen Clusters zu Cluster 2 als (4·1 + 2,5·2) / 3 = 3.

Den entstehenden Cluster nennen wir Cluster 1. Cluster 4 wird aus der Ähnlichkeitsmatrix eliminiert:

		Cluster 1	Cluster 2	Cluster 3	Cluster 5
		$\{D_1, D_4, D_6\}$	$\{D_2\}$	$\{D_3\}$	$\{D_5\}$
Cluster 1	$\{D_1, D_4, D_6\}$	∞	3	1,67	6,33
Cluster 2	$\{D_2\}$	3	∞	3	**9**
Cluster 3	$\{D_3\}$	1,67	3	∞	4
Cluster 5	$\{D_5\}$	6,33	**9**	4	∞

Der höchste Ähnlichkeitswert von 9 ergibt sich nun zwischen den Clustern 2 und 5. Wir verschmelzen also die Zeilen und die Spalten dieser Cluster. Den entstehenden Cluster nennen wir Cluster 2. Cluster 5 wird aus der Ähnlichkeitsmatrix eliminiert:

		Cluster 1	Cluster 2	Cluster 3
		$\{D_1, D_4, D_6\}$	$\{D_2, D5\}$	$\{D_3\}$
Cluster 1	$\{D_1, D_4, D_6\}$	∞	**4,67**	1,67
Cluster 2	$\{D_2, D_5\}$	**4,67**	∞	3,5
Cluster 3	$\{D_3\}$	1,67	3,5	∞

Der höchste Ähnlichkeitswert von 4,67 ergibt sich nun zwischen den Clustern 1 und 2. Wir verschmelzen die Zeilen und Spalten dieser Cluster, wobei bei der Durchschnittsbildung die Werte der bisherigen Cluster wieder gewichtet mit der Zahl der Dokumente in den Clustern eingehen müssen. Den entstehenden Cluster nennen wir Cluster 1. Cluster 2 wird aus der Ähnlichkeitsmatrix eliminiert:

		Cluster 1	Cluster 3
		$\{D_1, D_4, D_6, D_2, D5\}$	$\{D_3\}$
Cluster 1	$\{D_1, D_4, D_6, D_2, D_5\}$	∞	**2,4**
Cluster 3	$\{D_3\}$	**2,4**	∞

Schließlich werden die beiden verbliebenen Cluster verschmolzen:

		Cluster 1
		$\{D_1, D_4, D_6, D_2, D_5, D_3\}$
Cluster 1	$\{D_1, D_4, D_6, D_2, D_5, D_3\}$	∞

Abbildung 6.5 fasst den Ablauf des Beispiels zusammen.

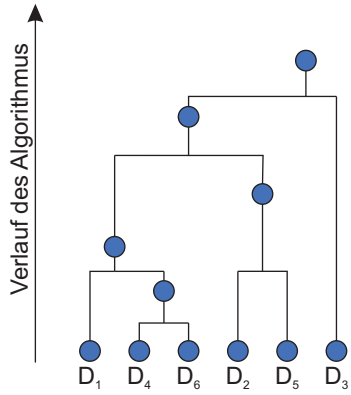


Abbildung 6.5 — Ablauf des Beispiels im Überblick

6.2.4 Anwendungsgebiete für Cluster-Verfahren

Die in diesem Abschnitt vorgestellten Techniken des Clustering lassen sich im Information Retrieval vielfältig einsetzen. An dieser Stelle können lediglich exemplarisch einige der Einsatzmöglichkeiten aufgezeigt werden:

Automatische Thesaurus-Erzeugung

automatische Thesaurus-Erzeugung

Um den Recall bei Anfragen zu erhöhen, wird häufig ein Thesaurus eingesetzt, der zu den Begriffen jeweils Synonyme und ähnliche Begriffe, allgemeinere Begriffe sowie speziellere Begriffe enthält. Im Rahmen einer Anfrageerweiterung können diese Terme zusätzlich in die Anfrage aufgenommen werden, um weitere Dokumente zu finden.

Statt auf einen manuell erstellten Thesaurus kann man bei einer solchen Anfrageerweiterung auch auf Term-Cluster zurückgreifen. Zur Erstellung der Cluster kann man z.B. auf die *Star-Technik* zurückgreifen und zu jedem Term ein Cluster mit den anderen Termen bilden, die einen gewissen Schwellenwert für die Ähnlichkeit überschreiten.

Ausgangspunkt der Überlegungen ist dabei die Term×Dokument-Matrix, bei der in den Spalten die Dokumente und in den Zeilen die Terme angetragen sind. Die Elemente der Matrix geben an, wie oft der Begriff im Dokument vorkommt. Als Beispiel sei hier die folgende

Term×Dokument-Matrix mit 10 Termen und 7 Dokumenten angenommen:

	D_1	D_2	D_3	D_4	D_5	D_6	D_7
t_1	0	4	0	3	0	1	0
t_2	3	0	4	0	0	0	0
t_3	0	2	0	2	0	0	0
t_4	1	0	0	4	0	0	0
t_5	2	0	2	0	3	0	0
t_6	0	3	0	0	3	0	2
t_7	2	0	7	0	0	0	0
t_8	0	4	0	4	0	0	0
t_9	0	0	0	0	3	0	6
t_{10}	0	1	0	0	2	0	4

Aus dieser Term×Dokument-Matrix kann nun z.B. mit dem Dice-Koeffizient die Term×Term-Ähnlichkeitsmatrix gewonnen werden. Diese Matrix ist im Folgenden angegeben. Dabei sind die Ähnlichkeitswerte fett gedruckt, die den willkürlich gewählten Schwellenwert 40 überschreiten.

Der Übersichtlichkeit halber wurde für die Berechnung der folgenden Tabelle nur der Zähler des Dice-Koeffizienten $(2 \cdot \sum_{k=1}^{t}(w_{ik} \cdot w_{jk}))$ verwendet:

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
t_1	**52**	0	28	24	0	24	0	**56**	0	8
t_2	0	**50**	0	6	28	0	**68**	0	0	0
t_3	28	0	16	16	0	12	0	32	0	4
t_4	24	6	16	34	4	0	4	32	0	0
t_5	0	28	0	4	34	18	36	0	18	12
t_6	24	0	12	0	18	**44**	0	24	**42**	34
t_7	0	**68**	0	4	36	0	**106**	0	0	0
t_8	**56**	0	32	32	0	24	0	**64**	0	8
t_9	0	0	0	0	18	**42**	0	0	**90**	**60**
t_{10}	8	0	4	0	12	34	0	8	**60**	**42**

Wir würden also in diesem kleinen Beispiel folgende ähnliche Terme ermitteln:

- für den Term t_1: t_8

- für den Term t_2: t_7
- für den Term t_6: t_9
- für den Term t_7: t_2
- für den Term t_8: t_1
- für den Term t_9: t_6 und t_{10}
- für den Term t_{10}: t_9

Die entsprechenden Terme könnten nun bei einer Anfrageerweiterung eingesetzt werden.

Suche entlang einer Cluster-Hierarchie

Suche durch die Cluster-Hierarchie

Eine weitere Nutzung von Cluster-Techniken besteht darin, eine Suche durch die Cluster-Hierarchie zu leiten. Dabei navigiert der Informationssuchende gewissermaßen in dem in Abbildung 6.4 skizzierten Dendogramm. Jedes Cluster wird dabei durch ein typisches Element – z.B. das Dokument, welches dem Zentroid des Clusters am ähnlichsten ist – repräsentiert. Von der Wurzel des Dendogramms ausgehend betrachtet man dann immer die Repräsentanten der beiden unterliegenden Cluster und navigiert zu dem Cluster, dessen Repräsentant für den gegebenen Informationswunsch relevanter erscheint. Dieses Vorgehen wird durch die Cluster-Hierarchie bis zur Blattebene fortgesetzt.

Der Vorteil dieses Vorgehens liegt darin, dass der Informationssuchende keine Anfrage formulieren muss. Er kann sich in seiner Suche durch die Cluster-Hierarchie leiten lassen. Das Vorgehen gehört damit zu den Verfahren des *Browsing*, die wir im nächsten Abschnitt 6.3 genauer betrachten werden.

Aufbereitung des Ergebnisses einer Anfrage

Aufbereitung des Ergebnisses einer Anfrage

Insbesondere bei IR-Modellen, die ein unstrukturiertes Ergebnis liefern – wie z.B. beim Booleschen Retrieval – erscheint es interessant, die Ergebnismenge durch die Bildung thematischer Cluster nachträglich zu strukturieren.

Ähnliches erscheint aber durchaus auch bei Verfahren sinnvoll, die als Ergebnis bereits eine Rangordnung liefern. So könnte man z.B. bei einer

Anfrage nach »Motorradfahren in den Alpen« versuchen durch eine Clusterbildung über dem Anfrageergebnis Ergebnisdokumente mit verschiedenen inhaltlichen Ausrichtungen voneinander zu trennen. So könnte es z.B. Dokumente mit Routenvorschlägen, Reiseberichte, Berichte über das erhöhte Verkehrsaufkommen, etc. geben, die im Ergebnis in getrennten Rangordnungen ausgewiesen werden könnten.

Ein Problem der automatischen Cluster-Verfahren besteht hier allerdings darin, dass im Allgemeinen keine explizite Beschreibung der Eigenschaften der Cluster existiert. D.h. selbst wenn es dem Cluster-Verfahren gelingt – z.B. durch eine geeignete Wahl des Schwellwertes – Reiseberichte sauber von anderen Dokumenten zu trennen, kann das Verfahren nicht erkennen, dass es sich bei den Dokumenten eines bestimmten Clusters eben um Reiseberichte handelt.

Verfahren der manuellen Klassifikation – wie z.B. die Dezimalklassifikation – sind hier klar im Vorteil, weil sie aufgrund der Klassifikation und der Facettierung eine Gruppierung mit sprechender Benennung der einzelnen Gruppen erlauben. Dem steht als Nachteil aber der erforderliche manuelle Aufwand zur Klassifikation gegenüber.

Wartung einer Klassifikation

Wartung einer Klassifikation

Bei der Zuordnung von Dokumenten zu einer manuell erstellten Klassifikation, kann man ebenfalls Cluster-Verfahren einsetzen.

Um z.B. ein neues Dokument in eine bestehende Klassifikation einzuordnen, kann man die von der Klassifikation gebildeten Gruppen als Cluster auffassen und das neue Dokument dem nach der *Single Link*, *Complete Link*, *Average Link*, *Zentroid* oder *Ward* Methode am besten passenden Cluster zuordnen (vgl. Abschnitt 6.2.3).

In einem zweiten Schritt kann dann ggf. analog zur *Reallocation-Technik* eine Überprüfung der aktuellen Zuordnung der Dokumente zu den Klassifikationsgruppen erfolgen (vgl. Abschnitt 6.2.2).

Schließlich kann man mit Hilfe der Ähnlichkeitsmatrix überprüfen, ob eine vorgenommene Klassifikation schlüssig erscheint. Dazu sollten die Ähnlichkeitswerte für die Dokumente, die der gleichen Klasse zugeordnet sind, höher sein als die Ähnlichkeitswerte von Dokumenten in verschiedenen Klassen. Sind die Ähnlichkeitswerte innerhalb einer Klasse zum Teil sehr hoch, so sollte man prüfen, ob eine weitere Untergliederung dieser Klasse sinnvoll erscheint.

6.3 Browsing

6.3.1 Begriff und Arten des Browsing

Browsing

Unter *Browsing* wollen wir hier allgemein den navigierenden Zugriff auf Informationen verstehen. Treffend drückt dieses Begriffsverständnis die Definition in Gablers Wirtschaftsinformatik-Lexikon [SGR97] aus:

> **browsing**, eine Art der Informationssuche, die zwar zum Ziel haben kann, eine bestimmte informationelle Einheit aufzufinden, bei der aber das Durchstöbern bzw. Überfliegen benachbarter Einheiten mit dem Zweck andere, nicht spezifizierte Informationen zu entdecken, ebenfalls eine Rolle spielt. Dabei kann der Effekt, durch das nicht-zielgerichtete Suchen Informationen zu finden, durchaus in den Vordergrund rücken.
>
> Vergleichbar ist diese Art des Suchens mit dem Schweifenlassen des Blicks über Bibliotheksregale oder dem Durchblättern von Büchern.
>
> In der Wirtschaftsinformatik spielt *browsing* insbesondere bei nicht-linearen Medien wie Hypertext eine Rolle, da hier während des Verfolgens semantischer (inhaltlicher) Verknüpfungen besondere »Mitnahmeeffekte« entstehen.

Browsing setzt dabei offensichtlich eine entsprechende Aufbereitung der Dokumentenkollektion voraus. Hierfür sind verschiedene, von uns zum Teil bereits betrachtete Ansätze denkbar:

- **Literaturverzeichnisse und Referenzen**

  Literaturverzeichnisse und Referenzen

  Bereits die in (fast) allen wissenschaftlichen Arbeiten enthaltenen Referenzen und Literaturverzeichnisse bilden eine erste Grundlage für das Browsing. Durch Referenzen wird es möglich, gezielt detailliertere Informationen zu einzelnen Gebieten zu betrachten. Allerdings ist dabei der Aufwand für das »Verfolgen einer Referenz« im Allgemeinen recht hoch (Beschaffen eines Buchs oder Artikels in einer Bibliothek oder im Buchhandel).

  Wichtig ist aber, dass bereits hier das Prinzip, bei Bedarf auf detailliertere Informationen zugreifen zu können, unterstützt wird.

- **Klassifikationen**

Klassifikationen

Auf Basis einer vorhandenen Klassifikation sind viele Möglichkeiten des Browsing gegeben. So kann man sich, wenn man aktuell ein Dokument betrachtet, auch andere Dokumente der gleichen Klasse ansehen. Ferner ist es möglich in der Klassenhierarchie allgemeiner oder spezieller zugeordnete Dokumente zu betrachten. Auch eine ggf. vorhandene Facettierung unterstützt das Browsing, indem z.B. gezielt auf Wörterbucher zum betrachteten Fachgebiet zugegriffen werden kann.

Eine weitere Möglichkeit bildet ein Browsing beginnend mit der Wurzel der Klassifikation, bei dem man sich sukzessive durch Auswahl der entsprechenden Unterklassen den interessanten Dokumenten nähert.

Insbesondere bei dieser Art des Browsing anhand einer Klassifikation ist die Analogie zu dem in der obigen Definition aus Gablers Wirtschaftsinformatik-Lexikon angeführten »Schweifenlassen des Blicks über Bibliotheksregale« offensichtlich, da die Aufstellung der Bücher in den Regalen ja typischerweise anhand einer Klassifikation erfolgt.

- **Hypertext-Dokumente**

Hypertext-Dokumente

Hypertext-Formate wie HTML bieten die Möglichkeit, Verweisen auf andere Stellen im selben Dokument oder in anderen Dokumenten sehr einfach zu folgen. Damit ist aber zunächst nur die technische Möglichkeit geschaffen. In jedem einzelnen Dokument müssen nun an den geeigneten Stellen sinnvolle Verweise eingefügt werden.

Im Gegensatz zu Klassifikationen erfolgt dabei die Navigation nicht auf Basis eines in sich geschlossenen Klassifikationssystems mit klar vorgegebener Struktur, sondern in einem nicht explizit gegebenen Netz von Verweisen. Daraus ergibt sich fast zwangsweise das Risiko, sich in diesem Netzwerk auf der Suche nach geeigneten Informationen quasi zu »verlaufen«. Dies führt dazu, dass eine Informationssuche z.B. im Internet immer aus einem gezielten Ausnutzen der gegebenen Möglichkeiten in Form von

  - Suchmaschinen,
  - Katalogen (vgl. Abschnitt 6.1.3) und
  - dem Verfolgen von Links in Dokumenten

bestehen sollte.

- **Cluster und Cluster-Hierarchien**

  Cluster
  Cluster-Hierarchien

  Auch Cluster und insbesondere Cluster-Hierarchien können als Basis für ein Browsing genutzt werden. So können zu einem aktuell betrachteten Dokument vom System Dokumente des gleichen Clusters als »inhaltlich ähnliche Dokumente« zur Betrachtung angeboten werden.

  Eine andere Möglichkeit bildet der bereits im Abschnitt 6.2.4 beschriebene Ansatz anhand von typischen Elementen entlang einer Cluster-Hierarchie zu suchen.

- **Visuelle Darstellungen von Dokumentenkollektionen**

  Die Analogie zwischen *Browsen* und dem *Navigieren in einem Informations- oder Dokumentraum* legt bereits nahe, dass man in einem solchen Informations- oder Dokumentraum auch Navigationshilfen wie z.B. eine Landkarte sinnvoll einsetzen könnte. Mit derartigen Problemstellungen beschäftigt man sich im Bereich der Informationsvisualisierung. Dabei versucht man der Tatsache Rechnung zu tragen, dass gerade komplexe Zusammenhänge leichter graphisch als textuell erfasst werden können.

  Informations-visualisierung

  Wir werden uns in den folgenden Abschnitten etwas ausführlicher mit dem Bereich der *Informationsvisualisierung* beschäftigen.

6.3.2 Grundlagen der Visualisierung von Dokumentenkollektionen

In seinem grundlegenden Buch *Designing the User Interface* [Shn98] formuliert Ben Shneiderman das folgende »visual-information-seeking mantra«:

> Overview first, zoom and filter, then details on demand

Die Aussage ist klar: Jeder Ansatz zur Visualisierung einer Informations- oder Dokumentenkollektion muss auf der obersten Ebene zunächst einen Überblick geben, der versucht alle Dokumente oder alle Gruppen von Dokumenten darzustellen. Hierzu kann man typischerweise auf die Anordnung der Dokumente oder Gruppen in einem 1-, 2- oder mehrdimensionalen Raum zurückgreifen. 1-dimensional wäre z.B. eine einfache sequentielle Auswahlliste. Dies mag zunächst profan erscheinen.

Wir sollten uns aber klar machen, dass jedes IR-System, das als Ergebnis einer Anfrage eine Rangordnung von Dokumenten liefert, eine solche 1-dimensionale Darstellung nutzt. Eine zweidimensionale Darstellung könnte entweder versuchen, die inhaltlichen Aspekte in zwei thematische Dimensionen aufzuspalten oder ein weiteres Kriterium wie das Veröffentlichungsjahr der Dokumente als zweite Dimension verwenden.

Auf der Basis dieser überblicksartigen Darstellung der Kollektion sollten nun Zoom- und Filter-Funktionalitäten angeboten werden, um sich mit einem Teilausschnitt der Kollektion näher beschäftigen zu können. Schließlich sollten bei Bedarf Detailinformationen zu einzelnen Dokumenten oder Dokumentgruppen angeboten werden.

Etwas genauer werden die Aufgaben bei der Informationsvisualisierung in Tabelle 6.3 dargestellt.

Aufgabe	Beschreibung
Overview	Gib einen Überblick über die gesamte Kollektion.
Zoom	Stelle die aktuell interessierenden Elemente vergrößert (detaillierter) dar.
Filter	Verberge Elemente, die aktuell nicht von Interesse sind.
Details-on-Demand	Erlaube die Selektion eines Elements oder einer Gruppe und zeige die zugehörigen Detailinformationen an.
Relate	Zeige die Zusammenhänge und Verbindungen zwischen den Elementen auf (z.B. inhaltliche Bezüge oder Zitationen).
History	Die vorangegangenen Aktionen sollten gespeichert werden, um Operationen wie *Undo*, *Replay* oder eine *proaktive Verfeinerung* durchführen zu können.
Extract	Erlaube die Extraktion von Teilkollektionen oder Anfrageparametern.

Tabelle 6.3 — Aufgaben bei der Visualisierung von Dokumentenkollektionen nach [Shn98]

6.3.3 Beispiele für die Visualisierung von Dokumentenkollektionen

In diesem Abschnitt wollen wir die von Shneiderman angegebenen Typen der Informationsvisualisierung kurz vorstellen. Dazu werden wir uns im Wesentlichen auf Beispiele aus dem Buch von Shneiderman [Shn98] und dem *Atlas of Cyberspace* von Martin Dodge

und Rob Kitchin beziehen [DK01] (vgl. hierzu auch die Website http://www.cybergeography.org/atlas, letzter Zugriff 20.11.2006).

1-dimensionale lineare Darstellung

1-dimensionale lineare Darstellung

Eine eindimensionale Darstellung entspricht letztlich einfach einer sortierten Liste mit den Dokumenten. Das Sortierkriterium kann dabei z.B. durch eine Klassifikation gewonnen werden. Ebenso ist aber auch eine Sortierung nach dem Veröffentlichungsdatum oder der veröffentlichenden Organisation denkbar.

Als Beispiel für eine eindimensionale Darstellung sei hier die Dokumentation der Java Programmierschnittstelle (API) betrachtet (siehe Abbildung 6.6). Die »Dokumentenkollektion« umfasst dabei die Beschreibungen zu den weit über 500 Klassen der Java 2 Standard Edition. Dazu wird auf der linken Seite oben eine alphabetisch sortierte Liste der Pakete, in denen die Klassen organisiert sind, und darunter eine ebenfalls alphabetisch sortierte Liste der Klassen angeboten. Mit der Möglichkeit, durch die Auswahl eines Pakets die angezeigten Klassen einzuschränken, bietet auch diese einfache Oberfläche bereits Filtermöglichkeiten.

2-dimensionale Darstellung

2-dimensionale Darstellung

Bei 2-dimensionalen Ansätzen zur Visualisierung werden die Dokumente in einer Ebene angeordnet. Welche Kriterien dabei für die Anordnung verwendet werden, ist offen.

- So können wir im Sinne des Latent Semantic Indexing (vgl. Abschnitt 7.3) die Dimensionalität der Beschreibungsvektoren auf zwei reduzieren und diese Dimensionen zur Anordnung im 2-dimensionalen Raum verwenden.
- Alternativ könnte man auch mit einem hierarchischen Verfahren Cluster von Termen bilden und die Clusterbildung solange fortsetzen, bis nur noch zwei Termcluster existieren. Die Koordinatenwerte der Beschreibungsvektoren über die entsprechenden Dimensionen könnte man nun für jedes der beiden Cluster mitteln und die beiden sich so ergebenden Werte zur Anordnung im 2-dimensionalen Raum nutzen.

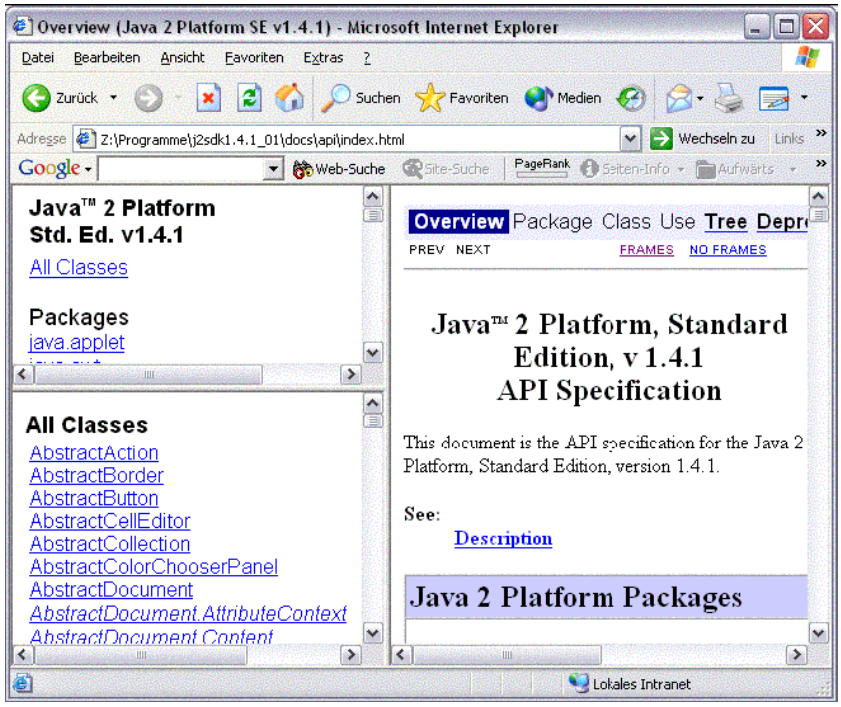


Abbildung 6.6 — Die Java 2 API Specification als Beispiel für eine eindimensionale Darstellung des Dokumentraumes

- Schließlich werden in der Literatur häufig auch Ansätze beschrieben, die auf Anziehung und Abstoßung beruhen. Dabei werden die Elemente zunächst initial im Raum angeordnet. Danach werden die Ähnlichkeiten zwischen den Elementen als Anziehungskräfte interpretiert und die Dokumente entsprechend verschoben.

Abbildung 6.7 zeigt beispielhaft die hierarchische 2-dimensionale so genannte »ET-Map«, die einen Informationsraum mit über 100.000 Webseiten zum Thema Unterhaltung visualisiert. ET-Map ist eine von mehreren Visualisierungstechniken, die von einer Foschergruppe um Hsinchun Chen am Artificial Intelligence Lab der University of Arizona, USA entwickelt wurde.

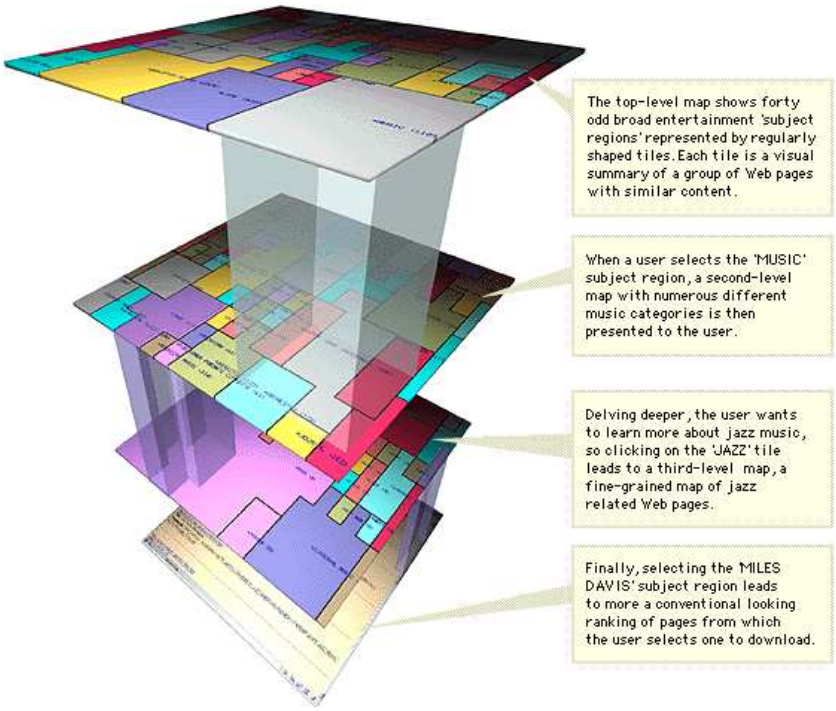


Abbildung 6.7 — ET-Map von Hsinchun Chen (Artificial Intelligence Lab der University of Arizona, USA) – übernommen von Mappa.Mundi Magazine http://www.mundi.net/maps/maps_009/etmap_feb.html, letzter Zugriff 11.12.2007

3-dimensionale Darstellung

3-dimensionale Darstellung

Bei dreidimensionalen Darstellungen wird eine weitere Dimension z.B. durch Höhenlinien oder andere Techniken mit in die Visualisierung aufgenommen. Abbildung 6.8 zeigt beispielhaft eine »Informationskarte« (*NewsMap*) für Nachrichten zu internationalen Themen. Die Karte fasst 121 Nachrichtenberichte vom 13. Dezember 1999 zusammen. Die dritte Dimension, die als Höhe interpretiert wird, gibt dabei an, wie viele Berichte zu dem jeweiligen Themenbereich existieren.

NewsMaps wurde mit dem System ThemeScape der Firma Cartia realisiert. Cartia wurde von Aurigin Systems, Inc. übernommen, und NewsMaps wird heute nicht mehr betrieben. ThemeScape wird aber weiter angeboten.

Zusätzlich ist im Übrigen die Aufnahme weiterer Charakteristika in eine Visualisierung möglich. Hierzu können z.B. Icons sowie die Größe oder Form von Gegenständen genutzt werden. Ein Überladen der Darstellung führt dabei aber nicht zu einer besseren Lesbarkeit.

Zeitliche Darstellung

zeitliche Darstellung

Häufig ist auch die zeitliche Dimension für die Betrachtung einer Dokumentenkollektion von Bedeutung. So kann z.B. das Erscheinungsdatum eines Buches oder die Gültigkeitsdauer eines Gesetzestextes von Bedeutung sein. In der »perspective wall« von Xerox PARC werden die Dokumente deshalb auf einer Zeitwand abgebildet, auf der der Benutzer navigieren kann. Abbildung 6.9 gibt einen Eindruck von diesem Ansatz.

Multidimensionale Darstellung

multidimensionale Darstellung

Zwei bis drei Dimensionen lassen sich typischerweise in einem entsprechenden Raum visualisieren. Zusätzlich können weitere Informationen z.B. über die Dokumentart oder die Dokumentlänge durch Icons, Farben oder die Größe der graphischen Symbole repräsentiert werden. Spätestens bei mehr als sechs Dimensionen versagen solche Ansätze aber. Einen Ansatz, wie in solchen Fällen verfahren werden kann, demonstriert die folgende auf Matt Ward vom Worcester Polytechnic Institute zurückgehende Darstellung, die im Beispiel verschiedene Kriterien zu Kraftfahrzeugen visualisiert. Analog könnten aber auch Kriterien zu Dokumenten visualisiert werden.

Abbildung 6.8 — Eine NewsMap als Beispiel für eine kartenartige Visualisierung von Dokumenten – Quelle: http://www.cybergeography.org/atlas, letzter Zugriff 20.11.2006

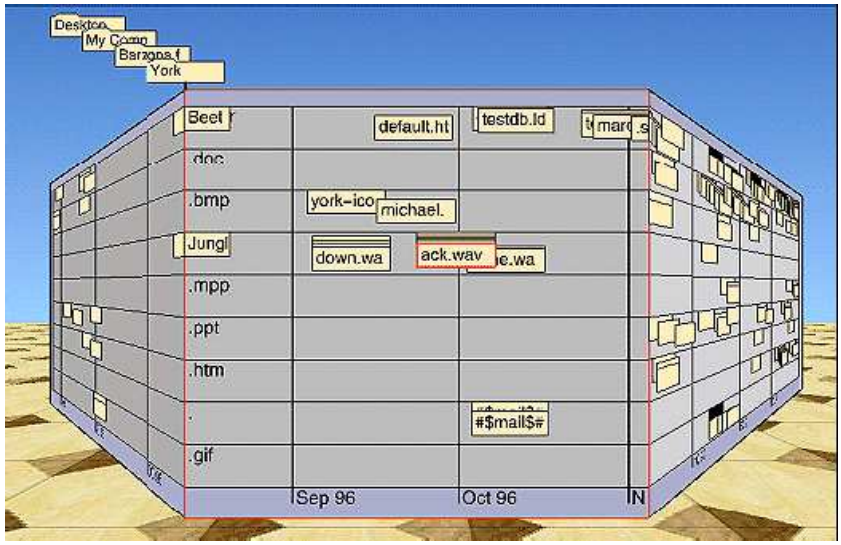


Abbildung 6.9 — Die »perspective wall« von Xerox PARC, Palo Alto, CA. [MRC91] – übernommen von http://syntheti.cc/ndimensions/zb/wall/index.html, letzter Zugriff 11.12.2007

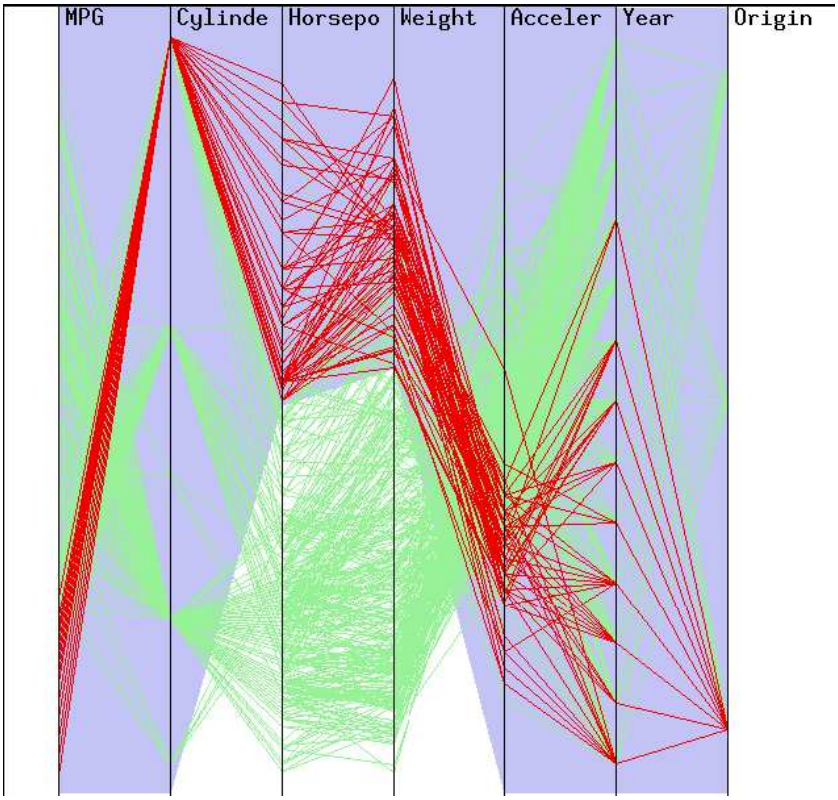


Abbildung 6.10 — Visualisierung von 7 Dimensionen zur Charakterisierung von Automobilen durch parallele Koordinaten [ID89]

Baumartige Darstellung

baumartige Darstellung

Eine häufig gewählte Visualisierungsform bilden Bäume. Wir haben derartige Bäume bereits im Zusammenhang von Klassifikationen und Cluster-Verfahren kennen gelernt. Abbildung 6.11 zeigt eine etwas gefälligere dreidimensionale Visualisierung einer Baumstruktur am Beispiel eines Verzeichnisbaumes.

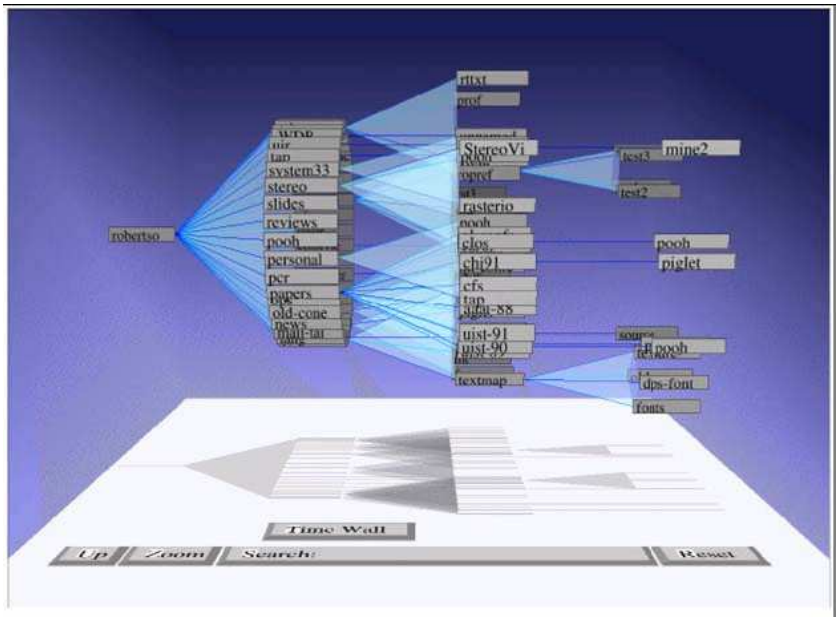


Abbildung 6.11 — Ein »cam-tree«, der die Verzeichnisstruktur eines Benutzers visualisiert (Xerox PARC, Palo Alto, CA.) [RMC91]

Darstellung als Netzwerk

Darstellung als Netzwerk

Da nicht alle Strukturen baumartig sind, bieten sich Netzwerke zur Visualisierung komplexerer Zusammenhänge an. Ein Problem ist allerdings, dass derartige Netzwerke schnell unübersichtlich werden. In diesem Fall ist daher die Beachtung des »visual-information-seeking mantra« von Shneiderman: *Overview first, zoom and filter, then details on demand* von besonderer Bedeutung.

Abbildung 6.12 zeigt als Beispiel das Werkzeug *Site Manager* von Silicon Graphics zur Visualisierung von Web-Seiten. Dabei wird die Farbe als weiteres Mittel genutzt, um in diesem Fall die Art der Seite zu visualisieren.

Die verschiedenen Möglichkeiten zur Visualisierung müssen im konkreten Einsatzgebiet gezielt eingesetzt werden. Die in diesem Kurs betrachteten Techniken des Information Retrieval können dabei die Ausgangsbasis für eine inhaltsbasierte Darstellung liefern, indem sie z.B. die Einordnung in den einzelnen Dimensionen bestimmen.

6.4 Zusammenfassung

Wir haben im aktuellen Kapitel drei wichtige Bereiche des Information Retrieval betrachtet.

Klassifkationen

Klassifikationen bilden seit langem eine wesentliche Stütze des Information Retrieval. Gerade manuell erstellte Klassifikationsinformationen bieten eine sehr fundierte Basis für die Informationssuche. Dieser Wert wird auch dadurch deutlich, dass sich für die Suche im Internet neben reinen Suchmaschinen wie Google Kataloge wie Yahoo! etabliert haben. Beide Ansätze stehen dabei keineswegs in Konkurrenz zueinander, sondern ergänzen sich. Klassifikationen können dabei durch Facettierungen wesentlich an Ausdruckskraft gewinnen. Für eine umfassendere Auseinandersetzung mit dem Thema *Klassifikationen* sei insbesondere [Buc89] empfohlen.

Cluster-Verfahren
automatische Klassifikation

Cluster-Verfahren werden auch als *automatische Klassifikation* beschrieben. Sie können z.B. auf der Basis von Beschreibungsvektoren für die Dokumente recht einfach realisiert werden. Etwas problematisch ist aber bei größeren Kollektionen ihre algorithmische Komplexität. Für einen etwas genaueren Überblick über das Gebiet sei der interessierte Leser auf [Ras92] und [KM97] verwiesen.

Informationsvisualisierung

Schließlich haben wir uns mit dem Aspekt der Visualisierung von Informationsräumen oder Dokumentenkollektionen beschäftigt. Die vielfältigen Möglichkeiten in diesem Bereich müssen im konkreten Einsatzgebiet zielgerichtet eingesetzt werden. Wichtig erscheint dabei einerseits eine Beschränkung auf das Wesentliche und Sinnvolle und andererseits der zielgerichtete, von den Anforderungen geprägte Einsatz der Möglichkeiten. Letztlich müssen sich auch die visuell eindrucksvollsten Oberflächen

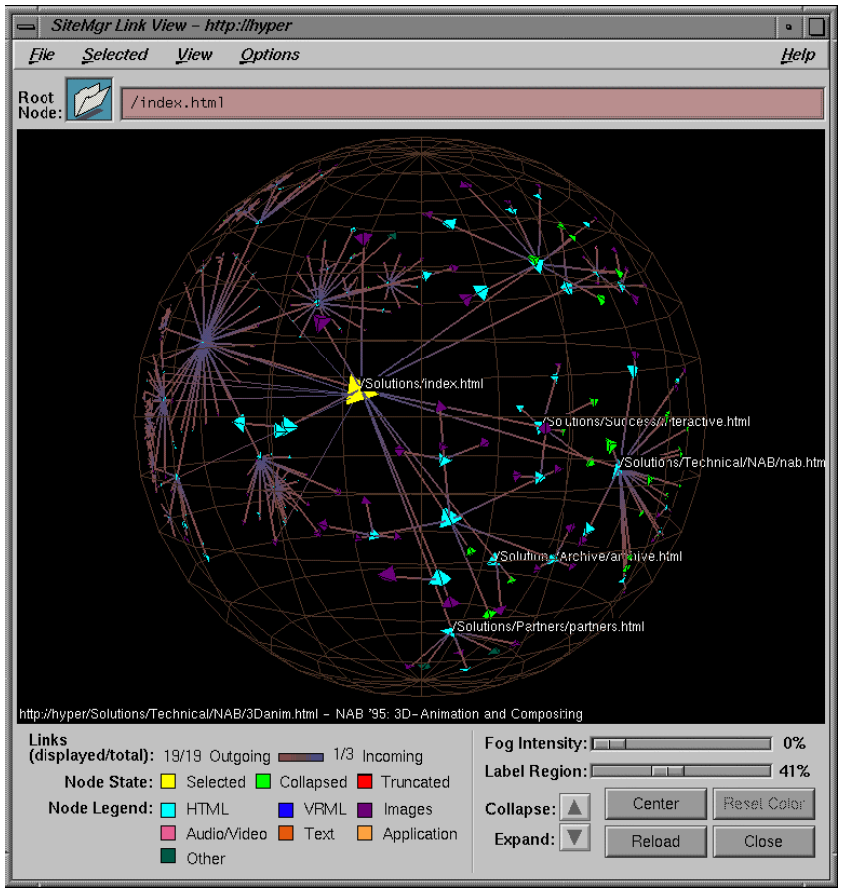


Abbildung 6.12 — *Site Manager* von Silicon Graphics zur Visualisierung von Web-Seiten – übernommen von http://www.nicolas-guillard.com/cybergeography-fr/atlas/atlas.html, letzter Zugriff 11.12.2007

an der Effizienz im praktischen Einsatz messen lassen. Sehr viele Anregungen zu diesem Gebiet findet man in [Shn98] und [DK01].

7 Weitere IR-Modelle

Zur Abrundung unserer Betrachtung verschiedener IR-Modelle wollen wir in diesem Kapitel weiterführende Ansätze betrachten, die die Forschung im Information Retrieval beeinflusst haben und beeinflussen, und zum Teil auch in kommerziellen Systemen genutzt werden.

7.1 Das Binary Independence Retrieval-Modell (BIR-Modell)
7.2 Okapi BM 25
7.3 Latent Semantic Indexing
7.4 Zusammenfassung

7.1 Das Binary Independence Retrieval-Modell (BIR-Modell)

probabilistisches Information Retrieval

Ein Vorwurf, der den in den bisherigen Kapiteln betrachteten Modellen des Information Retrieval häufig gemacht wird, ist ihre mangelnde theoretische Fundierung. Das Probabilistische Information Retrieval versucht daher auf einem klaren theoretischen Fundament zu arbeiten. Dabei wird die Aufgabenstellung des Information Retrieval in der Weise interpretiert, dass man die Wahrscheinlichkeit – oder Chance – dafür berechnen möchte, dass ein Dokument für eine gegebene Anfrage relevant ist.

Binary Independence Retrieval

Das klassische Modell des Probabilistischen Information Retrieval wurde 1976 von Robertson und Sparck Jones [RJ76] vorgestellt. Dieses *Binary Independence Retrieval-Modell* (BIR-Modell) geht – wie auch das Vektorraummodell – davon aus, dass die einzelnen Dimensionen der Repräsentationsvektoren voneinander unabhängig sind. Zusätzlich wird beim BIR-Modell aber nur mit binären Termgewichten in den Dokumentrepräsentationen gearbeitet – also $w_{dk} \in \{0, 1\}$.

Ziel des BIR-Modells ist die Berechnung der Wahrscheinlichkeit, dass ein Dokument D für eine Anfrage Q relevant ist. Wenn wir mit $R^+(Q)$ die

zur Anfrage Q relevanten Dokumente bezeichnen, dann suchen wir also für jedes Dokument D die Wahrscheinlichkeit:

$$P(D \in R^+(Q)) \tag{7.1}$$

Das BIR-Modell arbeitet nun – wie die meisten anderen IR-Modelle – nicht auf den Dokumenten selbst, sondern auf Dokumentrepräsentationen, die aus den Dokumenten abgeleitet werden. Bisher haben wir die Repräsentation eines Dokumentes D mit $\mathcal{D}$ bezeichnet. Da der Aspekt der Repräsentation für das BIR-Modell sehr wichtig ist, werden wir die Repräsentation eines Dokumentes D in diesem Kapitel zur besseren Unterscheidung mit $\beta(D)$ bezeichnen.

Damit versucht das BIR-Modell – weil es ja auf den Repräsentationen der Dokumente arbeitet – statt $P(D \in R^+(Q))$ nun genauer die folgende Wahrscheinlichkeit zu berechnen:

$$P(D' \in R^+(Q) | \beta(D') = \beta(D)) \tag{7.2}$$

Kurz gesagt berechnen wir also die Wahrscheinlichkeit dafür, dass ein Dokument D', das die gleiche Repräsentation wie D hat, relevant bezüglich Q ist. Dies drücken wir durch eine bedingte Wahrscheinlichkeit $P(A|B)$ aus, die die Wahrscheinlichkeit für A für den Fall berechnet, dass B gilt. Wir fassen also alle Dokumente mit gleicher Repräsentation zu einer Klasse zusammen und bestimmen die Wahrscheinlichkeit dafür, dass ein Dokument relevant ist, unter der Bedingung, dass es der gleichen Klasse wie D angehört.

Beispiel

Gegeben sei eine Dokumentenkollektion mit den neun Dokumenten D_1 bis D_9 und eine Anfrage Q. Die folgende Tabelle zeigt die Repräsentationen dieser Dokumente, wobei gemäß dem BIR-Modell $w_{dk} \in \{0, 1\}$ gilt. Zusätzlich ist die Relevanz der Dokumente im Hinblick auf die Anfrage Q angegeben.

Dokument	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9
w_{d1}	1	1	1	0	0	0	1	1	0
w_{d2}	1	0	1	0	1	0	1	0	0
Relevanz bzgl. Q	+	+	+	-	-	-	-	-	-

Für diese konkrete Dokumentenkollektion und die gegebene Anfrage Q kann man nun empirisch in Experimenten die Wahrscheinlichkeiten $P(D' \in R^+(Q)|\beta(D') = \beta(D))$ ermitteln. Wir erhalten:

- $P(D' \in R^+(Q)|\beta(D') = (1,1)) = 2/3$, weil zwei der drei Dokumente mit dem Beschreibungsvektor $(1,1)$ relevant sind und eines nicht.
- $P(D' \in R^+(Q)|\beta(D') = (1,0)) = 1/2$, weil eines der zwei Dokumente mit dem Beschreibungsvektor $(1,0)$ relevant ist und eines nicht.
- $P(D' \in R^+(Q)|\beta(D') = (0,1)) = 0$, weil das einzige Dokument mit dem Beschreibungsvektor $(0,1)$ nicht relevant ist.
- $P(D' \in R^+(Q)|\beta(D') = (0,0)) = 0$, weil keines der drei Dokumente mit dem Beschreibungsvektor $(0,0)$ relevant ist.

Wenn wir ein weiteres Dokument D_{10} mit $\beta(D_{10}) = (1,0)$ betrachten, dessen Relevanz im Hinblick auf Q nicht bekannt ist, dann ordnen wir D_{10} aufgrund seiner Repräsentation eine Relevanzwahrscheinlichkeit von $1/2 = 50\%$ zu.

Dieses Beispiel deutet bereits eines der Probleme des BIR-Modells an. Zur empirischen Ermittlung der Relevanzwahrscheinlichkeiten für die 2^t möglichen Repräsentationen sind auf jeden Fall weit mehr als 2^t Dokumente zu betrachten. Für realistische Werte von t, die weit jenseits von 100 oder auch 1000 liegen, ist dies offensichtlich nicht möglich. Wir werden im Folgenden sehen, wie dieses Problem beim BIR-Modell gelöst wird.

Grundidee des BIR-Modells

Im Folgenden skizzieren wir die dem BIR-Modell zugrunde liegende Idee in Anlehnung an [BYRN99].

Zunächst unterstellt das BIR-Modell, dass es zu einer Anfrage eines Benutzers eine Menge von Dokumenten gibt, die genau die relevanten Dokumente beinhaltet und damit die *ideale Antwort* darstellt. Wenn wir nun eine genaue Beschreibung dieser idealen Antwort hätten, dann wäre es kein Problem, ihre Elemente zu ermitteln. Da wir aber die ideale

ideale Antwort

Antwort nicht kennen, fassen wir den Anfrageprozess als den Prozess zur Bestimmung der Eigenschaften dieser idealen Antwort auf.

Offen ist dabei zunächst die Frage, worauf sich diese »Eigenschaften« beziehen. Das BIR-Modell unterstellt, dass sich diese Eigenschaften an den in den Dokumenten enthaltenen Termen festmachen lassen, die in der Repräsentation der Dokumente abgebildet sind. Es verwendet hierzu Vektoren der Dimension t, die für jeden Indexterm als Komponente entweder eine 1 oder eine 0 enthalten, je nachdem ob ein Indexterm das Dokument charakterisiert oder nicht. Typischerweise können diese Repräsentationen dadurch gewonnen werden, dass man die Komponenten zu den Termen, die im Dokument vorkommen, auf 1 setzt und die anderen Komponenten auf 0.

Nun wissen wir also, dass wir die Eigenschaften der idealen Antwort auf Basis der Repräsentationen beschreiben wollen. Wenn wir die ideale Antwort kennen würden, könnten wir leicht die Repräsentationen der Dokumente in dieser idealen Antwort angeben. Da wir die ideale Antwort aber nicht kennen, muss ein initialer Schritt unternommen werden, der mit einer groben Näherung dieser Eigenschaften ein erstes Ergebnis liefert. Wenn wir die als relevant erachteten Dokumente aus diesem ersten initialen Ergebnis betrachten und als repräsentativ für alle Dokumente in der idealen Antwort erachten, dann können wir aus ihnen eine vorläufige probabilistische Beschreibung der Eigenschaften der idealen Antwort ableiten. Im Zusammenspiel mit dem Anfragenden kann nun mittels Relevance Feedback ein iterativer Prozess stattfinden, in dem diese probabilistische Beschreibung weiter verfeinert wird.

Relevance Feedback

Ablauf des iterativen/interaktiven Prozesses

Der Ablauf des iterativen und interaktiven Prozesses zur Bestimmung der Eigenschaften der idealen Antwort kann nun wie folgt beschrieben werden:

- Zunächst wird auf Basis einer sehr groben Abschätzung der Eigenschaften der idealen Antwort ein initiales Ergebnis ermittelt.

initiales Ergebnis

- Der Anfragesteller betrachtet die ermittelten Dokumente in diesem ersten Ergebnis und entscheidet – zumindest für die ersten von ihnen – welche tatsächlich relevant sind, und welche nicht. Dieses Vorgehen entspricht dem Vorgehen beim Relevance Feedback. Das

Relevance Feedback

System nutzt diese Information um die Eigenschaften der idealen Antwort genauer zu beschreiben.

- Durch Wiederholung dieses Vorgangs wird erwartet, dass sich die Beschreibung der Eigenschaften an die tatsächlichen Eigenschaften der idealen Antwort annähert. *Iterationen*

Annahmen des BIR-Modell

Dem BIR-Modell liegen folgende Annahmen und Festlegungen zugrunde:

- Für eine Anfrage Q und ein Dokument D aus der Dokumentenkollektion versucht das probabilistische Modell die Wahrscheinlichkeit abzuschätzen, dass D im Hinblick auf Q für relevant erachtet wird.
- Das Modell unterstellt, dass diese Wahrscheinlichkeit nur von der Anfrage- und der Dokumentrepräsentation abhängig ist.
- Ferner nimmt das Modell an, dass es eine ideale Antwort auf die Anfrage Q gibt, die alle aus der Sicht des Anfragenden für Q relevanten Dokumente enthält.
- Diese ideale Antwort bezeichnen wir als $R^+(Q)$.

Für eine Anfrage Q ordnet das BIR-Modell jedem Dokument D als Retrieval-Status-Wert (RSV) eine auf Basis der Repräsentationen berechnete Näherung des Quotienten *Retrieval-Status-Wert*

$$\frac{P(D \in R^+(Q))}{P(D \notin R^+(Q))} \tag{7.3}$$

zu. Dieser RSV nähert die *Chance* an, dass Dokument D relevant für die Anfrage Q ist. *Chance*

Die Chance ist dabei für eine Wahrscheinlichkeit von 0 ebenfalls 0. Für eine Wahrscheinlichkeit von 1 ist die Chance unendlich. Abbildung 7.1 zeigt den Verlauf der Chance in Abhängigkeit von der Wahrscheinlichkeit.

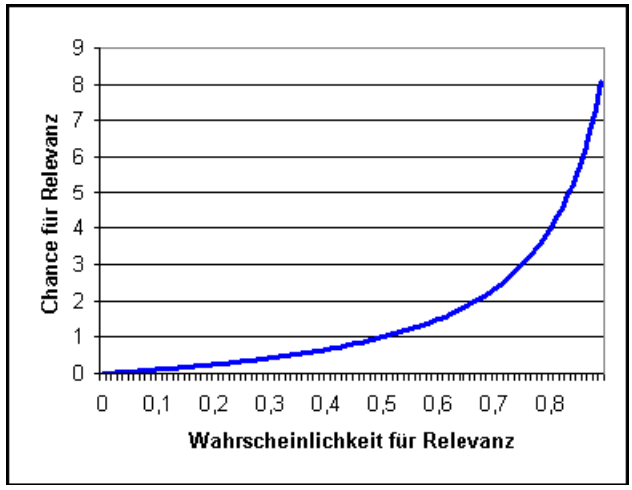


Abbildung 7.1 — Chance in Abhängigkeit von der Wahrscheinlichkeit

Definitionen

Zur genaueren Betrachtung des BIR-Modells benötigen wir noch einige Definitionen: Die Gewichte in den Beschreibungsvektoren zu den Dokumenten sind im BIR-Modell binär. Es gilt also $\beta(D) = (w_{d1}, w_{d2}, ..., w_{dt})$ mit $w_{dk} \in \{0, 1\}$ für alle $k \in \{1, ..., t\}$.

Ferner gehen wir davon aus, dass es auch zur Anfrage Q einen binären Beschreibungsvektor $\beta(Q) = (w_{q1}, w_{q2}, ..., w_{qt})$ mit $w_{qk} \in \{0, 1\}$ für alle $k \in \{1, ..., t\}$ gibt. Dieser Beschreibungsvektor für die Anfrage wird für den initialen Schritt benötigt.

Wir legen zudem fest, dass $R^-(Q)$ die Menge aller nicht relevanten Dokumente und damit das Komplement zu $R^+(Q)$ ist. $R^-(Q)$ vereinigt mit $R^+(Q)$ ist damit die Gesamtheit aller Dokumente.

Schließlich steht wie bereits eingeführt $P(D' \in R^+(Q)|\beta(D') = \beta(D))$ für die Wahrscheinlichkeit, dass ein Dokument D', dessen Repräsentation $\beta(D')$ der Repräsentation $\beta(D)$ von Dokument D entspricht, relevant ist.

Ähnlichkeit eines Dokumentes zur Anfrage

Wir können damit den Retrieval Status Wert für eine Anfrage Q und ein Dokument D als inhaltliche Ähnlichkeit zwischen Q und D wie folgt

angeben:

$$sim(D,Q) = \frac{P(D' \in R^+(Q) \mid \beta(D') = \beta(D))}{P(D' \in R^-(Q) \mid \beta(D') = \beta(D))} \tag{7.4}$$

Es handelt sich hier um die Chance, dass D' relevant ist sofern seine Repräsentation mit der von D übereinstimmt. Anders ausgedrückt ist dies die Chance, dass ein Dokument relevant ist, sofern es die gegebene Repräsentation $\beta(D)$ hat.

Dies können wir mit Hilfe des Theorems von Bayes vereinfachen.

Das Theorem von Bayes

Zunächst wollen wir die bedingte Wahrscheinlichkeit dafür bestimmen, dass a gilt, wenn auch b gilt – also $P(a|b)$. Wenn wir die Wahrscheinlichkeit $P(b)$ für das Eintreten von b und die Wahrscheinlichkeit $P(a \wedge b)$ für das gleichzeitige Eintreten von a und b kennen, dann gilt $P(a|b) = \frac{P(a \wedge b)}{P(b)}$.

Nehmen wir z.B. an, $P(b)$ sei 10% und $P(a \wedge b)$ sei 5%. Dann ist $P(a|b)$ offensichtlich $\frac{0{,}05}{0{,}1} = 0,5$ und damit 50%.

Wenn aber $P(a|b) = \frac{P(a \wedge b)}{P(b)}$ gilt, dann gilt natürlich auch $P(b|a) = \frac{P(a \wedge b)}{P(a)}$ durch einfachen Austausch der Bezeichner a und b. Dies lässt sich durch Multiplikation beider Seiten mit $P(a)$ umformen zu $P(a \wedge b) = P(a) \cdot P(b|a)$.

Setzen wir nun in $P(a|b) = \frac{P(a \wedge b)}{P(b)}$ für $P(a \wedge b)$ den Term $P(a) \cdot P(b|a)$ ein, so erhalten wir $P(a|b) = \frac{P(a) \cdot P(b|a)}{P(b)}$.

Dies kann wie folgt zusammengefasst werden:

$$
\begin{aligned}
& P(a|b) = \frac{P(a \wedge b)}{P(b)} && (1)\\
\Leftrightarrow \quad & P(b|a) = \frac{P(a \wedge b)}{P(a)} &&\\
\Rightarrow \quad & P(a \wedge b) = P(a) \cdot P(b|a) && (2)\\
\Rightarrow \quad & P(a|b) = \frac{P(a) \cdot P(b|a)}{P(b)} && (2) \text{ in } (1)
\end{aligned}
$$

Wenn wir das Theorem von Bayes nun auf Zähler und Nenner unserer obigen Formel für $sim(D, Q)$ anwenden, erhalten wir:

$$
\begin{aligned}
sim(D,Q) &= \frac{P(D' \in R^+(Q)) \cdot P(\beta(D') = \beta(D) \mid D' \in R^+(Q))}{P(\beta(D') = \beta(D))} \\
&\quad \cdot \frac{P(\beta(D') = \beta(D))}{P(D' \in R^-(Q)) \cdot P(\beta(D') = \beta(D) \mid D' \in R^-(Q))} \\
&= \frac{P(D' \in R^+(Q)) \cdot P(\beta(D') = \beta(D) \mid D' \in R^+(Q))}{P(D' \in R^-(Q)) \cdot P(\beta(D') = \beta(D) \mid D' \in R^-(Q))} \qquad (7.5)
\end{aligned}
$$

Von der ersten Gleichung zur zweiten Gleichung werden dabei lediglich die beiden Brüche zusammengezogen und im Zähler und Nenner $P(\beta(D') = \beta(D))$ gekürzt.

Wir müssen uns nun vor Augen führen, dass $P(D' \in R^+(Q))$ die Wahrscheinlichkeit dafür ist, dass ein beliebiges Dokument relevant ist. Dies entspricht aber gerade der Anzahl der bezüglich Q relevanten Dokumente dividiert durch die Gesamtzahl der Dokumente. Dieser Quotient ist aber für eine gegebene Anfrage Q über alle Dokumente hinweg konstant. Wenn aber $P(D' \in R^+(Q))$ ohnehin konstant ist, dann kann man diesen Term bei der Berechnung von $sim(D, Q)$ vernachlässigen, weil er als Konstante die Rangordnung der Dokumente bzgl. Q nicht beeinflusst.

Eine analoge Überlegung gilt für $P(D' \in R^-(Q))$, die Wahrscheinlichkeit dafür, dass ein beliebiges Dokument nicht relevant ist. Auch diese Wahrscheinlichkeit ist für eine gegebene Anfrage über alle Dokumente hinweg konstant und kann daher bei der Berechnung von $sim(D, Q)$ vernachlässigt werden.

Wir erhalten damit durch Weglassen des konstanten Faktors

$$\frac{P(D' \in R^+(Q))}{P(D' \in R^-(Q))}$$

ein neues vereinfachtes Ähnlichkeitsmaß:

$$sim(D,Q) \quad \approx \quad \frac{P(\beta(D') = \beta(D) \mid D' \in R^+(Q))}{P(\beta(D') = \beta(D) \mid D' \in R^-(Q))} \tag{7.6}$$

Der Zähler entspricht dabei der Wahrscheinlichkeit, dass ein Dokument die gegebene Repräsentation hat, wenn es relevant ist, und der Nenner entspricht der Wahrscheinlichkeit, dass ein Dokument die gegebene Repräsentation hat, wenn es nicht relevant ist.

Beispiel

Wir greifen hier das Beispiel mit den Dokumenten D_1 bis D_9 wieder auf.

Dokument	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9
w_{d1}	1	1	1	0	0	0	1	1	0
w_{d2}	1	0	1	0	1	0	1	0	0
Relevanz bzgl. Q	+	+	+	-	-	-	-	-	-

Betrachten wir nun ein neues Dokument D_{11} mit $\beta(D_{11}) = (1,1)$, so erhalten wir $P(\beta(D') = \beta(D_{11}) | D' \in R^+(Q)) = \frac{2}{3}$, weil zwei der drei relevanten Dokumente diese Repräsentation haben, und $P(\beta(D') = \beta(D_{11}) | D' \in R^-(Q)) = \frac{1}{6}$, weil nur eines der sechs nicht relevanten Dokumente diese Repräsentation hat. Damit ergibt sich $sim(D_{11}, Q) = 4$.

Das Problem mit der obigen Formel für $sim(D,Q)$ ist nun, dass wir die Wahrscheinlichkeiten $P(\beta(D') = \beta(D) | D' \in R^+(Q))$ und $P(\beta(D') = \beta(D) | D' \in R^-(Q))$ für das obige Beispiel anhand einer Stichprobe von neun Dokumenten ausrechnen können. Dies liegt daran, dass es für $t = 2$ nur $2^2 = 4$ mögliche Repräsentationen gibt. Schon für $t = 30$ gibt es aber über eine Milliarde mögliche Beschreibungsvektoren. Dies führt dazu, dass es keine sinnvolle Möglichkeit gibt, die benötigten Wahrscheinlichkeiten für praktisch relevante Werte von t empirisch zu bestimmen. Der entscheidende Schritt ist nun, die Wahrscheinlichkeiten nicht mehr über die ganzen Beschreibungsvektoren zu bestimmen, sondern je Komponente. Wir erhalten damit zunächst die folgende zur bisherigen Definition von $sim(D,Q)$ äquivalente Formulierung:

$$sim(D,Q) \approx \frac{P\left(\forall_{\{i|w_{di}=1\}}(w_{d'i}=1) \wedge \forall_{\{i|w_{di}=0\}}(w_{d'i}=0)|D' \in R^+(Q)|\right)}{P\left(\forall_{\{i|w_{di}=1\}}(w_{d'i}=1) \wedge \forall_{\{i|w_{di}=0\}}(w_{d'i}=0)|D' \in R^-(Q)|\right)}$$

Wir haben damit die Bedingung $\beta(D') = \beta(D)$ durch eine UND-Verknüpfung über die Gleichheit auf den einzelnen Komponenten ersetzt. Dabei werden zunächst jeweils alle Dimensionen i betrachtet, für die $w_{d'i} = 1$ ist und dann (mit UND verknüpft) alle Dimensionen i mit $w_{d'i} = 0$.

Wenn a und b unabhängige Ereignisse sind, dann kann man $P(a \wedge b)$ durch $P(a) \cdot P(b)$ berechnen. Wenn wir nun annehmen, dass für $i \neq j$ die Wahrscheinlichkeiten $P(w_{di} = 1|D' \in R^+(Q))$ und $P(w_{dj} = 1|D' \in R^+(Q))$ unabhängig voneinander sind, dann können wir $P(\forall_{\{i|w_{d'i}=1\}}(w_{d'i} = 1)|D' \in R^+(Q))$ durch $\prod_{\{i|w_{d'i}=1\}} P(w_{d'i} = 1|D' \in R^+(Q))$ berechnen.

Wenden wir dies analog auch auf die anderen Teile der obigen Formel an, so erhalten wir:

$$\begin{aligned} sim(D,Q) \approx \quad & \frac{\left(\prod_{\{i|w_{di}=1\}} P(w_{d'i}=1 \mid D' \in R^+(Q))\right)}{\left(\prod_{\{i|w_{di}=1\}} P(w_{d'i}=1 \mid D' \in R^-(Q))\right)} \\ & \cdot \frac{\left(\prod_{\{i|w_{di}=0\}} P(w_{d'i}=0 \mid D' \in R^+(Q))\right)}{\left(\prod_{\{i|w_{di}=0\}} P(w_{d'i}=0 \mid D' \in R^-(Q))\right)} \end{aligned} \tag{7.7}$$

Diese Vereinfachung, die auf der Annahme der Unabhängigkeit der einzelnen Terme basiert, hat dem Verfahren auch den mittleren Teil seines Namens als Binary Independence Modell gegeben. Ohne diese Annahme wäre das Verfahren (wegen der fehlenden empirischen Basis aufgrund der vielen verschiedenen Beschreibungsvektoren) nicht praktisch anwendbar.

Beispiel

Für die obigen Beispieldokumente D_1 bis D_9 erhalten wir:

- $P(w_{d'1} = 1|D' \in R^+(Q)) = 1$, da bei allen relevanten Dokumenten $w_{d1} = 1$ ist.

- $P(w_{d'2} = 1|D' \in R^+(Q)) = 2/3$, da bei allen relevanten Dokumenten w_{d2} 2-mal 1 und 1-mal 0 ist.
- $P(w_{d'1} = 0|D' \in R^+(Q)) = 0$, usw.
- $P(w_{d'2} = 0|D' \in R^+(Q)) = 1/3$
- $P(w_{d'1} = 1|D' \in R^-(Q)) = 1/3$
- $P(w_{d'2} = 1|D' \in R^-(Q)) = 1/3$
- $P(w_{d'1} = 0|D' \in R^-(Q)) = 2/3$
- $P(w_{d'2} = 0|D' \in R^-(Q)) = 2/3$

Damit ergibt sich für D_{11} mit $\beta(D_{11}) = (1, 1)$:

$$sim(D_{11}, Q) = \frac{1 \cdot \frac{2}{3}}{\frac{1}{3} \cdot \frac{1}{3}} = 6$$

Man sieht also bereits an diesem einfachen Beispiel, dass sich durch die nicht erfüllte Annahme der Unabhängigkeit ein veränderter Wert ergibt (ohne Unabhängigkeitsannahme hatten wir zuvor ja den Wert 4 erhalten).

Um die Produktbildung über alle Terme zu vermeiden nutzt man nun zwei Eigenschaften aus:

1. Der Logarithmus über ein Produkt ist gleich der Summe über die Logarithmen der einzelnen Terme. Genauer gilt $\log(a_1 \cdot a_2 \cdot \ldots \cdot a_x) = \log(a_1) + \log(a_2) + ... + \log(a_x)$.
2. Ferner bleibt die Ordnung zwischen Zahlen durch deren Logarithmierung erhalten. Wenn also $a_1 > a_2 > \ldots > a_x$ gilt, dann gilt auch $\log(a_1) > \log(a_2) > \ldots > \log(a_x)$.

Da wir im Hinblick auf $sim(D, Q)$ nicht an den absoluten Werten sondern nur an der Ordnung interessiert sind, können wir die Berechnung weiter vereinfachen,

- indem wir von diesem Ausdruck den Logarithmus nehmen,

- beachten, dass $P(w_{d'i} = 1|D' \in R^+(Q)) + P(w_{d'i} = 0|D' \in R^+(Q)) = 1$ gilt
- und Faktoren weglassen, die für alle Dokumente im Hinblick auf die gleiche Anfrage gleich sind oder als gleich angenommen werden.

Wir erhalten damit:

$$sim(D,Q) \approx \sum_{\{i|w_{qi}=w_{di}=1\}} \Big(\log \frac{P(w_{d'i} = 1 \mid D' \in R^+(Q))}{1 - P(w_{d'i} = 1 \mid D' \in R^+(Q))} + \log \frac{1 - P(w_{d'i} = 1 \mid D' \in R^-(Q))}{P(w_{d'i} = 1 \mid D' \in R^-(Q))} \Big) \quad (7.8)$$

$sim(D,Q)$ hängt nun also nur noch von den Termen ab, die sowohl in der Anfrage als auch im Dokument vorkommen. Im Folgenden werden wir betrachten, wie diese Formel angewendet werden kann.

Bestimmung der initialen Ähnlichkeit

Initial müssen wir ohne Vorabinformationen eine grobe Schätzung für $P(w_{d'i} = 1|D' \in R^+(Q))$ und $P(w_{d'i} = 1|D' \in R^-(Q))$ machen. Vorgeschlagen wurde hierzu:

$P(w_{d'i} = 1|D' \in R^+(Q))$ sei für alle Terme i mit $w_{qi} = 1$ gleich und zwar bspw. 0,5. Wir setzen also für die betrachteten Terme i mit $w_{qi} = 1$:

$$P(w_{d'i} = 1|D' \in R^+(Q)) = 0,5$$

Die Verteilung der Indexterme über den nicht relevanten Dokumenten entspreche der Verteilung über allen Dokumenten. Wir setzen hier also:

$$P(w_{d'i} = 1|D' \in R^-(Q)) = \frac{n_i}{N}$$

Wir erhalten somit:

$$sim(D,Q) \approx \sum_{\{i|w_{qi}=w_{di}=1\}} \left(\log \frac{0,5}{0,5} + \log \frac{1-\frac{n_i}{N}}{\frac{n_i}{N}} \right)$$
$$= \sum_{\{i|w_{qi}=w_{di}=1\}} \log \frac{N-n_i}{n_i}$$

Wenn wir unterstellen, dass N sehr viel größer als die Werte für n_i ist, vereinfacht sich dies zu $\sum_{\{i|w_{qi}=w_{di}=1\}} \log(\frac{N}{n_i})$. Dies entspricht der Summe über die inversen Dokumenthäufigkeiten *idf* aus dem Vektorraummodell. Betrachtet werden dabei nur die Terme, die sowohl im Dokument als auch in der Anfrage vorkommen.

Beispiel

Nehmen wir ein Indexierungsvokabular mit $t = 8$ Termen an. Für den Anfragevektor gilt $w_{q1} = 1$, $w_{q4} = 1$ und $w_{q5} = 1$. Alle anderen w_{qi} seien gleich null. N sei 500, n_1 sei 87, n_4 sei 23 und n_5 sei 100. Dann erhalten wir für ein Dokument mit dem Beschreibungsvektor $(1;0;0;0;1;1;0;0)$ einen Wert $sim(D,Q)$ von $\log(\frac{500-87}{87}) + \log(\frac{500-100}{100}) \approx 1,28$.

Anpassung des initialen Ergebnisses

Haben wir mit der Ähnlichkeitsfunktion aus dem vorangegangenen Abschnitt ein erstes Ergebnis ermittelt, so können wir mit oder ohne Unterstützung durch den Anfragenden unser Ergebnis fortentwickeln.

Anpassung des Ergebnisses ohne Unterstützung des Anfragenden

Sehen wir uns zunächst das Szenario ohne Unterstützung durch den Anfragenden an. Dieses Verfahren bezeichnet man auch als *Pseudo-Relevance Feedback*.

Pseudo-Relevance Feedback

Die initiale Formel hat eine Ergebnismenge ergeben, deren r am höchsten gerankte Elemente – höchste Werte für $sim(D,Q)$ – wir als Menge V bezeichnen wollen.

Wenn wir diese r-elementige Menge als repräsentative Stichprobe von $R^+(Q)$ auffassen, dann erhalten wir:

$$P(w_{d'i} = 1 \mid D' \in R^+(Q)) = \frac{|\{D' \in V \mid w_{d'i} = 1\}|}{|V|} \tag{7.9}$$

Wir teilen also den Anteil der Dokumente in V, für die $w_{d'i} = 1$ gilt, durch die Gesamtzahl der Dokumente in V (wobei per Definition gilt $|V| = r$). Analog können wir auch die Wahrscheinlichkeit $P(w = 1|D' \in R^-(Q))$ bestimmen. Dabei müssen wir im Zähler die Anzahl der Dokumente aus V, die eine 1 in der i-ten Komponente enthalten, von der Gesamtzahl der Dokumente abziehen, die eine 1 in der i-ten Komponente enthalten. Im Nenner müssen wir von der Gesamtzahl der Dokumente die Anzahl der Dokumente in V abziehen:

$$P(w_{d'i} = 1 \mid D' \in R^-(Q)) = \frac{n_i - |\{D' \in V \mid w_{d'i} = 1\}|}{N - |V|} \tag{7.10}$$

Mit diesen Formeln können wir das Ergebnis jetzt über mehrere Iterationen hinweg verbessern, ohne dass der Benutzer hierzu eine Relevanzbeurteilung treffen müsste. Die Annahme, dass V eine repräsentative Stichprobe aus $R^+(Q)$ ist, ist dabei offensichtlich kritisch. Wenn man aber den Benutzer nicht mit einer Relevanzbeurteilung belasten will, ist dies die einzig mögliche Vorgehensweise.

Anpassung des Ergebnisses mit Unterstützung des Anfragenden

Typischerweise verwendet man aber nicht V, sondern eine von dem Anfragenden ausgewählte Menge von relevanten Dokumenten zur Verbesserung des Ergebnisses. Es wird also analog dem Relevance Feedback vorgegangen.

Relevance Feedback

Dazu sucht der Anfragende aus der initialen Ergebnismenge die r am höchsten gerankten Dokumente aus, die er für relevant hält.

Wir können die Menge dieser Dokumente z.B. als V^* bezeichnen. In den obigen Formeln verwenden wir dann V^* statt V. Dadurch lassen sich natürlich deutlich bessere Ergebnisse erzielen als beim Pseudo Relevance Feedback.

Varianten

Da die obigen Formeln bei kleinen Mengen V, bzw. V^*, Probleme bereiten, verwendet man häufig auch

$$P(w_{d'i} = 1 \mid D' \in R^+(Q)) = \frac{|\{D' \in V \mid w_{d'i} = 1\}| + 0,5}{|V| + 1} \quad (7.11)$$

$$P(w_{d'i} = 1 \mid D' \in R^-(Q)) = \frac{n_i - |\{D' \in V \mid w_{d'i} = 1\}| + 0,5}{N - |V| + 1} \quad (7.12)$$

oder:

$$P(w_{d'i} = 1 \mid D' \in R^+(Q)) = \frac{|\{D' \in V \mid w_{d'i} = 1\}| + \frac{n_i}{N}}{|V| + 1} \quad (7.13)$$

$$P(w_{d'i} = 1 \mid D' \in R^-(Q)) = \frac{n_i - |\{D' \in V \mid w_{d'i} = 1\}| + \frac{n_i}{N}}{N - |V| + 1} \quad (7.14)$$

In beiden Varianten werden im Zähler und Nenner Konstanten addiert, die verhindern, dass bei sehr kleinen oder leeren Mengen Probleme in der Berechnung auftreten können.

Zusammenfassung BIR-Modell

Vom Vorgehen her kann man das BIR-Modell am besten mit dem Relevance Feedback vergleichen. Deshalb wurden in der bereits im Abschnitt Relevance Feedback erwähnten Untersuchung von Salton und Buckley [SB90] unter anderem die Formeln mit den für das Relevance Feedback vorgeschlagenen Formeln verglichen. Ausgangspunkt war dabei allerdings jeweils ein initiales Ergebnis, das mit dem Vektorraummodell bestimmt wurde. Die Tabelle 7.1 stellt die Ergebnisse der Formel Ide (dec hi) und des BIR-Modells mit den obigen Formeln für die iterative Verbesserung gegenüber.

eingesetzte Methode		CACM 1033 Dok. 30 Anfr.	CISI 12684 Dok. 84 Anfr.	CRAN 1397 Dok. 225 Anfr.	INSPEC 1460 Dok. 112 Anfr.	MED 3204 Dok. 64 Anfr.	Durchschnitt
initiale Anfrage							
	Precision	0,1459	0,1184	0,1156	0,1368	0,3346	
IDE (dec hi)							
mit allen Termen	Precision	0,2704	0,1742	0,3011	0,2140	0,6305	
	Verbesserung	+86%	+47%	+160%	+56%	+88%	+87%
ausgewählte Terme	Precision	0,2479	0,1924	0,2498	0,1976	0,6218	
	Verbesserung	+70%	+63%	+116%	+44%	+86%	+76%
BIR-Modell							
mit allen Termen	Precision	0,2289	0,1436	0,3108	0,1621	0,5972	
	Verbesserung	+57%	+21%	+169%	+19%	+78%	+69%
ausgewählte Terme	Precision	0,2224	0,1634	0,2120	0,1876	0,5643	
	Verbesserung	+52%	+38%	+83%	+37%	+69%	+56%

Tabelle 7.1 — Experimentelle Ergebnisse zum Relevance Feedback und zum BIR-Modell

Aus Tabelle 7.1 ergibt sich, dass die probabilistische Formel zumindest nicht besser abschneidet als Relevance Feedback nach der Formel Ide (dec hi).

Insgesamt ergeben sich für das BIR-Modell folgende Vor- und Nachteile:

Vorteile

Der Vorteil des probablilistischen Modells besteht in der Theorie darin, dass die Dokumente absteigend nach ihrer Relevanzwahrscheinlichkeit sortiert werden. Das Modell hat somit eine probabilistische Fundierung und klar definierte Annahmen.

Eine Implementierung ist ohne Probleme bspw. mit invertierten Listen möglich.

Nachteile

Einen Nachteil stellt die Notwendigkeit dar, ein initiales Ergebnis aufgrund sehr grober Annahmen zu berechnen. Hierzu kann allerdings auch

– wie bei den oben angegebenen Experimenten – das Vektorraummodell genutzt werden.

Experimentelle Ergebnisse weisen für das BIR-Modell allerdings eher schlechtere Werte auf als für das Vektorraummodell mit Relevance Feedback.

Ein weiteres Problem stellt die Verwendung binärer Werte in den Beschreibungsvektoren dar, die keine feine Abstufung bzw. Gewichtung der einzelnen Terme erlauben.

Zudem stellt die Unabhängigkeitsannahme bezüglich der Indexterme eine unrealistische vereinfachende Annahme dar.

Weitere Ansätze zum probabilistischen Information Retrieval

Ein wesentliches Problem des BIR-Modells – und auch des Relevance Feedback – ist, dass sich die Verbesserung des Ergebnisses durch das Feedback immer nur auf die aktuelle Anfrage bezieht und auswirkt. Bei der Bearbeitung der nächsten Anfrage beginnt man wieder von vorne.

Anders ausgedrückt »lernt« das System im Rahmen des iterativen Prozesses der Anfragebearbeitung. Das Gelernte bezieht sich aber nur auf die aktuelle Anfrage und ist nicht auf andere Anfragen übertragbar.

Um dieses Problem zu lösen haben Fuhr und Buckley in [FB91] vorgeschlagen, Dokumenten-Retrieval als probabilistisches Lernen zu verstehen. Ihr Ansatz basiert auf drei Aspekten:

probabilistisches Lernen

1. Zunächst wird von spezifischen Anfragen, Termen und Dokumenten abstrahiert. Durch den Rückkopplungsprozess mit dem Anfragenden sollen hier allgemeinere Zusammenhänge »gelernt« werden als dass bestimmte Terme im Hinblick auf eine bestimmte Anfrage wichtig sind. Das Gelernte soll so über alle Anfragen und alle Dokumente hinweg hilfreich sein. Beim BIR-Modell wird dagegen nur im Hinblick auf die aktuelle Anfrage gelernt. Nach Ende des Anfrageprozesses wird diese »Lernmenge« verworfen.

2. Den zweiten Aspekt stellt die Flexibilität der Repräsentation dar. Beim BIR-Modell ist die Repräsentation in Form von Vektoren mit binären Werten festgelegt. Das probabilistische Lernen soll dagegen auf fast beliebige Repräsentationen anwendbar sein.

3. Den dritten Aspekt bildet schließlich die Nutzung des probabilistischen Lernens zur Gewinnung der Indexierungsgewichte.

Die Indexierung besteht dabei aus zwei Schritten: einem *description step* zur Beschreibung eines Paares aus Anfrage und Dokument und einem *decision step* zur Entscheidung über die Relevanz des Dokumentes im Hinblick auf die Anfrage.

Wir können hier aus Zeitgründen auf dieses und weitere Modelle aus dem Gebiet des probabilistischen Information Retrieval nicht näher eingehen. In [BYRN99] findet der interessierte Leser aber einige Beispiele solcher Modelle und zahlreiche Verweise auf weiterführende Literatur.

7.2 Okapi BM 25

BM 25 *BM 25* (vgl. [RWHB98, FTZ04]) ist der Name einer Ranking-Funktion, die von *S.E. Robertson*, *K. Spärck Jones* und anderen entwickelt wurde. Als Bestandteil des von der City Universität London entwickelten Okapi *Okapi* IR-Systems, wurde *BM 25* bei der Lösung verschiedener TREC-Tasks (s. Abschnitt 2.3.1) über mehrere Jahre hinweg sehr erfolgreich eingesetzt. Grundlage von *BM 25* ist das Probabilistische IR-Modell (s. Abschnitt 7.1). Der Relevanzwert für ein Dokument $\mathcal{D}$ bezüglich einer Anfrage $\mathcal{Q}$ wird mit folgender Formel berechnet[1]:

$$score(\mathcal{Q},\mathcal{D}) = \sum_{i=1}^{t} \ln \frac{N - n_i + 0.5}{n_i + 0.5} \cdot \frac{(k_1+1)tf_{di}}{k_1((1-b) + b\frac{dl}{avdl}) + tf_{di}} \cdot \frac{(k_3+1)tf_{qi}}{k_3 + tf_{qi}}$$

- k_1 (zwischen $1,0$ und $2,0$), b (normalerweise $0,75$) und k_3 (zwischen 0 und 1000) sind Konstanten
- dl entspricht der Länge des Dokumentes (Anzahl der Worte)
- $avdl$ entspricht der durchschnittlichen Dokumentenlänge in der Kollektion

[1] Wie in den Originalartikeln verwenden wir in diesen Formeln den natürlichen Logarithmus. Wie bereits auf Seite 49 angemerkt, könnte man ebenso den dekadischen Logarithmus (zur Basis 10) verwenden.

- Die anderen Parameter entsprechen der im Rahmen des Vektorraummodells (s. Abschnitt 5) eingeführten Notation.

IDF

Die IDF-Komponente $\ln \frac{N-n_i+0.5}{n_i+0.5}$ unterscheidet sich geringfügig von der IDF-Komponente des Vektorraummodells. Falls $n_i > \frac{N}{2}$ ist, kann diese Komponente auch negativ werden. Terme, die in mehr als der Hälfte aller Dokumente vorkommen, sind jedoch in der Regel Stoppwörter oder sie werden aufgrund der geringen Trennschärfe bei der Anfragebearbeitung ignoriert. Im typischen Bereich zwischen 1 und $\frac{N}{10}$ verlaufen $\ln \frac{N-n_i+0.5}{n_i+0.5}$ und $log\frac{N}{n_i}$ dagegen sehr ähnlich, da sich das $-n_i$ im Zähler hier ebenso wie die Addition von 0.5 in Zähler und Nenner kaum auswirken.

Einfluss von tf_{di}

Die Komponente $\frac{(k_1+1)tf_{di}}{k_1((1-b)+b\frac{dl}{avdl})+tf_{di}}$ der Formel *Okapi BM 25* berücksichtigt den Einfluss der Vorkommenshäufigkeit der Begriffe in den Dokumenten. Wie Abbildung 7.2 zeigt wird dabei die Erkenntnis beachtet, dass eine vollständige Dokumentlängennormierung nicht sinnvoll ist (siehe hierzu Abschnitt 5.3). In der Darstellung gehen wir von einer durchschnittlichen Dokumentlänge $avdl$ von 100 aus. Ferner setzen wir $tf_{di} = \frac{dl}{20}$, so dass der fragliche Begriff mit einer Vorkommenshäufigkeit von 5% im Dokument auftritt. Während bei einer vollständigen Dokumentlängennormierung alle Kurven deckungsgleich wären, ergeben sich in Abbildung 7.2 für längere Dokumente höhere Werte. Dieser Effekt entspricht exakt den Überlegungen aus Abschnitt 5.3 und er kann über den Parameter k_1 gesteuert werden (X-Achse in der Abbildung).

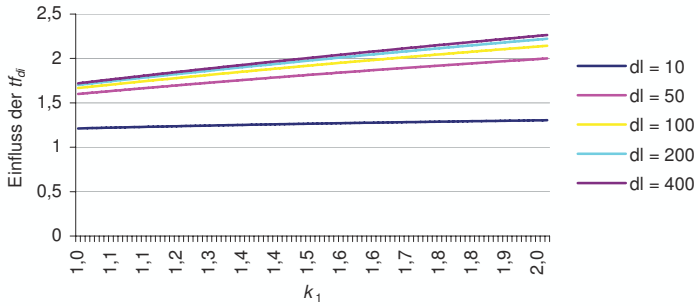


Abbildung 7.2 — Wert von $\frac{(k_1+1)tf_{di}}{k_1((1-b)+b\frac{dl}{avdl})+tf_{di}}$ unter der Annahme, dass $tf_{di} = \frac{dl}{20}$ und $avdl = 100$

Einfluss von tf_{qi}

Abbildung 7.3 zeigt wie über den Parameter k_3 der Einfluss der Vorkommenshäufigkeit von Begriffen in der Anfrage dosiert werden kann. Bei einem sehr kleinen Wert für k_3 spielt tf_{qi} keine Rolle während bei hohen Werten für k_3 der Wert von tf_{qi} praktisch linear in $score(\mathcal{Q}, \mathcal{D})$ eingeht. Welcher Wert konkret für k_3 zu wählen ist, hängt von den Charakteristika der zu betrachtenden Anfragen ab.

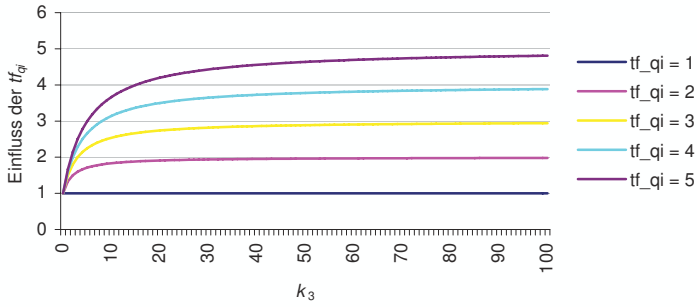


Abbildung 7.3 — Wert von $\frac{(k_3+1)tf_{qi}}{k_3+tf_{qi}}$ in Abhängigkeit von k_3

Insgesamt ist die Formel *Okapi BM 25* ein sehr erfolgreiches Beispiel für die gezielte Kombination der Ergebnisse langer Forschungsarbeit im Bereich des Vektorraummodells.

7.3 Latent Semantic Indexing

7.3.1 Einführung

Latent Semantic Indexing

Beim *Latent Semantic Indexing* werden die Beschreibungsvektoren für Dokumente und Anfragen in einen Raum geringerer Dimensionalität transformiert, dessen Dimensionen nicht den Indextermen, sondern abstrakteren »Konzepten« entsprechen. Auf diese Weise soll der Tatsache Rechnung getragen werden, dass die Ideen in einem Text stärker mit den im Text beschriebenen »Konzepten« als mit den verwendeten Termen zusammenhängen. Dazu wird die Theorie der Singularwertzerlegung (engl. *singular value decomposition*) eingesetzt.

Singularwertzerlegung

In den vorigen Kapiteln haben wir Information Retrieval mit invertierten Listen kennen gelernt. Mit diesem Ansatz kommt man schon sehr weit,

und zwar überraschend weit, wenn man die Vereinfachungen berücksichtigt, die gemacht werden müssen. Unter anderem bedeutet die Annahme der statistischen Unabhängigkeit der Anfrageterme untereinander (die ja auch im BIR-Modell genutzt wird) folgendes:

- Es wird nicht berücksichtigt, dass verschiedene Worte den gleichen Sinn haben können (Synonymie).
- Es wird nicht berücksichtigt, mit welchen Wörtern ein gegebenes Wort zusammen auftritt.

Typischerweise wird uns bei einer Suchanfrage nicht interessieren, ob der Autor eines Dokuments über Autos oder Personenkraftwagen oder gar PKWs schreibt. Eigentlich würden wir erwarten, oder zumindest hoffen, dass ein IR-System erkennen würde, dass die Wörter »PKW«, »Personenkraftwagen« und »Auto« den gleichen Sinn haben.

Dies zu erreichen, ist genau das Ziel von LSI, dem *Latent Semantic Indexing*. Kurz zusammengefasst versucht Latent Semantic Indexing Wörter, die häufig gemeinsam auftreten, miteinander in Beziehung zu bringen. Diese Beziehungen zwischen Wörtern *repräsentieren* dann zwar nicht den Sinn (die Semantik) des Textes, Teile der Semantik des Textes sind jedoch in den Wortbeziehungen *verborgen* (*latent*). Daher also der Name, der übersetzt soviel wie *»Indexierung der latenten Semantik«* bedeutet.

7.3.2 Ein Beispiel zur Motivation

Wie im Vektorraummodell betrachten wir auch hier jedes Dokument als einen Spaltenvektor einer Dokument×Term-Matrix. Jede Zeile repräsentiert ein Wort, jeder Eintrag in einer Zeile repräsentiert die Häufigkeit des Wortes in dem gegebenen Dokument, oder ein anderes sinnvolles Maß für die Wichtigkeit des Wortes im Text. Als Beispiel geben wir hier die Dokument×Term-Matrix an, die auch schon von den Erfindern des LSI in ihrem Artikel verwendet wurde [DDF+90].

Begriff	Dokument 1	Dokument 2	Dokument 3
Access	1	0	0
Document	1	0	0
Retrieval	1	0	1
Information	0	1	1
Theory	0	1	0
Database	1	0	0
Indexing	1	0	0
Computer	0	1	1

Wenn wir uns nun eine Anfrage nach *»IDF in computer-based information look-up«* ansehen, dann wird der Vorteil einer Technik klar, die den Sinn von Worten mit einbezieht. Übersetzt heißt die Anfrage: *»IDF in computer-gestützter Suche nach Information«*. Vergleichen wir nun die drei Dokumente unserer Datensammlung mit der Anfrage.

1. Das erste Dokument in unserer drei Dokumente umfassenden Datensammlung enthält die Terme »Access« (Zugriff), »Document«, »Retrieval«, »Database« und »Indexing«, würde also nach menschlichem Ermessen zur Anfrage passen. Jedoch stimmt kein einziger Term mit unserer Anfrage überein. Ein Retrieval-System, das nach dem Vektorraum-Modell arbeitet, würde dieses Dokument als irrelevant für die Anfrage ansehen.

2. Das zweite Dokument enthält die Anfrageterme »Computer« und »Information«, sowie den Term »Theory«.

3. Das dritte Dokument enthält ebenfalls die Anfrageterme »Computer« und »Information«, sowie den Term »Retrieval«. Die Dokumente zwei und drei würden von einem Vektorraum-basierten Retrieval-System daher als gleich relevant für die Anfrage erkannt.

Das vom Vektorraum-Modell erstellte Ranking steht in diesem Fall in krassem Widerspruch zu der Reihenfolge, die ein Mensch den Dokumenten geben würde: Jeder Term in Dokument 1 hat entweder mit dem Anfrageterm »look-up« (Auffinden) oder mit dem Anfrageterm »information« zu tun. Wir würden also dazu neigen, Dokument 1 als das relevanteste anzusehen. Ferner würden wir Dokument 3 für relevanter als Dokument 2 halten, da der Term »Retrieval« mehr mit dem Sinn der Anfrage zu tun hat als der eher unspezifische Term »Theory«.

7.3.3 Die Idee des Latent Semantic Indexing

LSI verwendet Konzepte aus der Linearen Algebra um die Dokument×Term-Matrix zu verkleinern, mit dem Ziel, dass in den Spalten der Dokument×Term-Matrix nicht mehr die Zuordnung von Dokumenten zu einem Wort, sondern zu »Begriffen« oder »Konzepten« steht.

Zu diesem Zweck modifiziert LSI die Dokument×Term-Matrix. Dabei werden wir im Folgenden die Anzahl der Terme, die wir bisher mit t bezeichnet haben, mit τ bezeichnen und die Anzahl der Dokumente in der Kollektion, die wir bisher mit N bezeichnet haben werden wir als δ ansprechen. Diese Namenskonventionen treffen wir hier, um – wie in der Literatur zu LSI üblich – später Matrizen mit T, S und D bezeichnen zu können.

Dokument×Term-Matrix

Wir erhalten damit die folgende Dokument×Term-Matrix X als $\tau \times \delta$-Matrix:

$$X = \begin{pmatrix} x_{11} & \cdots & x_{1\delta} \\ \vdots & \ddots & \vdots \\ x_{\tau 1} & \cdots & x_{\tau\delta} \end{pmatrix} \tag{7.15}$$

In dieser Matrix X ordnen die x_{ij}, aus denen X besteht, jedem Dokument (Spalte) verschiedene Terme (Zeilen) zu. Der Spaltenvektor $x_{i=1\ldots\tau,j}$ zeigt uns also, welche Terme das Dokument j enthält. Der Zeilenvektor $x_{i,j=1\ldots\delta}$ enthält die Information, welche Dokumente Term i enthalten.

Cosinusmaß

Eine mögliche Methode zum Vergleich zweier Dokumente ist das Cosinusmaß. In unserer Matrix entspricht das Cosinusmaß bis auf Normierungsfaktoren dem Produkt zweier Spaltenvektoren:

$$sim(k,l) = \sum_{i=1}^{\tau} \frac{x_{il} \cdot x_{ik}}{\sqrt{\sum_{j=1}^{\tau} x_{jl}^2} \cdot \sqrt{\sum_{j=1}^{\tau} x_{jk}^2}} \tag{7.16}$$

$sim(k,l)$ berechnet die Ähnlichkeit zwischen den Dokumenten in den Spalten k und l. Wir können damit eine Matrix definieren, die als Koeffizient k,l die Entfernung zwischen den Dokumenten k und l enthält. Diese $\delta \times \delta$-Matrix lässt sich, sofern wir davon ausgehen, dass die Spal-

$\delta \times \delta$-Matrix

tenvektoren, die die einzelnen Dokumente darstellen, normiert sind, sehr kurz schreiben als:

$$SIM = X^T X$$

(Wobei X^T die so genannte Transponierte von X mit $x_{ij}^T = x_{ji}$ bezeichnet.)

Beispiel

Betrachten wir als Beispiel die folgende nicht normierte Dokument×Term-Matrix:

$$X_u = \begin{pmatrix} 2 & 2 & 1 \\ 1 & 1 & 2 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

Eine Normierung der Spalten (und damit eine Dokumentlängennormierung im Sinne des Vektorraummodells) führt zu:

$$X = \begin{pmatrix} 0.756 & 0.816 & 0.447 \\ 0.378 & 0.408 & 0.894 \\ 0.378 & 0.408 & 0 \\ 0.378 & 0 & 0 \end{pmatrix}$$

Damit erhalten wir als Matrix SIM der Ähnlichkeit zwischen den 3 Dokumenten:

$$
\begin{aligned}
SIM &= X^T X \\
&= \begin{pmatrix} 0.756 & 0.378 & 0.378 & 0.378 \\ 0.816 & 0.408 & 0.408 & 0 \\ 0.447 & 0.894 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0.756 & 0.816 & 0.447 \\ 0.378 & 0.408 & 0.894 \\ 0.378 & 0.408 & 0 \\ 0.378 & 0 & 0 \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0.926 & 0.676 \\ 0.926 & 1 & 0.730 \\ 0.676 & 0.730 & 1 \end{pmatrix}
\end{aligned}
$$

Wir ersehen hieraus, dass z.B. die Ähnlichkeit zwischen dem ersten Dokument mit dem Beschreibungsvektor $(2, 1, 1, 1)$ und dem zweiten Dokument mit dem Beschreibungsvektor $(2, 1, 1, 0)$ den Wert 0.926 hat.

Wie im obigen Beispiel gezeigt, kann die in der Dokument×Term-Matrix X enthaltene Information gut zum Vergleich von Dokumenten (also zu deren Ähnlichkeitsberechnung) benutzt werden. Wie jedoch weiter oben schon ausgeführt wurde, sind die mittels des Cosinusmaßes erzielten Ergebnisse nicht unbedingt befriedigend. In gewisser Weise könnte man sagen, dass das Vektorraummodell *zu exakt* arbeitet: Schon nach einer kleinen Änderung kann ein Text vollkommen anders indexiert sein. LSI nähert daher die Dokument×Term-Matrix an, um *weniger exakt* und somit *weniger empfindlich* gegen Änderungen zu sein.

Das Näherungsverfahren, das hier verwendet wird, ist die *Singular Value Decomposition*, *SVD*. Hierbei wird die Dokument×Term-Matrix in drei Matrizen zerlegt:

$$
\begin{aligned}
X &= T_0 \cdot S_0 \cdot D_0^T \\
&= \begin{pmatrix} t_{11} & \cdots & t_{1m} \\ \vdots & \ddots & \vdots \\ t_{\tau 1} & \cdots & t_{\tau m} \end{pmatrix} \cdot \begin{pmatrix} s_{11} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & s_{mm} \end{pmatrix} \cdot \begin{pmatrix} d_{11} & \cdots & d_{1\delta} \\ \vdots & \ddots & \vdots \\ d_{m1} & \cdots & d_{m\delta} \end{pmatrix}
\end{aligned} \tag{7.17}
$$

Die wichtigste Eigenschaft in diesem Zusammenhang ist, dass S_0 *diagonal* (d.h. nur die Elemente auf der Diagonalen sind besetzt) und *quadratisch* ist, sowie dass sie eine $m \times m$-Matrix ist. Sie ist die Matrix einer *linearen Abbildung* und hat *Rang* m.

7.3.4 Anwendung der SVD

Wir können nun mittels der Singularwertzerlegung eine Zerlegung der Dokument×Term-Matrix X bestimmen. Dazu betrachten wir unser bereits eingeführtes Beispiel (Implementierungen der Singularwertzerlegung finden sich z.B. in Mathematikpaketen wie $\mathbf{R}$[2]):

$$
\begin{aligned}
X &= \begin{pmatrix} 0.756 & 0.816 & 0.447 \\ 0.378 & 0.408 & 0.894 \\ 0.378 & 0.408 & 0 \\ 0.378 & 0 & 0 \end{pmatrix} \\
&= T_0 \cdot S_0 \cdot D_0^T \\
&= \begin{pmatrix} -0.74 & 0.36 & -0.21 \\ -0.59 & -0.75 & 0.14 \\ -0.29 & 0.49 & -0.19 \\ -0.14 & 0.28 & 0.95 \end{pmatrix} \cdot \begin{pmatrix} 1.60 & 0 & 0 \\ 0 & 0.61 & 0 \\ 0 & 0 & 0.27 \end{pmatrix} \\
&\quad \cdot \begin{pmatrix} -0.59 & -0.60 & -0.54 \\ 0.46 & 0.30 & -0.84 \\ 0.67 & -0.74 & 0.10 \end{pmatrix}
\end{aligned}
$$

Entscheidend ist nun, dass die Werte in der Diagonalmatrix S_0 nach ihrer Größe sortiert sind und in gewisser Weise Konzepten entsprechen, die die Dokumente beschreiben.

Näherungen von X lassen sich nun durch das gezielte Entfernen der »unwichtigen Konzepte« bestimmen. So erhalten wir durch Weglassen des »unwichtigen Konzeptes« und der entsprechenden Spalte in T_0 und der entsprechenden Zeile in D_0^T

[2] http://www.r-project.org/ (letzter Abruf 7.1.2008)

$$
\begin{aligned}
X &= T_0 \cdot S_0 \cdot D_0^T \\
&= \begin{pmatrix} -0.74 & 0.36 \\ -0.59 & -0.75 \\ -0.29 & 0.49 \\ -0.14 & 0.28 \end{pmatrix} \cdot \begin{pmatrix} 1.60 & 0 \\ 0 & 0.61 \end{pmatrix} \cdot \begin{pmatrix} -0.59 & -0.60 & -0.54 \\ 0.46 & 0.30 & -0.84 \end{pmatrix}
\end{aligned}
$$

folgende vereinfachte Dokument×Term-Matrix X_1':

$$
\begin{aligned}
X_1' &= T_1 \cdot S_1 \cdot D_1^T \\
&= \begin{pmatrix} -0.74 & 0.36 \\ -0.59 & -0.75 \\ -0.29 & 0.49 \\ -0.14 & 0.28 \end{pmatrix} \cdot \begin{pmatrix} 1.60 & 0 \\ 0 & 0.61 \end{pmatrix} \cdot \begin{pmatrix} -0.59 & -0.60 & -0.54 \\ 0.46 & 0.30 & -0.84 \end{pmatrix} \\
&= \begin{pmatrix} 0.794 & 0.774 & 0.453 \\ 0.354 & 0.435 & 0.891 \\ 0.411 & 0.371 & 0.005 \\ 0.210 & 0.186 & -0.024 \end{pmatrix}
\end{aligned}
$$

Das Weglassen eines weiteren »Konzeptes« ergibt:

$$
\begin{aligned}
X_2' &= T_2 \cdot S_2 \cdot D_2^T \\
&= \begin{pmatrix} -0.74 \\ -0.59 \\ -0.29 \\ -0.14 \end{pmatrix} \cdot \begin{pmatrix} 1.60 \end{pmatrix} \cdot \begin{pmatrix} -0.59 & -0.60 & -0.54 \end{pmatrix} \\
&= \begin{pmatrix} 0.686 & 0.709 & 0.634 \\ 0.561 & 0.572 & 0.511 \\ 0.277 & 0.282 & 0.252 \\ 0.132 & 0.134 & 0.120 \end{pmatrix}
\end{aligned}
$$

Man sieht an diesem Beispiel wie sich die Dokument×Term-Matrix X durch das Entfernen einzelner Konzepte verändert. Die Matrizen X_i' werden schrittweise der ursprünglichen Dokument×Term-Matrix X unähnlicher. Die Stellen, die bei der ursprünglichen Dokument×Term-Matrix gleich null waren, werden jetzt mit Werten ungleich null besetzt, ein Zeichen, dass Korrelationen zwischen Termen in den Dokumenten erkannt worden sind. Dies kann man als Erkennung von Begriffen oder Konzepten interpretieren.

Im Hinblick auf die Bearbeitung von Anfragen besteht die Idee nun in zwei Punkten:

- Zunächst verwendet man zur Bearbeitung einer Anfrage ein weiteres »Pseudo-Dokument« in der Dokument×Term-Matrix X (also eine zusätzliche Spalte, die die Anfrageterme umfasst). Durch die Bestimmung von $SIM = X^T X$ kann dann die Ähnlichkeit des »Pseudo-Dokuments« zu allen »normalen« Dokumenten bestimmt werden.
- Bei der Bestimmung der Matrix $SIM = X^T X$ wird dabei nicht X sondern X_i' verwendet, wobei die Wahl des optimalen Wertes für i ein gewisses Problem darstellt.

7.4 Zusammenfassung

In diesem Kapitel haben wir LSI vorgestellt, eine Methode, die Konzepte aus der Linearen Algebra verwendet, um die Dokument×Term-Matrix so zu verändern, dass nicht Wörter, sondern die dahinter stehenden Begriffe zur Anfrage verwendet werden.

LSI stellt dabei zweifelsfrei ein wertvolles theoretisches Konzept dar. Seine praktische Relevanz ist aber noch Gegenstand der Forschung. Dabei sind effiziente Formen der Berechnung ebenso von Interesse wie die Wahl des am besten geeigneten i für X_i'.

8 Multimedia Information Retrieval

In den bisherigen Kapiteln haben wir uns primär mit der Suche nach Textdokumenten beschäftigt. In diesem Kapitel wollen wir unsere Überlegungen nun auf andere Medientypen ausweiten. Dabei geht es zunächst um die Suche nach Bildern, Videos oder Audiodateien. Darüber hinaus sind aber auch multimediale Dokumente zu betrachten, die sich aus einer Vielzahl einzelner Medienobjekte zusammensetzen.

8.1 Einleitung
8.2 Allgemeine Überlegungen
8.3 Bild-Retrieval
8.4 Audio Retrieval
8.5 Video Retrieval
8.6 Retrieval für Multimedia-Dokumente
8.7 Zusammenfassung

8.1 Einleitung

Multimedia

Der Begriff *Multimedia* ist in der Literatur vielfach definiert worden. Eine der häufig zitierten Definitionen stammt von Ralf Steinmetz [Ste00]:

> »Ein *Multimediasystem* ist durch die rechnergesteuerte, integrierte Erzeugung, Manipulation, Darstellung, Speicherung und Kommunikation von unabhängigen Informationen gekennzeichnet, die in mindestens einem kontinuierlichen (zeitabhängigen) und einem diskreten (zeitunabhängigen) Medium kodiert sind.«

Im Hinblick auf die Medien werden dabei kontinuierliche (zeitabhängige) und diskrete (zeitunabhängige) Medien unterschieden: Diskrete (zeitunabhängige) Medien hängen nicht von der Zeit ab. Bei diesen Medien ergibt sich keine Veränderung im Zeitablauf. Die wichtigsten Beispiele für solche Medien sind Texte und Bilder oder Graphiken. Dem stehen zeitabhängige (kontinuierliche) Medien gegenüber, bei denen die Information nicht nur in einzelnen Werten, sondern auch im Zeitpunkt ihres Auftretens steckt. Beispiele wären Audio, Bewegtbild/Video oder auch Temperatursensoren.

Wenn man die obige Definition von Steinmetz nun auf Multimedia-Dokumente überträgt, so würde die Definition fordern, dass ein solches Dokument mindestens ein kontinuierliches und ein diskretes Medium enthalten muss. Im alltäglichen Sprachgebrauch wird der Begriff Multimedia allerdings oft sehr viel breiter genutzt. Hier spricht man bereits bei einer einfachen Bilddatenbank oder bei der Verwaltung von Audiodateien von Multimediasystemen.

Für unsere Betrachtungen ist diese Unterscheidung insofern von Belang, als ein Retrieval-System für Bilder natürlich anders vorgehen muss, als ein Retrieval-System für Audios oder Videos oder ein Retrieval-System, das multimediale Dokumente im strengen Sinn von Steinmetz betrachtet.

Im Folgenden werden wir uns daher zunächst mit einzelnen Medien (Bildern, Audiodaten und Videos) beschäftigen, bevor wir uns zusammengesetzten (strukturierten) Multimedia-Dokumenten zuwenden.

8.2 Allgemeine Überlegungen

Bei der Suche nach Bildern, Audios oder Videos stellt sich die Frage, wie eine Anfrage für diese Medientypen definiert werden kann.

Betrachten wir dazu zunächst die Vorgehensweise, die wir für Textdokumente z.B. beim Vektorraummodell angewendet haben. Hier haben wir die Anfrage selbst auch als (Anfrage-)Text verarbeitet, der durch einen Beschreibungsvektor repräsentiert wurde. Gesucht wurden dann Textdokumente mit einem »ähnlichen« Beschreibungsvektor. Wenn wir dieses Vorgehen z.B. auf Bilder übertragen, dann benötigen wir ein Anfragebild, zu dem »ähnliche« Bilder gesucht werden. Dieses Anfragebild kann ein Beispielbild oder auch nur eine grobe Skizze sein.

Übertragen auf Audios könnte dies bedeuten, dass wir ein Beispielaudio oder eine gepfiffene Melodie vorgegeben haben, zu der eine passende Audiodatei gesucht wird.

Der Ablauf einer Anfragebearbeitung kann damit medientypunabhängig wie in Abbildung 8.1 dargestellt werden:

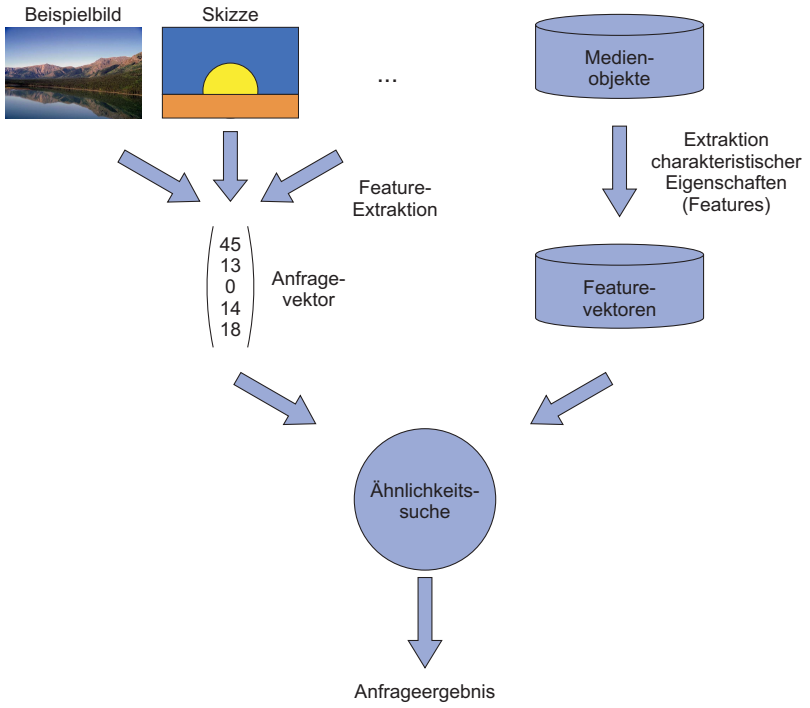


Abbildung 8.1 — Typische Architektur für ein Retrieval-System am Beispiel von Bilddaten

Sowohl für das Anfragebild, das als Beispielbild oder als Skizze gegeben sein kann, als auch für die Bilder im Datenbestand müssen dabei charakteristische Eigenschaften bestimmt und in einem Beschreibungsvektor zusammengefasst werden. Typischerweise greift man hierzu auf Farb- oder Textureigenschaften zurück, auf die wir in Abschnitt 8.3 noch zurückkommen werden. Auf den Beschreibungsvektoren kann dann eine Ähnlichkeitssuche durchgeführt werden.

Abbildung 8.2 verdeutlicht die Zusammenhänge bei der Verwaltung und

dem Retrieval von Multimediadaten. Dabei werden die Aufgabenbereiche *1. Einfügen in die Datenbank* und *2. Extraktion der Features* unabhängig von den Anfragen zum Zeitpunkt des Einfügens der Daten in die MM-Datenbank ausgeführt. Die Aufgabenbereiche *3. Aufbereitung der Anfrage*, *4. Ähnlichkeitsberechnung & Anfragebearbeitung* sowie *5. Ergebnisaufbereitung* werden dagegen für jede Anfrage einzeln bearbeitet.

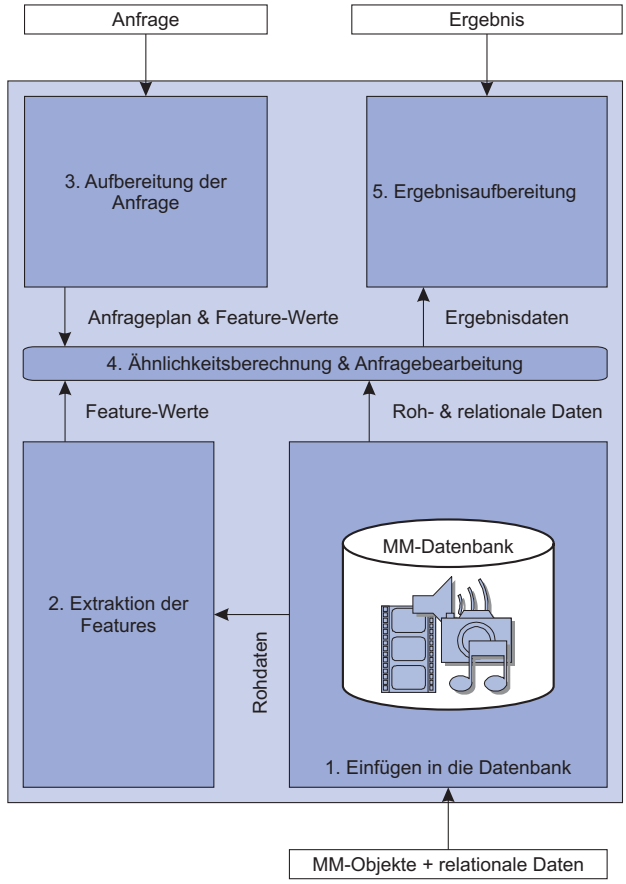


Abbildung 8.2 — Ablauf der Verwaltung und des Retrieval von Multimediadaten (nach [Sch02])

Wichtig ist zunächst für die einzelnen Medientypen geeignete »Feature-Mengen« zu finden. Daneben ist auch zu beachten, dass man nicht immer

an so genannten *»globalen Eigenschaften«* interessiert ist. Betrachten wir als Beispiel die Farbverteilung in einem Bild. Wenn wir nach einem Bild suchen, das ein bestimmtes Logo enthält, dann muss nicht die Farbverteilung des ganzen Bildes der Farbverteilung des Logos entsprechen. Vielmehr müssen wir uns beim Vergleich der Farbverteilungen auf einzelne Bildsegmente beziehen. Ähnliches gilt auch bei Videos oder Audios, wo einzelne Teilstücke für das Retrieval von Bedeutung sein können. Für die so gewonnenen Segmente können wir dann wieder entsprechende Verfahren der Feature-Extraktion anwenden.

Ein anderer Ansatz zum Retrieval könnte auch darin bestehen, ein in einem bestimmten Medientyp (z.B. als Bild) vorliegendes Medienobjekt in ein Objekt eines anderen Typs zu überführen, der unter Umständen geeigneter zum Retrieval ist. So könnte man ein Bild, in dem auch Texte zu sehen sind, mittels *Optical Character Recognition* (OCR) in ein Textdokument überführen.

Optical Character Recognition, OCR

Insgesamt sind damit die drei folgenden Arten von Operationen von entscheidender Bedeutung für die Suche nach Medienobjekten:

- **Feature-Extraktion**: Bestimmung charakteristischer Eigenschaften für ein Medienobjekt — *Feature-Extraktion*
- **Segmentierung**: Zerlegung eines Medienobjekts vom Typ t ($t \in Text, Bild, Audio, Video$) in mehrere Teilobjekte des gleichen Typs t — *Segmentierung*
- **Konvertierung**: Überführung eines Medienobjekts vom Typ t ($t \in Text, Bild, Audio, Video$) in ein Objekt eines anderen Typs $t' \neq t$ — *Konvertierung*

Tabelle 8.1 fasst einige konkrete Umsetzungen dieser Operationstypen für die verschiedenen Medientypen zusammen.

Im Folgenden wollen wir nun gezielt auf die Medientypen *Bild*, *Audio* und *Video* eingehen, bevor wir uns strukturierten Multimedia-Dokumenten zuwenden.

8.3 Bild-Retrieval

Die wichtigsten Kriterien im Hinblick auf die Suche nach den zu einem vorgegebenen Anfragebild ähnlichen Bildern sind Farbe und Textur. Daneben wird teilweise auch die Form als Kriterium herangezogen, was hier

	Feature-Extraktion	Segmentierung	Konvertierung
Textdaten	Statistiken zu Wortvorkommen, ...	Gliederung in Kapitel, Abschnitte, Paragraphen, Sätze, ...	als Bild drucken, vorlesen, ...
Bilddaten	Farbhistogramme, Textur, ...	regionenbasierte Segmentierung, kantenbasierte Segmentierung, ...	Optical Character Recognition (OCR), ...
Audiodaten	Sprechererkennung, Erkennung von Musikinstrumenten, Tonhöhen, ...	Erkennung von Pausen, ...	Spracherkennung, ...
Videodaten	Bewegungserkennung, ...	Szenenerkennung, ...	Extraktion von Schlüsselbildern, Erkennen von Untertiteln, ...

Tabelle 8.1 — Beispieloperationen für verschiedene Medientypen

aber nicht betrachtet werden soll (vgl. hierzu z.B. [VH01]). Wir werden uns im Folgenden auf die Kriterien Farbe und Textur konzentrieren und zusätzlich Verfahren zur Bildsegmentierung betrachten.

8.3.1 Farbe

Farbmodelle

Für die Verwaltung und Verarbeitung von Bildern sind sehr unterschiedliche Farbmodelle gebräuchlich. Hier sollen exemplarisch das RGB-Modell, das CMY-Modell, das HSV-Modell und das Munsell-Modell vorgestellt werden:

RGB-Modell

- **Das RGB-Modell**

  Beim RGB-Modell spricht man auch von einer additiven Farbmischung. Eine Farbe wird hier aus der Überlagerung der drei Grundfarben Rot, Grün und Blau in unterschiedlicher Intensität zusammengesetzt. Die drei Beispiele in Abbildung 8.3 sollen diese Farbmischung verdeutlichen. Dabei wird angenommen, dass wie üblich zur Codierung der Intensität jeder einzelnen Farbe ein Byte – und damit ein Wert zwischen 0 und 255 – verwendet wird.

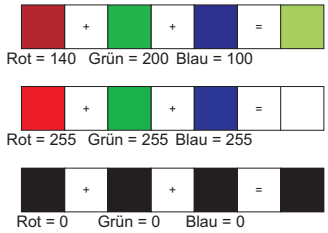


Abbildung 8.3 — Beispiele für die additive Farbmischung

Eine konkrete Farbe kann nun durch ein Tripel – z.B. (140, 200, 100) – repräsentiert werden. Eine Farbe entspricht damit einem Punkt in einem dreidimensionalen Würfel. Abbildung 8.4 verdeutlicht diese Sichtweise.

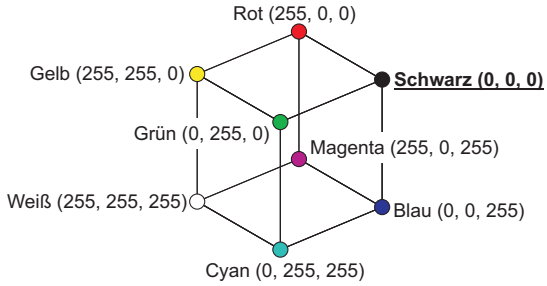


Abbildung 8.4 — Der RGB-Farbraum

Das additive Zusammensetzen der Farben entspricht dabei dem technischen Vorgehen in einem Röhren- oder CRT-Monitor (CRT für *Cathode Ray Tube*), also einem aktiv Licht erzeugenden Medium. Es wird daher typischerweise zur Ausgabe auf Displays verwendet.

- **Das CMY-Modell** *CMY-Modell*

  Das CMY-Farbmodell (CMY für Cyan, Magenta und Yellow) wird häufig für die Ausgabe auf Druckern verwendet. Man spricht dabei auch von einem subtraktiven Farbmodell, weil die spektralen Intensitäten der einzelnen Lichtkomponenten hier gemäß dem Farbwert aus dem weißen Licht entfernt werden. Abbildung 8.5 verdeutlicht dieses Modell.

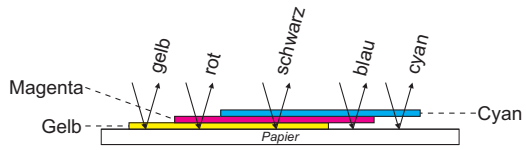


Abbildung 8.5 — Farben im subtraktiven CMY-Farbmodell

Die Tatsache, dass sich Schwarz dabei durch die Überlagerung von Cyan, Magenta und Gelb ergibt, würde z.B. bei einem Tintenstrahldrucker zu einem sehr hohen Verbrauch an Tinte führen. Man ergänzt das Modell daher häufig zum CMYK-Modell, in dem zusätzlich eine explizite Komponente für die Farbe Schwarz aufgenommen ist (Das »K« steht dabei für *blacK*, weil der Buchstabe »B« bereits für *Blue* verwendet wird.). Abbildung 8.6 zeigt den Farbwürfel für das CMY-Farbmodell.

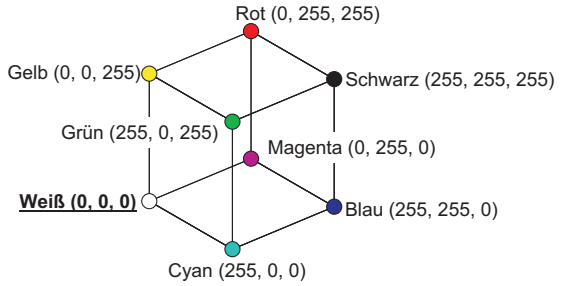


Abbildung 8.6 — Der CMY-Farbraum

HSV-Modell

- **Das HSV-Modell**

  Das HSV-Farbmodell beschreibt die Farben nicht durch die Mischung von Grundfarben, sondern durch die Farbparameter Farbton (*Hue*), Sättigung (*Saturation*) und Helligkeit (*Value*). Die Farbe wird durch einen Farbwinkel im HSV-Farbkreis angegeben und die Sättigung durch den Abstand vom Weißpunkt. Der HSV-Farbkreis entsteht dabei durch die Projektion des RGB-Farbwürfels entlang der Schwarz-Weiß-Achse in eine Ebene senkrecht zu dieser Achse. Abbildung 8.7 verdeutlicht diesen Farbraum.

  Ein Vorteil des HSV-Farbmodells gegenüber den eher technisch motivierten RGB- und CMY-Modellen ist, dass das HSV-Modell

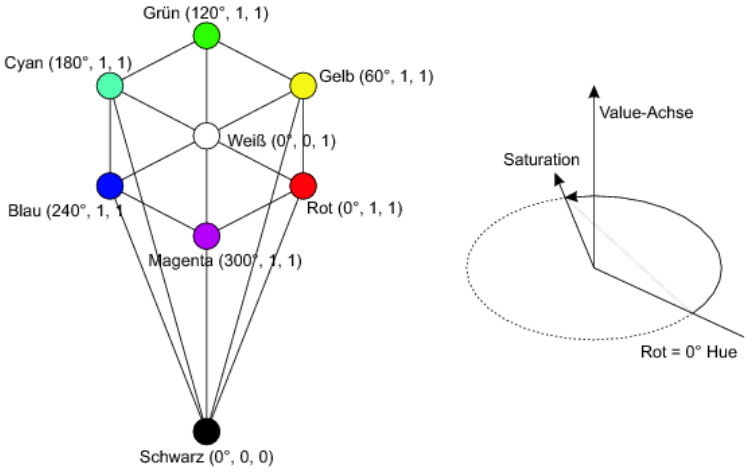


Abbildung 8.7 — Der HSV-Farbraum und die Bedeutung der Komponenten

die Farben wie das menschliche Auge erfasst. Die Farben werden durch den Farbton (Hue), die Sättigung (Saturation) und die Helligkeit (Value oder Brightness) beschrieben.

Der Farbton ist dabei das farbunterscheidende Merkmal, die Sättigung bezieht sich auf die Reinheit oder Intensität einer Farbe und die Helligkeit bezieht sich auf den prozentualen Anteil von Weiß in einer Farbe.

- **Das Munsell-Modell** *Munsell-Modell*

  Neben den technisch orientierten Farbmodellen RGB und CMY und dem an der menschlichen Wahrnehmung orientierten HSV-Modell existieren alternative Farbmodelle mit einem »künstlerisch« orientierten Hintergrund (vgl. hierzu z.B. auch [Hil98]). Der Modelliervorgang der Farben ist hier an das Mischen von Farben angelehnt: Ausgehend von einer Grundfarbe werden weiße Pigmente (Deckweiß) hinzugefügt, um die Grundfarbe aufzuhellen und schwarze Pigmente (Deckschwarz), um die Farbe abzudunkeln. Eines der ältesten dieser Farbmodelle, das auch kommerziell eingesetzt wird, ist von A.H. Munsell 1915 entwickelt worden. Es verwendet die Farbinformationen *Farbschattierung*, *Chromatik* und *Abstufung*. Munsell unterteilt die Farbschattierung in zehn Grund-

farben, die Chromatik in fünf Halbtöne und die Reflexion in zehn Stufen. Um die Reproduzierbarkeit zu sichern, wurden kleine Papiertafeln erstellt, die, entsprechend eingefärbt, dauerhaft die 500 verschiedenen Farben des Munsell-Systems darstellen.

Dieses Farbmodell von Munsell ist z.B. von Seaborn et al. [SHS99] verwendet worden, um eine Zuordnung der vom Menschen tatsächlich wahrgenommenen Farben zum HSV-Modell vorzunehmen. Abbildung 8.8 zeigt, welche Farben (angegeben durch ihre Hue- und Value-Koordinate) als Grün, Blau, Pink, ... wahrgenommen werden.

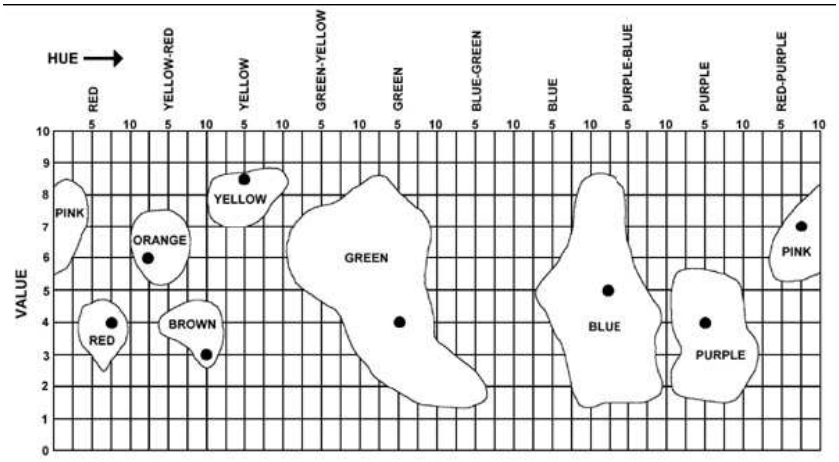


Abbildung 8.8 — Positionierung der vom Menschen wahrgenommenen Farben in einem HSV-Farbraum nach [SW95]

Im Hinblick auf die Suche nach Bildern stellt sich nun die Frage, wie die obigen Farbmodelle bei der Suche nach Bildern verwendet werden können, die farblich ähnlich zu einem Anfragebild sind. Hierzu wird ein Farbraum üblicherweise in Teilbereiche unterteilt. Für jeden dieser Teilbereiche wird dann gezählt, wie viele Pixel des betrachteten Bildes in diesen Teilbereich fallen. Wenn man diese Zahlen über alle Teilbereiche hinweg betrachtet, entsteht für das Bild ein so genanntes *Farbhistogramm*.

Farbhistogramm

Betrachten wir hierzu ein Beispiel auf Basis des RGB-Modells. Der Farbraum sei gemäß Abbildung 8.9 in 36 Teilbereiche gegliedert. Hierzu werden der Rot- und der Grünbereich jeweils in 3 und der Blaubereich

in 4 Abschnitte unterteilt. Die einzelnen Teilbereiche werden nach dem angedeuteten Schema von 0 bis 35 durchnummeriert.

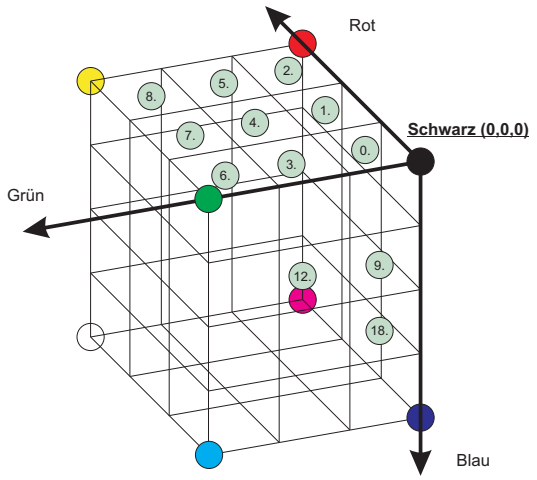


Abbildung 8.9 — Unterteilung des RGB-Farbraums in 36 Teilbereiche

Wenn wir den Farbwert eines Pixels mit (f_R, f_G, f_B) bezeichnen, dann können wir zunächst die Zuordnung des Pixels zu den Teilintervallen für die einzelnen Farben wie folgt bestimmen:

$$i_R = \begin{cases} 0 & \text{falls} \quad f_R \in \{0, \ldots, 85\} \\ 1 & \text{falls} \quad f_R \in \{86, \ldots, 170\} \\ 2 & \text{falls} \quad f_R \in \{171, \ldots, 255\} \end{cases}$$

$$i_G = \begin{cases} 0 & \text{falls} \quad f_G \in \{0, \ldots, 85\} \\ 1 & \text{falls} \quad f_G \in \{86, \ldots, 170\} \\ 2 & \text{falls} \quad f_G \in \{171, \ldots, 255\} \end{cases}$$

$$i_B = \begin{cases} 0 & \text{falls} \quad f_B \in \{0, \ldots, 63\} \\ 1 & \text{falls} \quad f_B \in \{64, \ldots, 127\} \\ 2 & \text{falls} \quad f_B \in \{128, \ldots, 191\} \\ 3 & \text{falls} \quad f_B \in \{192, \ldots, 255\} \end{cases}$$

Die Nummer des Teilbereichs, zu dem ein Pixel gehört, ergibt sich dann als $i_R + i_G \cdot 3 + i_B \cdot 3 \cdot 3$.

Wenn wir nun über das betrachtete Bild hinweg zu jedem der 36 Teilbereiche die Anzahl der Pixel ermitteln, die in diesen Teilbereich fallen, dann erhalten wir ein Histogramm über die Farbverteilung des Bildes, dessen Werte wir zur Eliminierung des Einflusses der Bildgröße noch durch die Gesamtzahl der Pixel teilen können. Abbildung 8.10 verdeutlicht ein solches Histogramm.

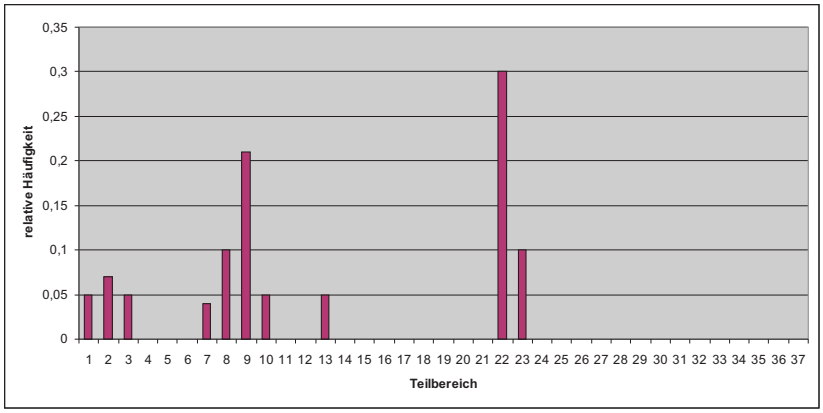


Abbildung 8.10 — Beispiel eines Farbhistogramms für ein Bild bei Unterteilung des Farbraums in 36 Teilbereiche

Um nun die Farbdistanz zwischen zwei Bildern zu berechnen, bezeichnen wir die einzelnen Teilbereiche des Farbraums mit tb_j und die Anzahl der Teilbereiche mit t. Wenn wir zusätzlich mit nh_x das normierte Histogramm des Bildes I_x bezeichnen, dann können wir die Farbdistanz zwischen einem Anfragebild I_q und einem Bild I_p wie folgt berechenen:

$$dist_{color}(I_q, I_p) = \sum_{j=1}^{t} \frac{|nh_q(tb_j) - nh_p(tb_j)|}{nh_q(tb_j) + nh_p(tb_j) + 1} \tag{8.1}$$

Dabei werden die Unterschiede in den einzelnen Histogrammsäulen mit der Gesamtzahl in den Säulen gewichtet.

Dieses Maß kann nun verwendet werden, um zu einem Anfragebild ähnliche Bilder – die mit den kleinsten Werten für $dist_{color}(I_q, I_p)$ – zu finden.

Welches Farbmodell bei der Betrachtung der Bildähnlichkeit zum Einsatz kommen sollte, ist in der Literatur stark umstritten. Relativ häufig

wird das HSV-Modell eingesetzt, weil es der menschlichen Farbwahrnehmung besser entspricht, als z.B. das RGB-Modell. Weil beim HSV-Modell der H-Wert für die Wahrnehmung besonders wichtig ist, wird dabei z.B. eine Aufteilung des Farbraums in Teilbereiche verwendet, die in der H-Achse 18 Teilintervalle und in den beiden anderen Dimensionen je 3 Teilintervalle betrachtet. Man erhält so insgesamt $18 \cdot 3 \cdot 3 = 162$ Teilbereiche und damit Farbbeschreibungen mit 162 Dimensionen.

In der Literatur gibt es nun eine Fülle von weitergehenden Vorschlägen zur Betrachtung der Farbähnlichkeit. So ist für die menschliche Wahrnehmung der Farbigkeit eines Bildes neben der Häufigkeit des Auftretens einer Farbe auch von Bedeutung, ob eine Farbe großflächig (in einer »kohärenten« Umgebung) oder verstreut auftritt. Zählt man in einem Bild lediglich, wie viele Pixel eine gewisse Farbe haben, so geht diese Information verloren. Deshalb wird in [PZM96] die Verwendung des so genannten Farbkohärenzvektors (*Color Coherence Vector*, CCV) vorgeschlagen. Zur Berechnung des CCV prüft man für jedes Pixel, ob es sich in einer hinreichend großen gleichfarbigen Umgebung befindet. Wenn ja, bezeichnet man es als kohärent, sonst als inkohärent. Man führt dann zwei getrennte Histogramme für die Zählung der kohärenten und der inkohärenten Pixel und berücksichtigt diese beiden Histogramme bei der Distanzberechnung.

Andere Vorschläge betreffen weitere Farbmodelle, die besonders gut für die Ähnlichkeitsbestimmung geeignet sein sollen, wie das L*a*b*-Farbmodell. Für eine genauere Darstellung dieser Ansätze und weitere Literaturhinweise sei an dieser Stelle auf [Gev01] und [Ste00] verwiesen.

8.3.2 Textur

Neben den Farbeigenschaften eines Bildes spielt für die menschliche Wahrnehmung auch die Textur eine wesentliche Rolle. Unter der Textur versteht man dabei eine kleinräumige Oberflächenstruktur, gleich ob natürlich oder künstlich, regelmäßig oder unregelmäßig. Beispiele für Texturen unterschiedlicher Charakteristik könnten eine Baumrinde, ein Strickmuster, eine Holzmaserung, die Oberfläche eines Schwamms, eine Tapete oder auch ein Teppich sein.

Für die Suche nach Bildern, die eine ähnliche Textur, wie ein gegebenes Beispielbild, aufweisen ist es nun wichtig, die Textur eines Bildes zu erfassen. Zu dieser so genannten *Texturanalyse* gibt es zwei grundlegende Ansätze:

Texturanalyse

- Die *strukturelle Analyse* sucht in einem Bild nach kleinen Grundbausteinen sowie einer Anordnungsregel zur Bildung der Textur. Sie ist für das Information Retrieval weniger geeignet.
- Die *statistische Texturanalyse* beschreibt die Textur als Ganzes anhand bestimmter Attribute wie z.B. der lokalen Grauwertvarianz, der Regelmäßigkeit, der Grobkörnigkeit, der Orientierung und des Kontrastes.

Für die Texturanalyse werden Farbbilder üblicherweise zunächst in Graustufenbilder konvertiert. Bei der Betrachtung der Graustufenbilder stellen sich dann aber die Fragen: Welche Strukturen möchte man als Textur bezeichnen? Wo im Bild befinden sich diese Strukturen?

Wie Texturen in einem Bild vorkommen, hängt dabei nämlich stark von der Skalierung ab. Stellt das Bild nur eine Aufnahme eines Teppichs dar, so wird man das Bild im Allgemeinen durch eine einzige Textur gut beschreiben können. Haben wir es aber mit einem Bild zu tun, dass z.B. einen Raum mit Personen, Möbeln, einem Parkettboden, Gardinen, ... zeigt, so treten in dem Bild zahlreiche unterschiedliche Texturen auf. In diesem Fall erscheint es sinnvoll, das Bild zunächst in Segmente einzuteilen und für diese Segmente einzeln die Textureigenschaften zu bestimmen. Wir werden uns im Abschnitt 8.3.3 noch ausführlich mit Fragen der Bildsegmentierung beschäftigen.

Das Kriterium für das Auftreten einer Textur in einer Bildregion ist dabei die signifikante, regelmäßige Variation der Grauwerte in einer Umgebung.

Das Vorgehen bei der Texturanalyse muss nun zunächst eine Entscheidung für ein Texturmaß beinhalten. Anschließend können im Bild Regionen bestimmt werden, die hinsichtlich des Maßes homogen sind und schließlich kann die Berechnung des Texturmaßes für jede Texturregion erfolgen.

Nach diesen Vorüberlegungen wollen wir nun beispielhaft ein verbreitetes statistisches Verfahren zur Texturanalyse betrachten.

Ein statistisches Verfahren zur Texturanalyse

Zunächst wird hier das Bild in ein Grauwertbild überführt. Hierzu kann man z.B. die Value-Komponente des HSV-Farbmodells nutzen.

Auf Basis der Grauwerte des Bildes wird dann für das Bild eine *Grauwertübergangsmatrix* bestimmt, die für jede Kombination von Grauwerten a und b angibt, wie oft die Grauwerte a und b in dem Bild in einer bestimmten Anordnung zueinander benachbart auftreten. Für die Anordnung $[a][b]$ (d.h. ein Pixel mit Grauwert b liegt unmittelbar rechts von einem Pixel mit Grauwert a) ergibt sich für das Bild, dessen Grauwerte auf der linken Seite der Abbildung 8.11 dargestellt sind, die auf der rechten Seite angegebene Grauwertübergangsmatrix. Dabei gehen wir vereinfachend von nur vier Grauwerten aus.

Grauwertübergangsmatrix

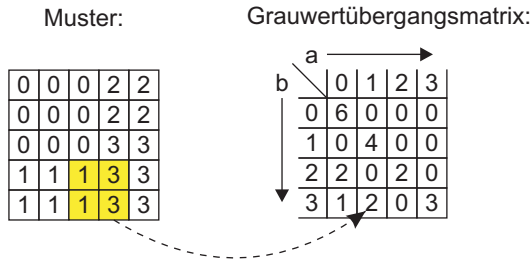


Abbildung 8.11 — Eine Grauwertübergangsmatrix

In der Abbildung sind beispielhaft die beiden Vorkommen des Grauwertübergangs [1][3] hervorgehoben. Da zwei solche Übergänge im Bild existieren, ist der entsprechende Eintrag in der Grauwertübergangsmatrix 2.

In der praktischen Anwendung müssten nun weitere Grauwertübergangsmatrizen für andere Nachbarschaftsanordnungen zweier Grauwerte betrachtet werden. So z.B. für übereinander stehende Grauwerte oder für diagonal benachbarte Grauwerte. Wir werden uns aber im Weiteren auf eine Grauwertübergangsmatrix beschränken.

Berechnung des Kontrasts

Der Kontrast bildet eine der wichtigsten Kenngrößen zur Charakterisierung einer Textur. Zu seiner Berechnung nehmen wir an, dass N Grauwerte unterschieden werden. Im obigen Beispiel gilt $N = 4$. In der Anwendung ist z.B. ein Wert von 256 typisch.

Kontrast

Ferner wollen wir mit $g(a, b)$ die Einträge der $N \times N$-Grauwert-Übergangsmatrix bezeichnen. Dann kann ein Maß K für den Kontrast

einer Textur wie folgt berechnet werden:

$$K = \sum_{a=0}^{N-1} \sum_{b=0}^{N-1} \underbrace{(a-b)^2}_{Differenz} \cdot \underbrace{g(a,b)}_{Häufigkeit} \tag{8.2}$$

Zum Verständnis der Formel ist es hilfreich zu bedenken, dass ein hoher Kontrast dazu führt, dass stark unterschiedliche Grauwerte dicht aneinander grenzen. Deshalb wird in der Formel $(a-b)^2$ als Maß für die Unterschiedlichkeit verwendet und mit der Häufigkeit multipliziert.

Auch hier gilt natürlich, dass in der Praxis bei der Berechnung des Kontrastes mehrere Nachbarschaftsanordnungen berücksichtigt werden müssen. Wir beschränken uns hier aber auf lediglich eine Nachbarschaftsanordnung.

Homogenität der Textur

Homogenität

Ergänzend zum Kontrast wird häufig die Homogenität einer Textur betrachtet. Dabei ist eine Textur umso homogener, je öfter die gleichen Grauwertübergänge wiederkehren. Viele unterschiedliche Grauwertübergänge stehen also für eine sehr inhomogene Textur, während das häufige Auftreten einiger weniger Grauwertübergänge eine sehr homogene Textur charakterisiert.

Die Homogenität H einer Textur kann daher wie folgt bestimmt werden:

$$H = \sum_{a=0}^{N-1} \sum_{b=0}^{N-1} g(a,b)^2 \tag{8.3}$$

Bei einer vorgegebenen Zahl von Pixeln erzielt ein Bild durch diese Berechnung offensichtlich die höchste Homogenität, wenn überall der gleiche Grauwertübergang auftritt. Treten dagegen alle Grauwertübergänge ungefähr mit der gleichen Häufigkeit – dies wäre dann $\frac{\text{Anzahl der Übergänge}}{N^2}$ – auf, so ergibt sich die geringste Homogenität.

Bestimmung der Texturähnlichkeit

Um nun die Texturähnlichkeit zwischen zwei Bildern zu bestimmen, könnten wir für beide Kontrast und Homogenität berechnen und die

beiden Bilder jeweils durch einen Vektor mit den beiden Komponenten *Kontrast* und *Homogenität* beschreiben. Auf die zweidimensionalen Vektoren könnten wir dann ein Ähnlichkeits- oder Distanzmaß wie z.B. den Euklidischen Abstand anwenden.

Die beschriebene Betrachtung von Kontrast und Homogenität ist dabei eine sehr einfache Herangehensweise. Leistungsfähigere Texturanalyseverfahren stützen sich u.a auf Verfahren wie die *Multiscale Simultaneous Autoregression*, *Markov Random Fields* oder *Tree-Structured Wavelet Transforms*. Der interessierte Leser sei hierzu auf [SL01] verwiesen.

8.3.3 Segmentierung

Bisher haben wir implizit so genannte »globale Bildeigenschaften« betrachtet, indem wir uns bei der Erstellung eines Farbhistogramms oder bei der Berechnung von Kontrast und Homogenität jeweils auf das gesamte Bild bezogen haben. Die Betrachtung globaler Bildeigenschaften hat einige Vorteile:

- Die entsprechenden Verfahren sind ohne vorherige Segmentierung anwendbar,
- sie erlauben eine vollständig automatische Durchführung von Anfragen und
- sie sind insbesondere bei abstrakten (und homogenen) Bildern nützlich.

Dem stehen aber einige gravierende Nachteile gegenüber:

- Verfahren, die Bilder als Ganzes betrachten, sind problematisch, wenn ein Bild verschiedene Teilobjekte (z.B. mit unterschiedlicher Textur) enthält.
- Ferner sind sie problematisch, wenn der Hintergrund eines Bildes recht dominant ist und dadurch sowohl die Farb- als auch die Textureigenschaften maßgeblich bestimmt.

Mögliche Verbesserungen können nun in zwei Richtungen zielen:

- Zunächst kann man versuchen, die Probleme dadurch abzumildern, dass man mehrere globale Bildeigenschaften bei der Ähnlichkeitsbestimmung berücksichtigt. So kann man die Bildähnlichkeit z.B. als gewichteten Durchschnitt aus der Farb- und der Texturähnlichkeit definieren. Dies ist beim Bildretrieval sehr gebräuchlich.

Segmentierung

- Eine weitere Möglichkeit ist die *Segmentierung* des betrachteten Bildes in Teilbilder. Farb- und Texturähnlichkeiten können dann auf Basis der gewonnenen Teilbilder ermittelt werden. So kann man z.B. nach Bildern suchen, die ein bestimmtes Logo enthalten, indem man Bilder sucht, die ein Teilbild enthalten, das dem gegebenen Logo ähnlich ist. Auch dabei kann man sich auf eine gewichtete Ähnlichkeit aus Farb- und Texturähnlichkeit beziehen.

Im Hinblick auf die Betrachtung von Teilbildern (oder Bildsegmenten) stellt sich nun die Frage, wie diese gewonnen werden können. Hierfür gibt es drei Möglichkeiten:

feste Aufteilung

- Zunächst kann man eine **feste Aufteilung** der Bilder in Teilbereiche vornehmen, um so z.B. bei Landschaftsbildern den Hintergrund vom primären Bildinhalt zu trennen.

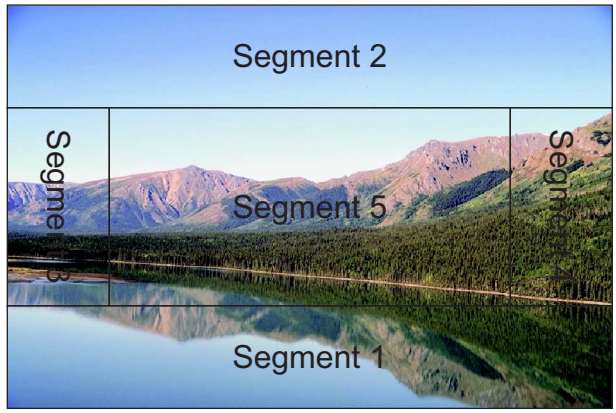


Abbildung 8.12 — Mögliche feste Unterteilung von Bildern in 5 Segmente

Dabei wird für alle Bilder die gleiche Aufteilung in Segmente verwendet. Wegen der starren Vorgehensweise ist dieses Verfahren am ehesten für relativ homogene Bildkollektionen geeignet. – z.B. für eine Sammlung von Landschaftsbildern.

- Eine andere Möglichkeit besteht darin, die **Segmentierung manuell** vorzunehmen. Der Benutzer zeichnet hier zunächst die Kontur des entsprechenden Objekts, das ein Segment bilden soll, und gibt ggf. noch manuell Zusatzinformationen an – z.B. den Namen der Person, die in dem Segment zu sehen ist. Der Vorteil dieses manuellen Vorgehens ist die mögliche hohe Qualität. Der Nachteil besteht aber im sehr hohen manuellen Aufwand.

manuelle Segmentierung

- Aus diesem Grund wurden zahlreiche Verfahren entwickelt, die eine **(teil-)automatische Segmentierung** vornehmen. Das System ermittelt hier durch Kantenerkennung oder auf Basis von Farb- und/oder Texturunterschieden zwischen Bildbereichen eine initiale Segmentierung. Die so gewonnene Segmentierung kann dann ggf. noch manuell verfeinert werden und auch hier kann ein Benutzer gebeten werden, zu den einzelnen Segmenten Metadaten (wie z.B. den Namen einer abgebildeten Person) anzugeben. Durch die (Teil-)Automatisierung verringert sich der manuelle Aufwand deutlich. Beim Verzicht auf eine manuelle Nachbearbeitung entfällt er ganz. Der Nachteil besteht aber in einer zum Teil zweifelhaften Qualität.

(teil-)automatische Segmentierung

Im den folgenden Abschnitten wollen wir uns nun den Verfahren der automatischen Bildsegmentierung zuwenden. Eine solche automatische Bildsegmentierung ist grob gesprochen eine Operation, die den einzelnen Pixeln eines Bildes je nach *Objekt- oder Segmentzugehörigkeit* pro Objekt bzw. Segment einen eindeutigen Identifikator zuweist. Den Pixeln des Hintergrundes wird dabei üblicherweise eine Null zugewiesen. Auf diese Weise kann dann für jedes Pixel des Bildes ermittelt werden, zu welchem Segment oder welchen Segmenten es gehört, wobei die Segmente idealerweise inhaltlich beschreibbaren Objekten entsprechen sollten.

Die Hauptaufgabe der Segmentierung besteht damit darin, inhaltlich zusammenhängende Pixelbereiche (Objekte) zu identifizieren. Die Erkennung der Objekte ist nicht Aufgabe der Segmentierung. So geht es bei der Segmentierung eines Bildes, auf dem Personen abgebildet sind, darum, die Bildbereiche einzelner Personen zu identifizieren. Die Erkennung der Personen ist nicht Gegenstand der Segmentierung.

Im Folgenden werden wir uns nun wieder auf Grauwertbilder beziehen. Bei einer Anwendung auf Farbbilder hat man die Möglichkeit entweder die einzelnen Komponenten der Farbdarstellung getrennt zu betrachten oder die Bilder in Grauwertbilder zu transformieren – z.B. durch Verwendung der H-Komponente bei HSV-Bildern.

Wir wollen nun drei Klassen von Segmentierungsverfahren betrachten:

Punktorientierte Segmentierungsverfahren

punktorientierte Segmentierungsverfahren

Bei punktorientierten Segmentierungsverfahren wird der Grauwert der isoliert betrachteten Pixel als Segmentierungskriterium verwendet. Hierzu wird die Grauwertverteilung eines Bildes in einem Histogramm erfasst, indem zu jedem der 255 Grauwerte die Anzahl der Pixel angetragen wird, die diesen Wert aufweisen. Es können dabei natürlich auch weniger oder mehr Grauwerte verwendet werden.

Auf Basis des Grauwert-Histogramms versucht man dann einen oder mehrere Schwellenwerte zu finden, die die häufig auftretenden Grauwertbereiche voneinander separieren. Dies funktioniert z.B. sehr gut, wenn im Bild ein einfarbiges Objekt dargestellt ist, dessen Helligkeit sich klar vom Hintergrund abhebt. Auch bei einigen wenigen Objekten mit unterschiedlichen Helligkeiten führt das Verfahren zu brauchbaren Ergebnissen. Man erhält dann ein so genanntes bi- oder multimodales Histogramm mit räumlich getrennten Maxima.

Abbildung 8.13 verdeutlicht dies, wobei 16 Grauwerte (von 0 bis 15) angenommen werden. Hier könnten z.B. die sehr schwachen Grauwerte (0 bis 3) für einen hellen Hintergrund und die beiden anderen Maxima (7 bis 11 sowie 14) für zwei farblich unterschiedliche Objekte stehen. Wir würden dann allen Pixeln im Bild mit einem Grauwert von 0, 1, 2 oder 3 den Segment-Identifikator Null für den Hintergrund zuweisen. Alle Pixel im Bild mit einem Grauwert zwischen 7 und 11 würden den Segment-Identifikator 1 erhalten, um ihre Zugehörigkeit zu diesem Segment zu verdeutlichen, und alle Pixel mit dem Grauwert 14 würden den Segment-Identifikator 2 erhalten.

Man kann das beschriebene einfache Verfahren nun natürlich verfeinern. So kann man eine punktorientierte Segmentierung auf Basis von Farbwerten z.B. als Segmentierung von Clustern im 3D-Raum des RGB- oder HSV-Farbmodells auffassen. Dies ändert aber nichts an den grundsätzlichen Nachteilen punktorientierter Segmentierungsverfahren:

- Zunächst treten klare bi- oder multimodale Verteilungen von Grauwerten in der Praxis eher selten auf. Vielmehr hat man es oft mit unimodalen oder sehr unstrukturierten Verteilungen zu tun.
- Oft zerstören auch Digitalisierungsfehler eine eigentlich bimodale Verteilung.

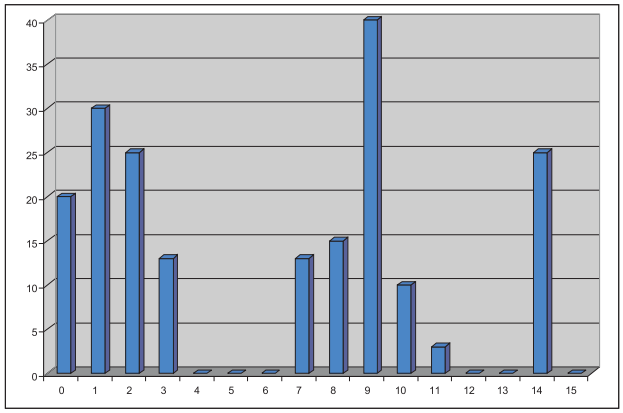


Abbildung 8.13 — Beispiel für ein Grauwert-Histogramm mit räumlich getrennten Maxima

- Wenn sich Schwerpunkte im Histogramm nicht klar voneinander abgrenzen lassen, werden die Punkte im Überlappungsbereich häufig falsch zugeordnet.
- Schließlich bleibt als wichtigster Kritikpunkt, dass die räumliche Lage der Pixel bei der Segmentierung nicht berücksichtigt wird. So kann es z.B. sein, das die Pixel mit dem Grauwert 14 im Bild sehr verstreut vorkommen. Sie werden trotzdem alle dem gleichen Segment zugeordnet.

Punktorientierte Verfahren setzen damit im Grunde eine manuelle Vorbearbeitung der Bilder voraus, in der einzelne Bereiche markiert werden, die dann unter Verwendung lokaler Histogramme segmentiert werden können. Hier können die einfach zu realisierenden punktorientierten Segmentierungsverfahren nützliche Dienste leisten.

Kantenorientierte Segmentierung

kantenorientierte Segmentierungsverfahren

Um einen etwas genaueren Blick auf die kantenorientierten Verfahren zur automatischen Segmentierung werfen zu können, müssen wir uns zunächst mit einem sehr grundlegenden Verfahren der Bildanalyse beschäftigen: mit der *Faltung eines Bildes mit einer Maske*.

Faltung eines Bildes

Man berechnet dazu aus einem Eingabebild E schrittweise ein zunächst »leer«-initialisiertes Ausgabebild A. Hierzu läßt man eine Faltungsmaske M zeilenweise über E laufen und verknüpft an jeder Position die Einträge der Maske mit den Grauwerten der darunter liegenden Bildpunkte.

Für eine genauere Betrachtung benötigen wir die folgenden Bezeichnungen:

- $e(x, y)$: der Grauwert des Punktes an der Stelle (x, y) im Eingabebild E.
- $a(x, y)$: der Eintrag im Ausgabebild A an der Stelle (x, y).
- m: die Größe der Maske M, die damit eine $m \times m$-Matrix ist; m muss immer ungerade sein.
- $m(u, v)$: der Eintrag in der Maske M an der Stelle (u, v) mit $u, v \in 0, \ldots, m-1$.

Damit ergibt sich die Berechnung des Ausgabebildes A mit $k = (m-1)/2$ zu:

$$a(x, y) = \sum_{u,v=0}^{m-1} e(x + k - u, y + k - v) \cdot m(u, v) \tag{8.4}$$

Randbereiche der Breite k in A bleiben dabei leer.

Betrachten wir hierzu ein Beispiel. Gegeben sei ein Eingabebild E mit den in Abbildung 8.14 dargestellten Grauwerten. Auf diese Eingabematrix wenden wir nun die Faltungsmatrix M an.

Im Prinzip wird bei der Berechnung eines $a(x, y)$-Wertes jeweils der Mittelpunkt der Faltungsmaske auf die Stelle (x, y) des Eingabebildes gelegt. Die dabei übereinander liegenden Werte in E und M werden miteinander multipliziert und anschließend aufsummiert.

Verfahren zur Kantenermittlung

Kantenermittlung

Das oben beschriebene Verfahren der Faltung kann nun zur Kantenermittlung verwendet werden. In einem Vorverarbeitungsschritt wird dazu das in Grauwertdarstellung vorliegende Bild geglättet: Der Grauwert jedes Pixels wird durch einen gewichteten Mittelwert der Grauwerte seiner

E:

6	1	3	8	7	8	...	...
6	2	5	7	7	8	...	...
6	1	4	8	9	9	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...

M:

1	0	-1
2	0	-2
1	0	-1

A:

						...	...
	7	-24	-13	-3	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...

Abbildung 8.14 — Beispiel für die Anwendung einer Faltungsmatrix

umgebenden Pixel ersetzt. Dadurch ergibt sich zwar ein gewisser Schärfeverlust aber auch ein weniger »verrauschtes« Bild, das sich besser zur Kantenermittlung eignet.

Wir können dann die folgenden Masken zur Ermittlung horizontaler und vertikaler Kanten verwenden, die auch als Sobel-Operatoren bezeichnet werden: *Sobel-Operatoren*

M_{horiz}:

1	0	-1
2	0	-2
1	0	-1

M_{vert}:

1	2	1
0	0	0
-1	-2	-1

Abbildung 8.15 — Masken zur Erkennung von Kanten in horizontaler und vertikaler Richtung

Dort, wo im jeweiligen Ausgabebild A ein vom Betrag her sehr hoher Wert auftritt, können wir eine vertikale bzw. horizontale Kante vermuten. So bedeutet ein betragsmäßig großer Eintrag im Ausgabebild A_{horiz}, das sich durch Anwendung von M_{horiz} ergibt, dass an der zugehörigen Stelle in E ein ausgeprägter Grauwertwechsel in links-rechts-Richtung vorliegt.

Konzeptionell hat man damit partielle Ableitungen von E in Spalten- und Zeilenrichtung durchgeführt. Um den Gradientenbetrag zu ermitteln, der die Gesamtstärke einer »schiefen« Kante an der Stelle (x, y)

angibt, bestimmt man nun

$$a_{grad}(x,y) = \sqrt{a_{horiz}(x,y)^2 + a_{vert}(x,y)^2} \tag{8.5}$$

und binärisiert das Ergebnis mit einem Schwellenwert e, um nur Punkte in das endgültige Ausgabebild aufzunehmen, an denen ein hinreichend starker Grauwertgradient ermittelt wurde. Hierzu werden im Ergebnis alle Einträge mit einem Betrag unter dem Schwellenwert e zu 0 und alle anderen zu 1 gesetzt.

Ergänzend kann nun in Folgeschritten eine Berechnung der Gradientenorientierung sowie eine Bestimmung von Mengen von Punkten auf der gleichen Kante vorgenommen werden.

Vorgehen kantenorientierter Segmentierungsverfahren

Kantenorientierte Segmentierungsverfahren arbeiten nun in zwei Schritten:

Kantenextraktion

Zunächst werden die Kanten eines Bildes extrahiert. Hierzu könnte man auf das oben vorgestellte Verfahren zurückgreifen. In der Praxis werden allerdings üblicherweise aufwändigere Verfahren verwendet, die prinzipiell aber ähnlich vorgehen. Ein Beispiel ist der so genannte Canny-Operator [Can86], der im Wesentlichen aus fünf Stadien besteht:

1. Das Bild wird mit einem zweidimensionalen Gaußfilter zur Rauschreduktion bearbeitet.
2. Das geglättete Bild wird mit gerichteten Ableitungsoperatoren überzogen.
3. Nicht-maximale Ableitungen werden in Richtung der maximalen Ableitungen überführt.
4. Angepasste Schwellenwerte werden genutzt, um »wahre Kanten« vom Rauschen zu trennen.
5. Eine Karte von konsistenten Kanten wird mit Hilfe einer »Merkmalssynthese« erzeugt.

Verbindung der Kanten

Im Anschluss daran werden dann die ermittelten Kanten derart verbunden, dass sie um die zu extrahierenden Objekte geschlossene Konturen bilden. Hierzu gibt es verschiedene Typen von Algorithmen:

- *Einfache Algorithmen*

  Diese Algorithmen arbeiten nach sehr einfachen Kriterien und verbinden z.B. jeweils Kantensegmente, die nicht weiter als eine Distanz d [in Pixeln] voneinander entfernt sind.

- *Komplexere Algorithmen*

  Aufwändigere Verfahren verbinden Kantensegmente, z.B. indem zusätzlich die Richtung der Kante in die Berechnung mit einbezogen wird.

- *Algorithmen auf Basis der Hough-Transformation*

  Die Hough-Transformation dient dazu, eine Regressionsgerade durch m vorgegebene Punkte zu legen. Dabei beschreibt die Hough-Transformation eine Linie im kartesischen Koordiantensystem durch einen Punkt aus Radius r und Winkel f im polaren Koordinatensystem. Eine genauere Betrachtung dieser Verfahren würde uns aber zu weit in das Feld der Bildverarbeitung führen. Deshalb sei an dieser Stelle auf die Literatur zur Bildverarbeitung verwiesen (z.B. [BK98]).

Regionenorientierte Segmentierungsverfahren

regionenorientierte Segmentierungsverfahren

Regionenorientierte Segmentierungsverfahren beruhen auf der nahe liegenden Beobachtung, dass geometrische Nähe bei der Objektsegmentierung eine wichtige Rolle spielen sollte. Benachbarte Pixel weisen meist ähnliche Eigenschaften auf. Dieser Aspekt wird aber von den punktorientierten und zum Teil auch von den kantenorientierten Segmentierungsverfahren nicht oder nicht hinreichend berücksichtigt.

Region Growing

Eine Alternative bildet nun das Verfahren des *Region Growing*. Hier startet man mit einem Saatpixel und zieht sukzessive die Nachbarn des Pixels hinzu, wenn sie zu diesem Saatpixel in einer Ähnlichkeitsbeziehung stehen. Die Ähnlichkeit kann dabei z.B. über die Standardabweichung der Grauwerte definiert werden.

Idealerweise sollte man zum Region Growing pro Region ein Saatpixel verwenden. Zur Automatisierung können hierfür z.B. lokale Extrema der Histogramme der Grauwertverteilung verwendet werden. Regionen, die »einander begegnen«, werden dann zusammengelegt, wenn sie homogen genug sind.

Water-Inflow

Als konkretes auf dem Prinzip des Region Growing basierendes Verfahren wollen wir nun die Segmentierung durch *Water-Inflow* betrachten.

Segmentierung durch Water-Inflow

Die Segmentierung durch Water-Inflow ist eine Kombination lokaler und globaler Ansätze. Die grundsätzliche Idee besteht darin, ein Grauwertbild schrittweise mit Wasser zu füllen. Die Grauwerte der Pixel werden dabei als Höhe betrachtet. Je höher das Wasser in den Iterationen des Algorithmus steigt, desto mehr Pixel werden unter Wasser verschwinden.

In jedem der Schritte existieren Regionen von Land und Wasser. Die Landregionen entsprechen hierbei den Objekten oder Segmenten.

Wir wollen hierzu ein Beispiel aus [Ste00] betrachten. Gegeben sei dabei Abbildung 8.16.

Abbildung 8.16 — Oiginalbild für Water-Inflow

Für dieses Bild ergibt sich ein Segmentierungsresultat wie in Abbildung 8.17 durch Water-Inflow für die Wasserhöhe 30.

8.3.4 SIFT – Scale Invariant Feature Transform

SIFT

Scale Invariant Feature Transform (SIFT) ist ein von *G. Lowe* vorgeschlagener Ansatz [Low04], der lokale Feature Deskriptoren aufspürt und extrahiert, die größtenteils invariant (d.h. unabhängig) gegenüber Änderungen bezüglich Lichtintensität, Rauschen, Rotation, Skalierung und kleiner Änderungen des Betrachtungswinkels sind. Dabei werden charakteristische Punkte eines Objektes (sog. *Keypoints*) extrahiert, die verwendet werden können um beispielsweise Objekte in einem Bild zu

Abbildung 8.17 — Durch Water-Inflow mit Höhe 30 segmentiertes Bild

identifizieren. Bei den mittels SIFT extrahierten Features handelt es sich im Gegensatz zu Farbhistogrammen (s. Abschnitt 8.3.1) um lokale Features, die von bestimmten Interessensregionen bzw. Objekten extrahiert werden. Es wird nicht das Bild als Ganzes charakterisiert, sondern nur charakteristische Punkte. SIFT transformiert ein Bild in eine große Anzahl lokaler Feature-Vektoren; aus einem 500x500 Pixel großen Bild würden ca. 2000 Feature-Objekte extrahiert. SIFT nutzt hierzu einen vierstufigen Filteransatz, bei dem die kostenintensiveren Schritte am Ende stehen:

Feature, lokal

1. Scale-Space Extrema Detection
2. Keypoint Localization
3. Orientation Assignment
4. Keypoint Descriptor

Scale-Space Extrema Detection

Der sog. *Skalenraum*[1] (engl. *scale space*) eines Bildes ist im 2-dimensionalen Fall definiert als Funktion $L(x, y, \sigma)$, die aus der *Faltung*[2] * (engl. *convolution*) einer Gaußverteilung $G(x, y, \sigma)$ =

Skalenraum

Faltung

[1] http://en.wikipedia.org/wiki/Scale_space 10.10.2007
[2] http://de.wikipedia.org/wiki/Faltung_(Mathematik) 10.10.2007

$\frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$ und einem Eingabebild entsteht (vgl. Abbildung 8.18):

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{8.6}$$

Der Parameter σ bestimmt die Glättung eines Bildes. Je größer σ, desto größer ist die Glättung und desto mehr sind kleine Strukturen im Bild unterdrückt.

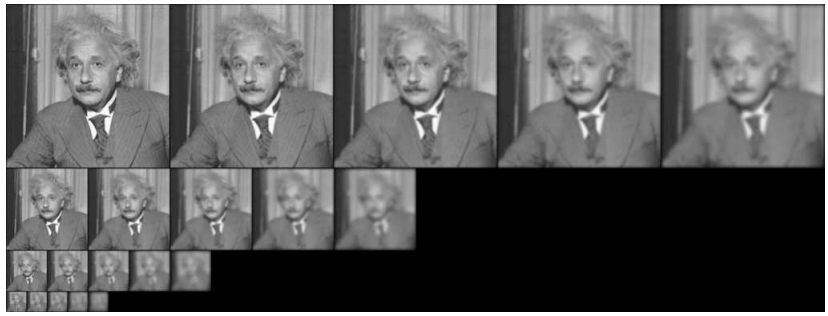


Abbildung 8.18 — Beispielbild, das auf verschiedenen Skalierungsebenen mit einem Gauß-Filter überzogen wurde [FEF04]. Die Glättung nimmt von links nach rechts zu.

Bei SIFT dient nun eine Funktion $D(x, y, \sigma)$ – eine Faltung aus Ursprungsbild und *Difference of Gaussians* (DOGs) – als Basis um *stabile* Features über verschiedene Skalierungen hinweg zu extrahieren (vgl. Abbildung 8.19):

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma) \end{aligned} \tag{8.7}$$

Abbildung 8.20 zeigt eine effiziente Methode wie sich $D(x, y, \sigma)$ konstruieren lassen. Das Ausgangsbild wird schrittweise gefaltet um Bilder zu erhalten, die sich im Skalenraum um einen konstanten Faktor k unterscheiden. k wird für eine Oktave bestehend aus s Intervallen derart gewählt, dass $k = 2^{1/s}$. Damit die Extremafindung eine komplette Oktave abdeckt, braucht es pro Oktave $s+3$ Bilder im Stapel der geglätteten Bilder. Im Originalpapier wird ein Wert von $s = 3$ verwendet. Mehr Intervalle pro Oktave würden zwar zu mehr Keypoint-Kandidaten führen,

Oktave

Keypoint

Abbildung 8.19 — Das Beispielbild dargestellt als DoGs (gruppiert nach sog. Oktaven) [FEF04].

diese wären im Durchschnitt aber weniger stabil. Eine Oktave im Skalenraum entspricht einer Verdopplung von σ, d.h. ein Bild wird mit jeder Oktave um einen Faktor 2 herunterskaliert.

Um nun lokale Maxima bzw. Minima von $D(x, y, \sigma)$ zu ermitteln, wird jeder Punkt mit den ihn umgebenden 8 Nachbarpunkten auf der gleichen Skalierungsebene sowie den 9 Nachbarpunkten der darüber bzw. darunter liegenden Skalierungsebene verglichen. Wenn der jeweilige Wert dem Maximum bzw. Minimum dieser Punkte entspricht, handelt es sich um ein Extremum. Abbildung 8.21 verdeutlicht dies.

Keypoint Localization

In diesem Schritt werden aus der Menge der Keypoint-Kandidaten diejenigen entfernt, die einen geringen Kontrast haben, sowie jene, die unzureichend entlang einer Kante lokalisiert sind. *Kontrast*

Um Extrema mit niedrigem Kontrast zu entfernen, wird durch Fitting einer quadratischen Gleichung in drei Variablen an die DoGs die Lage des Maximums auf Sub-Pixel bzw. Sub-Skalierungsebene festgelegt und damit die Genauigkeit der Keypoint-Kandidaten verbessert. Hierbei wird zunächst die Taylorreihe zweiter Ordnung von $D(x, y, \sigma)$ verschoben, so dass der Ursprung auf dem betrachteten Sample-Punkt liegt: *Taylorreihe*

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T \frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x} \tag{8.8}$$

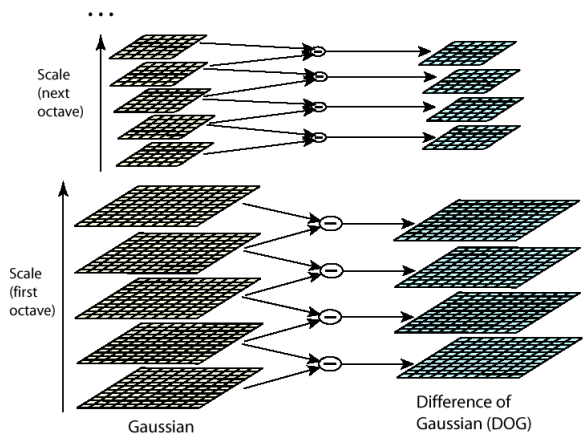


Abbildung 8.20 — Für jede Oktave im Skalenraum wird das Ausgangsbild iterativ mit einem Gauß-Filter gefaltet. Daraus resultieren die auf der linken Seite dargestellten Bilder im Skalenraum. Benachbarte Bilder werden anschließend voneinander subtrahiert, um die DoG-Bilder auf der rechten Seite zu erhalten [Low04].

Pixeldifferenz

D und die zugehörigen partiellen Ableitungen, die durch Pixeldifferenzen benachbarter Sample-Punkte approximiert werden, werden hierbei am entsprechenden Keypoint-Kandidaten berechnet. $\mathbf{x} = (x, y, \sigma)^T$ entspricht der Koordinate des Punktes im Skalenraum. Die Lage des Extremums $\hat{\mathbf{x}}$ der Abschätzung ergibt sich demnach (durch Nullsetzen der Ableitung von Formel 8.8) als:

$$\hat{\mathbf{x}} = -\frac{\partial^2 D^{-1}}{\partial \mathbf{x^2}} \frac{\partial D}{\partial \mathbf{x}} \tag{8.9}$$

Somit ergibt sich der Funktionswert an der Stelle $\hat{\mathbf{x}}$ (durch Einsetzen von Formel 8.9 in Formel 8.8) als:

$$D(\hat{\mathbf{x}}) = D + \frac{1}{2} \frac{\partial D^T}{\partial \mathbf{x}} \hat{\mathbf{x}} \tag{8.10}$$

Falls der Offset $\hat{\mathbf{x}}$ entlang aller Dimensionen größer als $0,5$ ist, bedeutet dies, dass das Extremum näher an einem anderen Sample-Punkt liegt. In diesem Fall wird die Lage des Sample-Punktes verändert und stattdessen

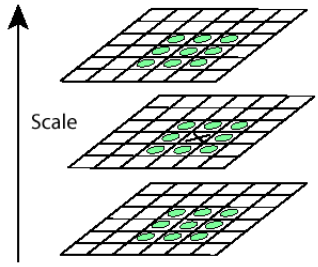


Abbildung 8.21 — Extrema der DoGs werden ermittelt, indem die 26 benachbarten Pixel (markiert mit einem Kreis) mit dem jeweiligen Pixel (markiert mit x) verglichen werden [Low04].

in der Umgebung dieses Punktes interpoliert. Um die interpolierte Näherung der Lage des Extremums zu erhalten, wird der resultierende Offset $\hat{\mathbf{x}}$ zur Lage des Sample-Punktes hinzuaddiert. Anschließend werden alle Extrema mit $|D(\hat{\mathbf{x}})|$ kleiner einem bestimmten Schwellenwert verworfen (s. Abbildung 8.22).

Hesse-Matrix

Hauptkrümmung

Eigenwert

Spur

Determinante

Ob ein Keypoint-Kandidat im Rahmen der Extremwertfindung im Skalenraum unzureichend entlang einer Kante lokalisiert wurde, wird mit Hilfe der 2x2 *Hesse-Matrix*[3] $\mathbf{H} = \begin{pmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{pmatrix}$ berechnet. Ein solches Extremum hat eine kleine *Hauptkrümmung*[4] (engl. *principal curvature*) entlang der Kante, aber eine starke in orthogonaler Richtung. Hierbei nutzt man die Tatsache, dass die Eigenwerte von $\mathbf{H}$ proportional zu den Hauptkrümmungen von D sind. Die Eigenwerte müssen nicht explizit berechnet werden, da nur ihr Verhältnis $r = \frac{\alpha}{\beta}$ von Interesse ist. Sei α der betragsmäßig größere Eigenwert und β der betragsmäßig kleinere. Dann kann die Summe der Eigenwerte mit Hilfe der *Spur*[5] von $\mathbf{H}$ und das Produkt der Eigenwerte mit Hilfe der Determinanten bestimmt

[3] http://de.wikipedia.org/wiki/Hesse-Matrix 10.10.2007
[4] http://en.wikipedia.org/wiki/Principal_curvature 10.10.2007
[5] http://de.wikipedia.org/wiki/Spur_(Mathematik) 10.10.2007

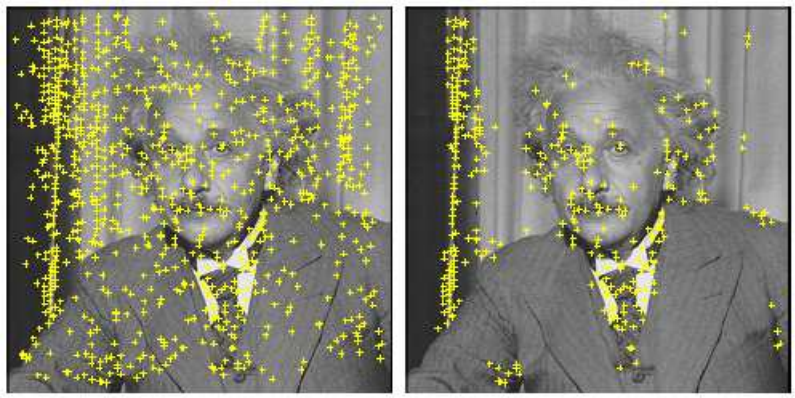


Abbildung 8.22 — Im rechten Beispielbild wurden alle Keypoint-Kandidaten entfernt, die einen niedrigen Kontrast haben ($|D(\hat{\mathbf{x}})| < 0,03$) [FEF04].

werden.

$$Sp(\mathbf{H}) = D_{xx} + D_{yy} = \alpha + \beta$$

$$Det(\mathbf{H}) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta$$

Im unwahrscheinlichen Fall, dass die Determinante negativ wird, haben die Krümmungen verschiedene Vorzeichen und daher wird der Keypoint-Kandidat verworfen.

$$\frac{Sp(\mathbf{H})^2}{Det(\mathbf{H})} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r}$$

Um zu überprüfen, ob das Verhältnis der Hauptkrümmungen unter einem gewissen Schwellenwert r bleibt, muss daher nur $\frac{Sp(\mathbf{H})^2}{Det(\mathbf{H})} < \frac{(r+1)^2}{r}$ überprüft werden. Liegt dieses Verhältnis unter dem Schwellenwert, wird ein Keypoint-Kandidat nicht verworfen. Diese näherungsweise Berechnung benötigt pro Keypoint-Kandidat lediglich 20 Floating-Point-Operationen. Abbildung 8.23 zeigt das resultierende Beispielbild.

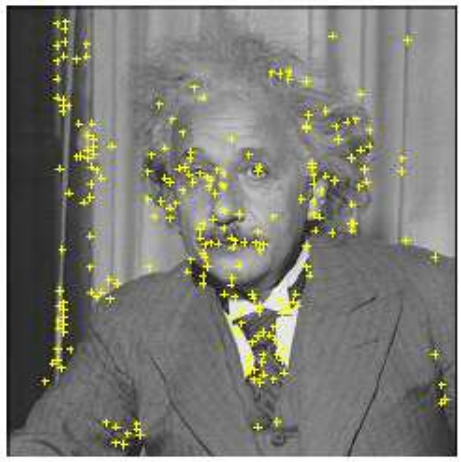


Abbildung 8.23 — Das Beispielbild nach Abschluss der Keypoint Localization. Im Originalpapier wird ein Schwellenwert von $r = 10$ verwendet [FEF04].

Orientation Assignment

Um Invarianz gegenüber Rotation zu erreichen, wird jedem Keypoint-Kandidaten eine konsistente Richtung zugewiesen. Jeder Keypoint wird relativ zu dieser Orientierung dargestellt. Experimentell hat sich dabei die folgende Methode bewährt. Die Skalierungsebene, die der Skalierung des Keypoint-Kandidaten am nächsten liegt, wird ausgewählt und für jeden Sample-Punkt des Gauß-gefilterten Bildes $L(x, y)$ auf dieser Stufe wird der Betrag $m(x, y)$ und die Richtung $\theta(x, y)$ des Gradienten mit Hilfe von Pixeldifferenzen berechnet:

Gradient

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

$$\theta(x,y) = tan^{-1}\left(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}\right)$$

Für jeden Keypoint-Kandidaten wird nun ein Histogramm (von 0 bis 360 Grad) bestehend aus den Richtungen der Gradienten $\theta(x, y)$ aller Sample-Punkte (x, y) innerhalb einer gewissen Entfernung vom

Histogramm

Keypoint-Kandidaten berechnet. Jeder der 36 Bins des Histogramms deckt hierbei einen Winkel von 10 Grad ab. Jeder Sample-Punkt (x, y), der zum Histogramm hinzugefügt wird, wird hierbei mit dem Betrag des Gradienten $m(x, y)$ sowie einem Gauß-gewichteten zirkulärem Fenster (mit einem σ, das dem 1,5-fachen der Skalierung des Keypoint-Kandidaten entspricht) gewichtet. Spitzen in diesem Histogramm entsprechen dominanten Richtungen lokaler Gradienten. Die höchste Spitze des Histogrammes wird ausgewählt und alle anderen Spitzen die mindestens 80% der Höhe des höchsten Histogrammwertes entsprechen, werden ebenso zu potenziellen Keypoints mit der entsprechenden Orientierung. Dieses Vorgehen kann zu mehreren Keypoints am gleichen Ort mit der gleichen Skalierung aber unterschiedlicher Orientierung führen. Aus Gründen der Genauigkeit wird in einem letzten Schritt eine Parabel durch die drei zu jeder Spitze nächsten Histogrammwerte gelegt, um die Position zu interpolieren.

Keypoint Descriptor

Ein Keypoint-Deskriptor wird berechnet (vgl. Abbildung 8.24), indem Betrag und Richtung des Gradienten an jedem Sample-Punkt in einer bestimmten Umgebung des Keypoints berechnet werden (s. Abb. 8.24, linke Seite). Um Rotationsinvarianz zu erreichen, werden die Koordinaten des Deskriptors sowie die Richtungen der Gradienten relativ zur Orientierung des Keypoints gedreht. Die Sample-Punkte werden mit einem Gauß-Fenster gewichtet, das in Abbildung 8.24 durch den schwarzen Kreis angedeutet ist. Diese Sample-Punkte werden anschließend in Histogrammen, die die Richtungen einer 4x4 Region erfassen, zusammengefasst. Die Länge der Pfeile entspricht der Summe des Gradientenbetrags in der Nähe dieser Richtung innerhalb der Region. Abbildung 8.24 zeigt ein 2x2 Deskriptor-Array, das aus einer 8x8 Region resultiert. Im Originalpapier werden 4x4 Deskriptoren aus einem 16x16 Array berechnet. Jedes dieser 16 Histogramme erfasst 8 Richtungen (den Kompassrichtungen Nord, Nord-West, West, Süd-West, etc. entsprechend), woraus

Feature-Vektor

ein 128-dimensionaler Feature-Vektor resultiert. Dieser wird abschließend normalisiert, um Invarianz gegenüber Änderungen in der Beleuchtungsstärke zu erreichen. Abbildung 8.25 zeigt das Beispielbild mit den resultierenden Keypoint-Deskriptoren.

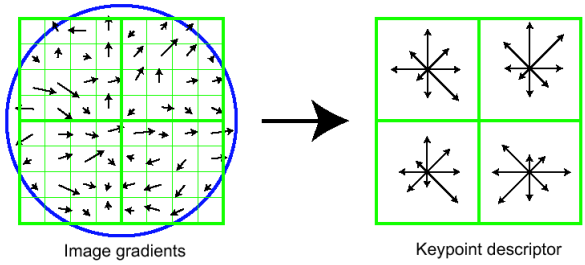


Abbildung 8.24 — Berechnung des Keypoint Deskriptors [Low04]

8.4 Audio Retrieval

Den Ausgangspunkt für eine akustische Indexierung von Audiosignalen bildet deren Signal, das, wenn es in analoger Form vorliegt, zunächst digitalisiert werden muss.

Dazu ist es hilfreich, die Entstehung eines Tones zu betrachten, die von der Vibration eines physikalischen Werkstoffs und den dadurch hervorgerufenen Druckwellenschwankungen in der umgebenden Luft ausgeht. Den Luftdruck kann man nun über die Zeit antragen und so ein Audiosignal darstellen.

Die Amplitude des Signals entspricht dabei der Lautstärke und die Periode bzw. Frequenz (in Oszilationen pro Sekunde) der Tonhöhe. Man unterscheidet dabei den Infraschall (0 bis 20 Hz), den Hörschall (20 Hz bis 20 kHz), den Ultraschall (20 kHz bis 1GHz) und den Hyperschall (1 GHz bis 10 THz).

Pulse Code Modulation

Zur Digitalisierung wird typischerweise die *Pulse Code Modulation* (PCM) eingesetzt, bei der die Werte des Signals zu diskreten Zeitpunkten (z.B. 44100 mal pro Sekunde) abgetastet und dann quantisiert (d.h. durch digitale Werte dargestellt) werden. Abbildung 8.27 verdeutlicht diesen Ablauf, wobei der Tiefpassfilter zur Unterdrückung hoher Frequenzen benötigt wird, die zu Aliasing führen würden – also dazu, das im digitalisierten Signal Töne auftreten, die im Ursprungssignal nicht enthalten waren.

zeitliche Diskretisierung

Wert-Diskretierung

Allgemein kann man dabei zwei Stufen der Diskretisierung unterscheiden. Die zeitliche Diskretisierung (auch *Abtasten* oder *Sampling* genannt) entspricht der Einteilung der Zeitachse in einzelne Stücke. Bei der Wert-Diskretisierung (auch *Quantisierung* genannt) geht es darum einen digi-

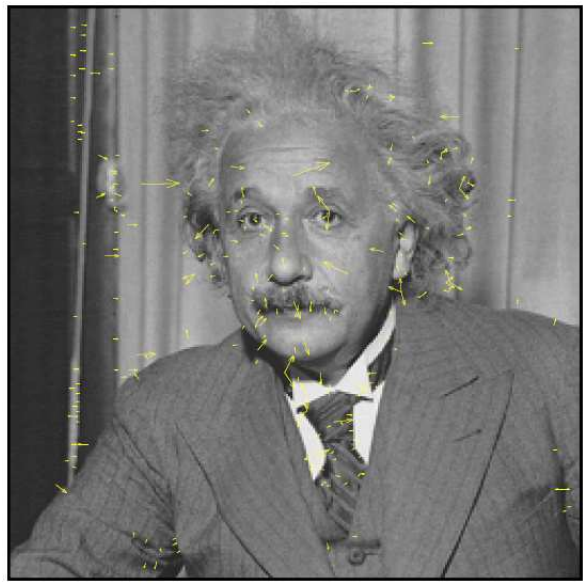


Abbildung 8.25 — Die resultierenden Keypoints dargestellt als Pfeile, die Skalierung und Richtung widerspiegeln [FEF04].

talen Näherungswert für die einzelnen abgetasteten Werte zu finden.

Ein Beispiel mag das Vorgehen bei der Pulse Code Modulation weiter verdeutlichen:

Windowing

Für die Extraktion von Features aus einem Audiosignal ist es nun zunächst sinnvoll, dieses Audiosignal nicht als Ganzes zu betrachten, sondern in einzelne Abschnitte zu zerlegen. Diese Zerlegung erfolgt durch das so genannte *Windowing*. Wie in Abbildung 8.29 dargestellt, wird dazu das Ausgangssignal mit einer Window-Funktion verknüpft, so dass wir ein so genanntes *Frame*, also einen Ausschnitt aus dem Ursprungssignal, erhalten.

Für die Berechnung eines oder mehrerer Features aus dem digitalisierten Audiosignal läuft, wie in Abbildung 8.30 dargestellt, nun das Window von links nach rechts über das Audiosignal und schneidet dabei Frames aus, die sich durchaus überlappen können. Für jedes Frame findet dann bspw. eine *Fouriertransformation* statt, deren Ergebnisse weiter für die Featureberechnung genutzt werden können.

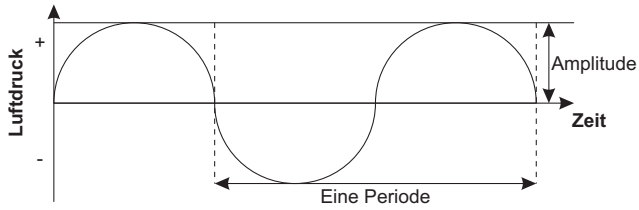


Abbildung 8.26 — Ein akkustisches Signal

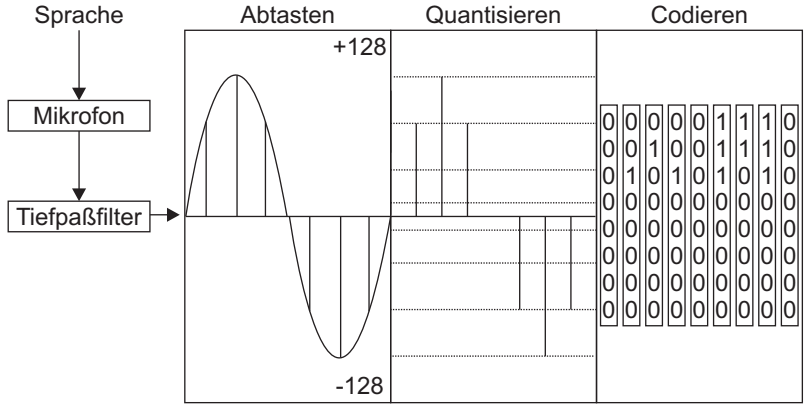


Abbildung 8.27 — Ablauf der Pulse Code Modulation

8.4.1 Audio Features

An dieser Stelle soll nur eine kleine Auswahl der Features aufgezeigt werden, die – analog zu Farbe und Textur bei Bildern – zur Charakterisierung von Audios verwendet werden können.

Energie (Energy)

Energie

Oft ist die akustische Energie von Bedeutung. Die Energie eines zeitdiskreten Signals, wie es bei einem Frame ja angenommen wird, ist wie folgt definiert:

$$E_x = \sum_{n=m-N+1}^{m} x(n)^2 \tag{8.11}$$

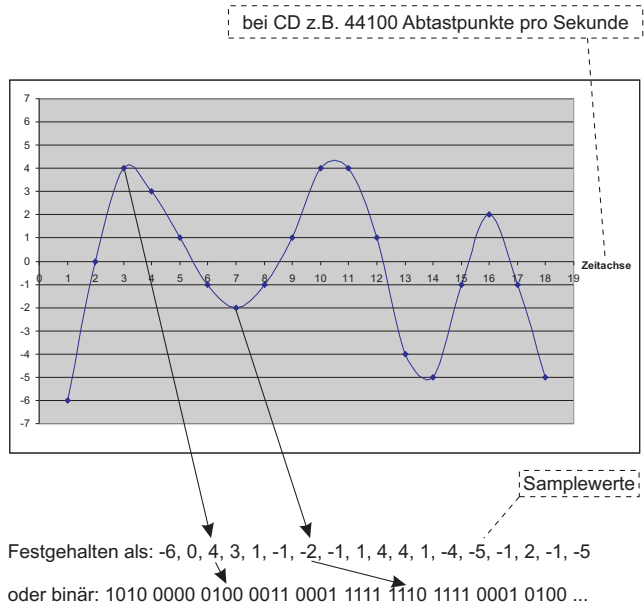


Abbildung 8.28 — Beispiel zur Pulse Code Modulation

$x(n)$ ist hier der Amplitudenwert eines Abtastzeitpunktes, m der Endzeitpunkt des Frames und N die Länge des Frames. In den meisten Systemen, die Energie als Feature verwenden, wird von *Short-Term Energy* gesprochen. *Short-Term* deshalb, weil sich die Energie nur auf einen kurzen Zeitraum, den Frame, bezieht (vgl. [LJZ01]).

Lautstärke

Lautstärke

Die Lautstärke in Dezibel kann näherungsweise als Quadratwurzel aus der Energie des Signals gemessen werden. Die Energie wird typischerweise aus der Fouriertransformation errechnet. Eine wesentlich genauere Lautstärkeabschätzung würde zusätzlich die unterschiedliche Frequenzempfindlichkeit des menschlichen Ohrs berücksichtigen, was mit einem Filter realisiert werden kann (vgl. [WBKW96]).

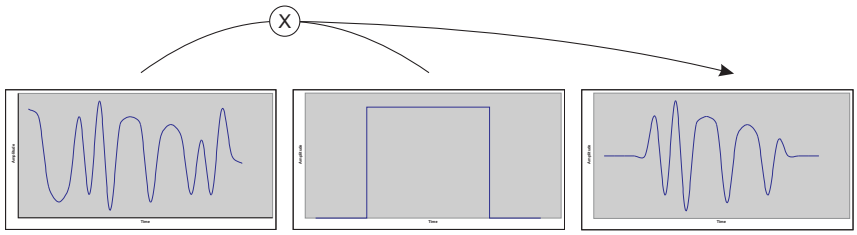


Abbildung 8.29 — Technik des Windowing

Tonhöhe

Tonhöhe

Die Tonhöhe kann abgeschätzt werden, indem man eine Reihe von Fourierspektren untersucht. Für jedes einzelne Frame wird die Frequenz und die Amplitude des Höchstwertes bestimmt. Anschließend wird ein Algorithmus, der den größten gemeinsamen Teiler bestimmt, benutzt, um die Tonhöhe näherungsweise zu bestimmen. Dieses Vorgehen ist sehr rudimentär und führt nur bei periodischen Signalen zu guten Ergebnissen. Für Sprachsignale z.B. ist die Tonhöhe anders zu bestimmen, weil die Wellenform hier in der Periode und in der Struktur innerhalb einer Periode variiert. Außerdem lässt sich oft kein exakter Anfangs- und Endzeitpunkt der Periode für stimmhafte Sprachsegmente bestimmen (vgl. [WBKW96]).

Avarage Zero-Crossing Rate (ZCR)

Avarage Zero-Crossing Rate

Die durchschnittliche Anzahl der Nulldurchgänge (Anzahl der Vorzeichenwechsel innerhalb einer Signalfolge), dementsprechend auch Average Zero-Crossing Rate (ZCR) genannt, ist ein nützliches Feature bei der Sprachanalyse, mit dem auf den Frequenzverlauf eines Signals geschlossen werden kann.

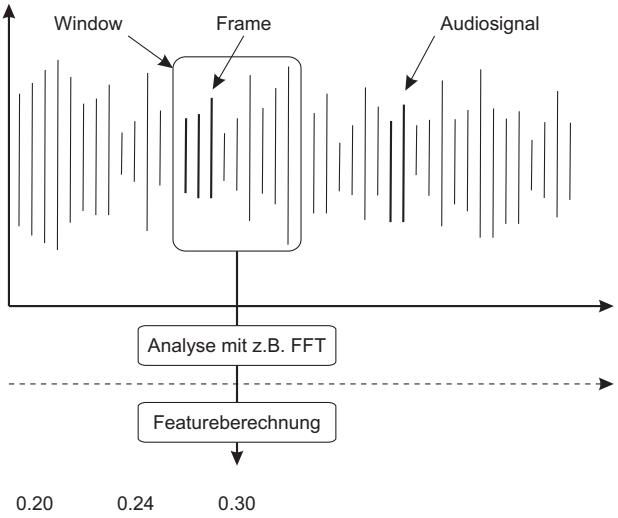
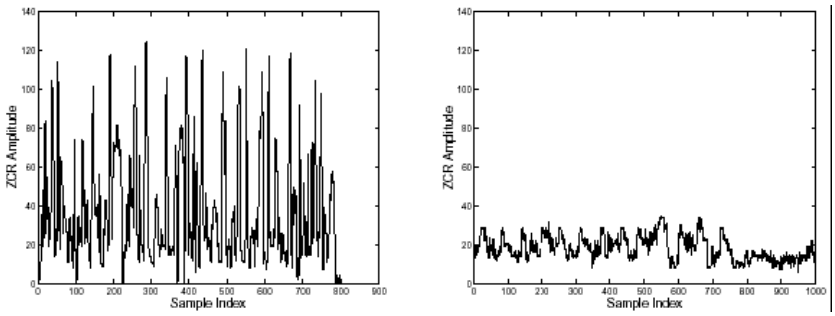


Abbildung 8.30 — Ablauf einer Audioanalyse



Ist innerhalb eines gewissen Zeitabschnittes die ZCR hoch (niedrig), so kann auf ein Signal geschlossen werden, das hohe (niedrige) Frequenzanteile enthält. Unter Berücksichtigung anderer Features, wie Short-Term Energy, können damit beispielsweise Wörter erkannt werden (vgl. [LJZ01]).

8.4.2 Techniken für das Audio Retrieval

inhaltsbasierte Segmentierung

Für die inhaltsbasierte Indexierung von Audiodaten ist zunächst eine inhaltsbasierte Segmentierung und Klassifikation notwendig. Wir wollen

uns hier auf die grobe Einteilung, die so genannte Grundklassifikation, von Audiodaten in Musik, Sprache, Stille und Umgebungsgeräusche beschränken. In weiteren Schritten können dann bspw. Wortgrenzen oder auch Noten erkannt werden.

Segmentierung von Audiodaten

Den ersten Schritt stellt zunächst eine Aufteilung der Audiodaten in einzelne Segmente dar. Dabei werden Audio-Features eingesetzt, um die Segmentierungsgrenzen zu erkennen. Typischerweise wird an Positionen im Audiostrom eine Segmentgrenze gesetzt, an denen die Werte von einem oder mehreren Audio-Features abrupte Wertänderungen aufzeigen.

8.4.3 Techniken zur Grundklassifikation

Die Techniken zur Grundklassifikation in Stille, Musik und Sprache sollen hier kurz exemplarisch aufgezeigt werden (vgl. [LJZ01]).

Erkennung von Stille

Erkennung von Stille

Als »stille Stellen« werden Bereiche definiert, wie nicht wahrnehmbare Geräusche, nicht bemerkbare Hintergrundgeräusche oder kurzes Knacken. Die Erkennung kann auf den ersten Blick ganz einfach mit Hilfe der Lautstärke beziehungsweise der Energie des Signals erfolgen. Einige Experimente haben allerdings ergeben, dass das Signal einiger Hintergrundgeräusche eine höhere Energie hat als das Signal von Musik. Da die Frequenz des Hintergrundgeräusches sehr viel geringer ist, wird zusätzlich die Zero-Crossing Rate zur Klassifikation genutzt.

Erkennung von Musik

Erkennung von Musik

Sprache und Musik sind primär über das Spektrum, das sie jeweils abdecken, unterscheidbar. Sprache liegt in einem Bereich zwischen 100 und 7000 Hz, Musik zwischen 16 und 16000 Hz. Der letztere Bereich deckt aber trotzdem noch die Hintergrundgeräusche mit ab. Eine Unterscheidung kann dann dadurch erfolgen, dass beispielsweise nach Rhythmusmustern gesucht wird oder festgestellt wird, dass für jeden Ton ein Oberton enthalten ist, was für Musik zwingend ist. Für die Erkennung von

Musik sind Zero-Crossing Rate (ZCR), Fundamentalfrequenz (das physikalische Pendant zur Tonhöhe) sowie Harmonie gute Audio-Features zur Bestimmung. Es können folgende Aspekte betrachtet werden: Die zeitliche Änderung der ZCR, die Amplitudenänderung der ZCR oder das Ausmaß der Konzentration der Fundamentalfrequenz an bestimmten Stellen. Für jedes Merkmal werden bestimmte Schwellenwerte eingeführt, nach denen dann klassifiziert werden kann.

Erkennung von Sprache

Erkennung von Sprache

Bei der Kennzeichnung von Sprache können ebenfalls mehrere Aspekte überprüft werden. Der erste ist die Beziehung zwischen Zero-Crossing Rate (ZCR) und Energiekurve. Bei Sprache hat die ZCR-Kurve Spitzen für stimmlose Stellen und Täler für stimmhafte Stellen, während das bei der Energiekurve genau umgekehrt ist. Schneidet man aus diesen Kurven jeweils das obere Drittel ab, so dass nur die Spitzen übrig bleiben, und multipliziert die einzelnen zusammengehörenden Amplitudenwerte miteinander, kommt für Sprache ein Wert nahe Null heraus. Für andere Audiotypen ist dieser Wert wesentlich höher. Der zweite Aspekt ist die Form der ZCR Kurve. Bei Sprache hat die ZCR Kurve eine stabile, niedrige Grundlinie mit mehreren Spitzen darauf. Die Grundlinie ist als Verbindungslinie der Täler definiert. Der Durchschnittswert und die Varianz der Grundlinie kann leicht ermittelt werden.

8.5 Video Retrieval

Segmentierung

Die Grundlage für die Indexierung eines Videostroms stellt dessen Einteilung in sinnvolle Segmente dar. Eine hierarchische Aufteilung eines Videostroms ist in Abbildung 8.31 dargestellt. Dabei wird ein Videostrom in einzelne Szenen, Shots (Kameraeinstellungen) und Frames (Videobild) unterteilt. Ein Frame kann nun verschiedene visuelle Elemente, wie bspw. hier einen Schneemann, enthalten.

Bei den Verfahren zur Segmentierung von Videoströmen wollen wir im Folgenden zwei Kategorien betrachten, zum einen Verfahren zur automatischen Erkennung von Shots und zum anderen Verfahren für die Klassifikation von Kamerabewegungen.

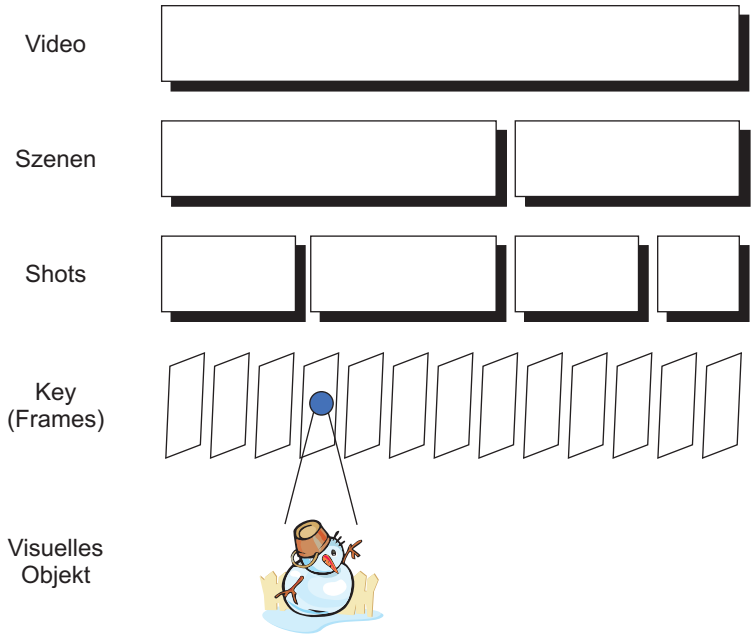


Abbildung 8.31 — Segmentierung eines Videos (aus [FDST98])

8.5.1 Videosegmentierung mittels Shot-Detection

Shot-Detection

Key-Frame

Die Videosegmentierung mittels Shot-Detection teilt ein Video in einzelne Shots auf (vgl. auch [DRA98] und [ZKS93]). Dabei werden für jeden Shot ein oder mehrere Frames als so genannte Key-Frames ausgewählt, die den Shot charakterisieren. Ein Shot bezieht sich auf eine Sequenz von aufeinanderfolgenden Frames mit kontinuierlicher Aktion. Ein Shot-Wechsel ist ein abrupter Wechsel des Inhalts zwischen zwei Frames. Die Aufgabe der Shot-Detection reduziert sich also auf die Erkennung von Grenzen zwischen aufeinanderfolgenden Shots. Basierend auf den unterschiedlichen Ähnlichkeitskriterien für Bilder, die wir im Abschnitt 8.3 kennengelernt haben, lässt sich ein Inhaltswechsel zwischen einzelnen Frames durch die geringe Ähnlichkeit von zwei aufeinanderfolgenden Bildern erkennen und somit der Übergang zwischen zwei Shots identifizieren.

Eine Alternative zur einfachen Nutzung von Ähnlichkeitsmaßen für Bilder bildet der paarweise Pixelvergleich. Dabei werden die einzelnen mit-

einander korrespondierenden Pixel zweier aufeinanderfolgender Frames verglichen, um festzustellen, wie viele Pixel sich geändert haben. Eine Pixeländerung ist gegeben, wenn der Differenzwert zwischen den Intensitätswerten der Pixel in den beiden Frames einen vorgegebenen Grenzwert überschreitet. Übersteigt die Anzahl der geänderten Pixel einen vorgegebenen Prozentsatz, so soll dies auf die Grenze zwischen zwei Shots schließen lassen. In der Realität ist diese Methode mit großen Problemen behaftet, da bspw. auch Kamerabewegungen zu einer starken Änderung der Pixelwerte zwischen zwei Frames führen und damit irrtümlich Shot-Grenzen erkannt werden. Um die Problematik des direkten Pixelvergleichs zu umgehen bietet sich bspw. der Vergleich von Pixelblöcken an, so dass die Toleranz gegenüber kleinen Objekten und langsamen Kamerabewegungen erhöht wird.

Eine gegenüber Bewegungen von Objekten im Shot und der Kamera selbst robustere Methode zur Identifizierung von Shots stellt der Einsatz von Metriken dar, die sich auf das gesamte Bild beziehen; so bspw. der Vergleich zweier Frames mittels ihrer Histogramme.

8.5.2 Klassifikation der Kamerabewegung (Motion Detection)

Motion Detection

Ein weiterer wichtiger Aspekt des Video Retrieval ist die Erkennung von Bewegungen (*Motion Detection*). Damit sind grundsätzlich zwei Ziele verbunden. Zum einen spielt Motion Detection bei der Shot Detection eine wichtige Rolle. Um die Effektivität der Shot Detection zu steigern, müssen Kamerabwegungen, wie bspw. Zooms und Schwenks, erkannt werden. Zum anderen kann anhand der Motion Detection die Charakterisierung von Shots vorgenommen werden, wobei hier nun die Erkennung von Objektbewegungen eine Rolle spielt.

Optical Flow

Ein Feature, das es ermöglicht, Kamerabewegungen zu erkennen, ist der so genannte *Optical Flow*. Mittels des Optical Flow können Kameraschwenks und Zoom-Operationen erkannt werden. Abbildung 8.32 zeigt Optical Flows für diese Kamerabewegungen.

8.5.3 Key Frame Extraction

Für eine Indexierung von Videos reicht die alleinige Erkennung von Shots nicht aus. Ein Shot wird vielmehr durch ein oder mehrere Frames charakterisiert. Diese Frames werden als Key Frames bezeichnet.

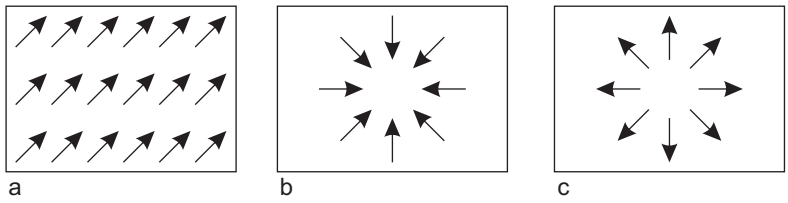


Abbildung 8.32 — Patterns für Bewegungsvektoren, die aus Kamerabewegungen (a) und Zoom-Operationen (b, c) resultieren (aus [ZKS93])

Key Frame Extraction

Eine einfache Methode der Key Frame Extraction stellt die Auswahl des ersten und des letzten Frames eines Shots dar. Auf diesen Frames basierend kann nun der Inhalt des Shots repräsentiert werden. Die Key Frames selbst können dabei durch verschiedene Eigenschaften, wie bspw. Farbe, Textur, Shapes, Kanten oder Bewegungen repräsentiert werden.

Mosaicing

Eine andere Methode, den Inhalt eines Shots abzubilden, stellt das so genannte *Mosaicing* dar. Bei dieser Methode wird ein Bild erzeugt, das den Inhalt des gesamten Shots repräsentiert, indem alle Frames des Shots in einem Bild vereinigt werden, wobei sich ihre Ausrichtung in dem künstlich erzeugten Bild an den Bewegungen von Bild zu Bild orientiert.

8.6 Retrieval für Multimedia-Dokumente

Wir haben in diesem Kapitel bisher primär die Suche nach einzelnen Medienobjekten betrachtet. Dabei kann es sich um Texte, Bilder, Audios oder Videos handeln. Häufig müssen aber strukturierte Multimedia-Dokumente verwaltet werden, die sich primär durch die Nutzung verschiedener Medien in einem Dokument und die Strukturierung des Dokuments – z.B. in Kapitel und Abschnitte – charakterisieren lassen.

Multimedia-Datenbanken

Bei den Systemen zur Verwaltung derartiger Dokumente spricht man häufig von *Multimedia-Datenbanken*. Zu den Aufgabenbereichen einer Multimedia-Datenbank gehören dabei die Speicherung, die Manipulation und das Retrieval für alle Typen digital repräsentierbarer Medienobjekte, wie Text, Standbilder, Video und Audio.

Ein Problem ist dabei natürlich, dass die Information über den Inhalt (oder die Semantik) der Objekte im Allgemeinen in den Medienobjekten »verborgen« ist und nicht explizit in Attributen vorliegt. Es geht hier also

darum, diesen Inhalt ansprechbar zu machen. Denkbare Ansätze hierzu wollen wir im Folgenden betrachten.

Binary Large Object, BLOB

An dieser Stelle ist noch anzumerken, dass in der Literatur und vor allem bei Produktbeschreibungen oft schon dann von Multimedia-Datenbanken gesprochen wird, wenn Multimedia-Objekte in einer Datenbank unstrukturiert als BLOBs (Binary Large Objects) abgelegt werden können. Die Datenbank selbst weiß in diesem Fall aber nicht, dass es sich bei einem Medienobjekt z.B. um ein Bild handelt. Sie verwaltet nur die entsprechende Bytefolge ohne Kenntnis über deren Interpretation zu haben. Dies ist in unserem Sprachgebrauch allerdings noch keine Multimedia-Datenbank.

Begrifflichkeiten zu Multimedia-Datenbanken

Wir unterscheiden hier: *Medien-Objekte*, *Multimedia-Objekte* und *Multimedia-Dokumente*.

Medien-Objekt

- Ein **Medien-Objekt** ist ein Objekt, das genau ein Medium nutzt. Beispiele sind einfache Attributwerte, Text, ein Bild, ein Video/eine Videosequenz oder ein Audio/eine Audiosequenz.

Multimedia-Objekt

- Ein **Multimedia-Objekt** ist eine Komponente, die sich strukturiert aus mehreren Medien-Objekten und/oder Multimedia-Objekten zusammensetzt.

Multimedia-Dokument

- Ein **Multimedia-Dokument** schließlich ist eine Komponente, die sich strukturiert aus mehreren Medien-Objekten und/oder Multimedia-Objekten zusammensetzt und einen abgeschlossenen Charakter hat. Damit ist ein Multimedia-Dokument also ein spezielles Multimedia-Objekt.

Die Suche nach Multimedia-Dokumenten oder Teilen davon sollte dann auf Basis der Struktur der Dokumente, der Attribute und des (semantischen) Inhalts der Medien-Objekte möglich sein.

Problemstellungen im Bereich Multimedia-Datenbanken

Im Weiteren interessiert uns natürlich primär der Aspekt der Suche in und nach Multimedia-Dokumenten. Für das Verständnis der Zusammenhänge ist es aber wichtig, einen breiteren Überblick über die

Problemstellungen zu haben, mit denen man sich im Zusammenhang mit Multimedia-Datenbanken beschäftigt. Diese Problemstellungen geben grob auch die aktuellen Forschungsschwerpunkte in diesem Bereich wieder. Sie können wie folgt gegliedert werden:

- **Verwaltung strukturierter, multimedialer Daten**

  Einen ersten Teilbereich bildet hier die Modellierung der Daten. Die Frage lautet: Wie kann die Struktur von Multimedia-Dokumenten abgelegt und verwaltet werden? Einen denkbaren Lösungsansatz bilden objektorientierte Datenbanken, die mächtigere Konzepte anbieten als klassische relationale Datenbanken.

  Einen wichtigen Teilbereich bildet in diesem Zusammenhang auch der Aspekt der *strukturierten* Daten. Dabei geht es vor allem darum, wie die Struktur der Dokumente (z.B. bei Hypermedia-Dokumenten) zweckmäßig verwaltet wird, so dass sie für das Arbeiten mit den Daten und die Suche in diesen adäquat zugreifbar ist.

  Einen weiteren Problembereich bei der Verwaltung multimedialer Daten bildet schließlich auch die Explosion des Datenvolumens. Durch die Verwaltung von Bildern, Audios und Videos übersteigt der Umfang der Daten schnell die Maßstäbe herkömmlicher Datenbanken. Dadurch ergeben sich z.B. auch im Hinblick auf die Transaktionsverwaltung neue Problemstellungen.

- **Inhaltsbasierte Suche**

  Ein erster Aspekt der inhaltsbasierten Suche ist die Formulierung der Anfragen. Die hierzu benötigten Anfragesprachen sollten die Möglichkeit bieten, die Eigenschaften einzelner Medienobjekte (z.B. Farbe oder Textur) und auch die Dokumentstruktur in einer Anfrage zu adressieren.

  Zur Anfragebearbeitung werden dann unter anderem Indexstrukturen benötigt, die analog zu invertierten Listen bei Texten und B-Bäumen in klassischen Datenbanken die effiziente Bearbeitung von Ähnlichkeitsanfragen auf Medienobjekten unterstützen.

  Die Feature-Extraktion bildet einen weiteren Problembereich. Hier müssen für alle Medientypen geeignete automatische Verfahren bereitgestellt werden.

- **Präsentation eines Anfrageergebnisses**

Eine tabellenartige Darstellung, wie sie in herkömmlichen Datenbanken verwendet wird, ist hier nur bedingt geeignet. Gerade bei Video- und Audiodaten ist das direkte »Abspielen« aus der Datenbank von Bedeutung. Man denke z.B. an Anwendungen im Bereich *Video on Demand*. Hier sind Probleme des Streaming und der Pufferverwaltung von Bedeutung.

Auch der Aspekt der Ausgabequalität muss berücksichtigt werden. So können bei Bildern zunächst nur Thumbnails erwünscht sein. Bei einem Video wird unter Umständen ein Abspielen mit einer an den Kommunikationskanal angepassten verminderten Auflösung und Qualität gewünscht.

- **Daten- und Formatunabhängigkeit**

  Einen weiteren wichtigen Problembereich bei Multimedia-Datenbanken bildet die Formatunabhängigkeit. So sollte bei der Ausgabe von Bildern oder Videos die Ausgabe in einem Format möglich sein, das der Anfragende bestimmt. Hierzu muss innerhalb der Datenbank zunächst eine Konvertierung vom internen Verwaltungsformat in das gewünschte Ausgabeformat erfolgen.

  Dabei kann es Konvertierungen innerhalb eines Medientyps (z.B. von MPEG nach RealVideo) oder über Mediengrenzen hinweg (z.B. von Audio nach Text durch eine Spracherkennung) geben.

Im Folgenden werden wir uns nun zunächst mit der Modellierung der Daten und anschließend mit Anfragen auf den Daten beschäftigen, weil die Möglichkeiten im Hinblick auf die Anfragen zwangsweise von der Modellierung der Daten abhängen.

Modellierung mit relationalen und objektorientierten Datenbanken

Herkömmliche relationale Datenbanken haben leider erhebliche Probleme bei der Verwaltung komplexer und/oder strukturierter Dokumente. Die klassische Modellierung mit Relationen oder Tabellen und den üblichen Normalformen stößt hier an ihre Grenzen. Beispiele hierfür sind die Daten in CAD-Systemen, Software-Entwicklungsumgebungen oder eben auch Multimedia-Datenbanken. Ein Hauptproblem dabei ist, dass klassische relationale Datenbanken keine Unterstützung bei der Verwaltung von geometrischen/geographischen Daten, Textdokumenten oder multimedialen Daten bieten. Dies ist umso schmerzlicher, weil es auch keine Möglichkeiten zur Definition neuer »Datentypen« gibt.

Als Lösung bieten sich hier objektorientierte Datenbanken an, die die Konzepte relationaler Datenbanken in mehrere Richtungen erweitern:

- »Datentypen«/»Objekte«/»Klassen« können neu definiert werden.
- Objekte in der Datenbank haben nun einen *Zustand* (gegeben durch ihre Daten) und ein *Verhalten* (Methoden, die auf ihnen ausgeführt werden können)

Als Beispiel mag hier eine einfache Klassendefinition in O2C dienen:

```
class Person
   type tuple (
      public name : string,
      public first_name : string,
      public birthday : Date)
   method
      public getAge : integer;
end;
```

Eine Anfrage an diese Klasse könnte nun z.B. nach Personen suchen, die 60 Jahre alt sind, und hierzu die Methode `getAge` nutzen:

```
select first_name, name from Person where getAge(this) = 60
```

Auch die Beziehungen zwischen Objekten können in objektorientierten Datenbanken explizit ausgedrückt werden. Hierzu ein Beispiel in ODL (*Object Definition Language*) nach ODMG 93:

```
interface Firma
( extent Firmen,
   keys Firmenname )
{
   attribute String Firmenname;
   attribute Struct
      Adresse{String Strasse, String Ort}
      Hauptsitz;
   relationship Leiter Chef;
      inverse Leiter::ist_Chef_von;
   relationship List<Zweigstelle>
      Zweigstellen
      inverse Zweigstelle::von_Firma;
};
```

Hier wird festgelegt, dass eine `Firma` eine Beziehung zu einem `Leiter` hat. Der Name dieser Beziehung ist `Chef`. Zu dieser Beziehung besteht

auch die inverse Beziehung mit dem Namen `ist_Chef_von`. Ferner gibt es zu einer Firma eine Reihe von Zweigstellen. Diese Beziehung wird durch eine Liste modelliert, da eine Firma mehrere Zweigstellen haben kann.

Schließlich bieten objektorientierte Datenbanken auch die Möglichkeit durch Vererbung eine Typhierarchie zu bilden. Hierzu betrachten wir wieder ein Beispiel in O2C:

```
class Employee inherit Person
   type tuple (
      public room: Office,
      public equipment : set(Computer))
      method
         assignOffice(room : Office);
end;
```

Objektorientierte Datenbanken bieten damit den Vorteil, dass komplexe Objekte gut nachgebildet, neue »Datentypen« definiert und Typen über die Vererbung abgeleitet werden können. All dies ist bei der Modellierung strukturierter Multimedia-Dokumente sehr hilfreich.

Einige Probleme können aber auch objektorientierte Datenbanken nicht lösen. So bieten sie keine Unterstützung für sehr große Objekte. Ebenso ist ihnen ein Konzept der Vagheit fremd, wie man es z.B. beim unsicheren Erkennen von Personen auf Bildern brauchen könnte. Auch eine Ähnlichkeitssuche oder eine Formatunabhängigkeit werden nicht unterstützt.

Dennoch erscheinen objektorientierte Datenbanken aufgrund ihrer umfangreichen Modellierungsmöglichkeiten als geeigneter Ausgangspunkt für Multimedia-Datenbanken.

Informationen in Multimedia-Datenbanken

Bei der Modellierung von Multimedia-Dokumenten stellt sich nun die Frage, welche Informationen in Multimedia-Datenbanken verwaltet werden müssen. Hierbei ist zunächst wichtig, dass zusätzliche Informationen zur Unterstützung der Suche nach Multimedia-Objekten wünschenswert wären. Hierzu zählen Informationen zur logischen Struktur (für Text z.B. die Struktur mit Kapiteln, Abschnitten und Paragraphen oder für Video die Struktur mit Szenen und Shots) ebenso wie konzeptuelle Informationen zum Inhalt (z.B. »die Person im Bild ist Nelson Mandela«).

Das Retrieval von Multimedia-Objekten auf Basis ihres semantischen Inhalts kann dabei auf verschiedenen Wegen erreicht werden:

- Eine erste Möglichkeit bilden manuell eingegebene Attribute, die den Inhalt der Multimedia-Objekte strukturiert beschreiben. Handelt es sich z.B. um eine Datenbank mit Vogelbildern, so könnten hier verschiedene Attribute zu den abgebildeten Vögeln, wie ihre Schnabelform, ihre Gefiederfarbe oder ähnliches mehr erfasst werden.
- Alternativ könnte die Beschreibung durch die manuelle Eingabe eines unstrukturierten Texts erfolgen, in dem die gleichen Informationen als Fließtext angegeben werden.
- Schließlich ist eine automatische semantische Analyse denkbar, die sich im Allgemeinen auf spezifisches Domänenwissen stützen muss. So könnte man z.B. die Gefiederfarbe auch mit Verfahren der Bildanalyse ermitteln, die ausnutzen, dass alle Bilder der Datenbank einzelne Vögel zum Hauptgegenstand haben.

Das Modell der Multimedia-Daten – oder auch das Schema – sollte daher vier grundsätzliche Bereiche abdecken:

1. *die eigentlichen Medienobjekte*

   Hier geht es um die Speicherung der eigentlichen Medienobjekte, die ggf. in verschiedenen Formaten erfolgen kann. Zu den Medienobjekten sollten dabei so genannte *Registrierungsdaten* verwaltet werden, die im Falle eines Bildes z.B. die Größe des Bildes, das Verwaltungsformat oder die Farbtiefe enthalten können.

   Registrierungsdaten

2. *die Strukturierung der Medienobjekte zu Multimediadokumenten*

   Unter diesem Aspekt verstehen wir die hierarchische Dekomposition der Multimedia-Dokumente oder -Objekte. Dabei ist typischerweise eine Baumstruktur zu modellieren, die ggf. um Querverweise und Links zu ergänzen ist.

3. *die Interpretation der Medienobjekte und Multimediadokumente*

   Um den Inhalt von Medienobjekten und Multimediadokumenten adressieren zu können erscheint es sinnvoll, Interpretationsobjekte mit in die Datenbank aufzunehmen, die die Dinge interpretieren, die in einem Medienobjekt oder Multimediadokument vorkommen.

Interpretationsobjekte

Potentiell können solche Interpretationsobjekte für alle Ebenen der Hierarchie eines Dokuments sinnvoll sein. So kann ein Interpretationsobjekt zu einem einzelnen Bild oder auch einem Bildsegment die darauf abgebildeten Dinge beschreiben. Ein Interpretationsobjekt zu einem ganzen Kapitel kann sich dagegen auf die Intention des Kapitels oder andere Aspekte beziehen, die für das gesamte Kapitel von Bedeutung sind.

4. *Informationen zu Realweltobjekten, die in den Medienobjekten vorkommen*

Realweltobjekt

Schließlich sollte es die Möglichkeit der Verbindung eines Interpretationsobjekts mit einem zugehörigen Realweltobjekt geben. Wenn wir z.B. eine Datenbank mit Tennisbildern verwalten, dann erscheint es nicht sinnvoll, in einem Interpretationsobjekt zu einem einzelnen Bild alle Informationen zu dem dort abgebildeten Tennisspieler anzugeben. Vielmehr sollten diese Informationen bei einem so genannten Realweltobjekt verwaltet werden, das die für das System relevanten Informationen zu dem Tennisspieler verwaltet, die von den konkreten Medienobjekten unabhängig sind. Das Interpretationsobjekt zu einem konkreten Bild verweist dann z.B. auf das Realweltobjekt für Boris Becker und gibt an, dass Boris Becker auf dem Bild z.B. »verkrampft« wirkt.

Abbildung 8.33 verdeutlicht nun eine mögliche Modellierung der Medienobjekte und der Struktur. Die Interpretationsobjekte und die Realweltobjekte werden wir dann in Abbildung 8.34 betrachten.

Die durchgezogenen Pfeile repräsentieren die Vererbungsstruktur. Die *media values* entsprechen den eigentlichen Medienobjekten. Diese werden in den *media units* um Registrierungsdaten ergänzt. Den *media units* stehen die *composites* gegenüber, die die parallele und die sequentielle Komposition (= Zusammenfassung von Bestandteilen) erlauben. Ein Beispiel für eine parallele Komposition wäre ein *movie shot*, bei dem ein *audio shot* und ein *video shot* kombiniert werden.

Die Multimedia-Objekte weisen nun Beziehungen zu den Interpretationsobjekten auf, deren Modellierung in Abbildung 8.34 dargestellt ist. Die Interpretation des Inhalts eines Multimedia-Objekts wird dabei in einem Interpretations-Objekt verwaltet, das seinerseits ein komplexes Objekt sein kann, welches aus einfacheren Teilen besteht.

Für *composite objects* und *media units* gibt es verschiedene Interpretationsobjekte, weil die Interpretationsobjekte zu *composite objects* z.B.

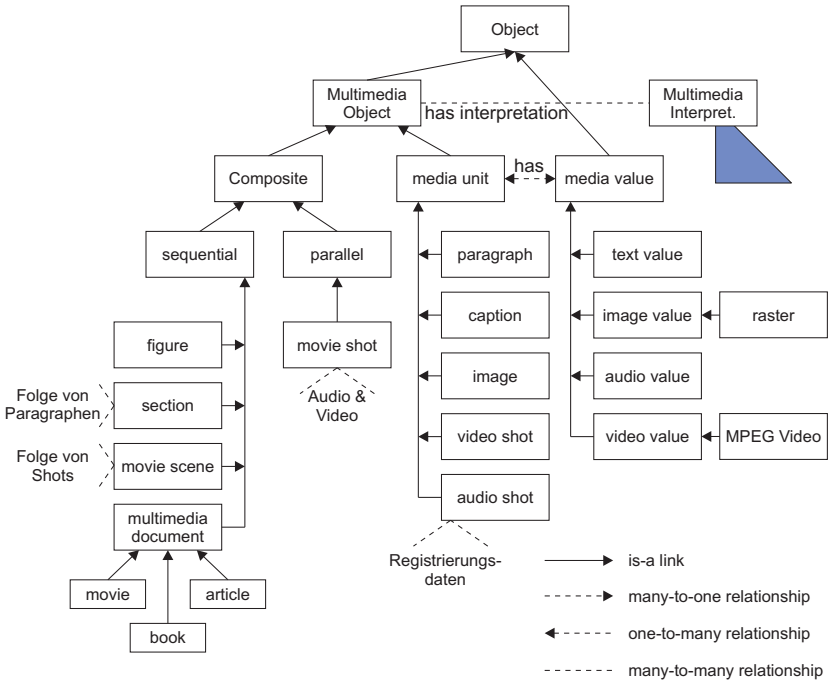


Abbildung 8.33 — Modellierung der Medienobjekte und der Struktur nach [AC97]

gezielt Informationen zur Komposition aufnehmen können.

Bei den Realweltobjekten sollte und muss man sich natürlich auf die Modellierung der Typen beschränken, die in der konkreten Anwendung benötigt werden. So macht es z.B. Sinn in einer Datenbank über Tennis Informationen zu Spielern, Tennisplätzen und Turnieren aufzunehmen, eine Modellierung der Vögel, die vielleicht zufällig auf einem Bild vorhanden sind, erscheint aber nicht sinnvoll.

Anfragen in Multimedia-Datenbanken

Nachdem wir im vorangegangenen Abschnitt die Modellierung von Multimedia-Dokumenten betrachtet haben, wollen wir uns nun der Formulierung von Anfragen auf diesen Daten zuwenden. Idealerweise sollte der Benutzer dabei in die Lage versetzt werden, den Inhalt von

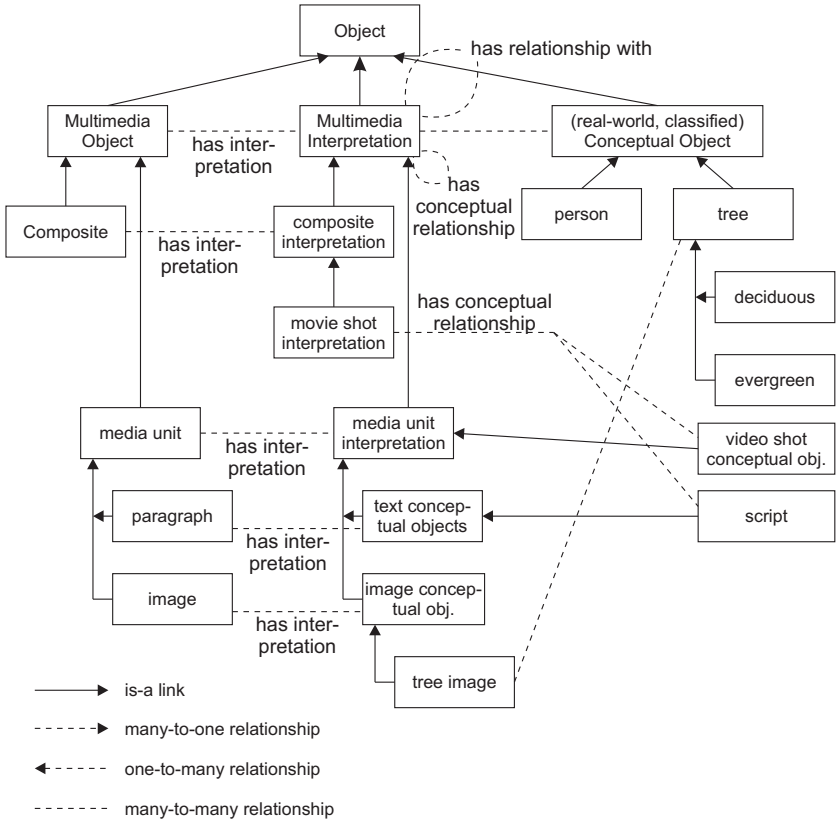


Abbildung 8.34 — Modellierung der Interpretations- und Realweltobjekte nach [AC97]

Multimedia-Objekten in Anfragen in folgenden Variationen zu adressieren:

- über die Werte, der die Semantik beschreibenden Attribute des Multimedia-Objekts

  Beispiel: Wenn in einer Bilddatenbank zu Vögeln deren Schnabelform in den Interpretations- oder Realweltobjekten vermerkt ist, sollte die Suche nach Vögeln mit einem spitzen Schnabel möglich sein.

- über Wörter oder Begriffe, die in der textuellen Beschreibung des Inhalts eines Multimedia-Objekts enthalten sind

  Beispiel: Es sollte eine Anfrage nach einem Film mit einem »ähnlichen« Titel wie »Häuser in Italien« möglich sein.

- über die globalen Eigenschaften der Medien-Objekte

  Beispiel: Man sollte in einer Bilddatenbank für ein Beispielbild nach Bildern mit einer »ähnlichen« Farbverteilung oder Textur suchen können.

- über lokale/partielle Eigenschaften der Medien-Objekte

  Beispiel: In einer Bildatenbank sollte die Suche nach Bildern mit einem roten Teilbereich unterstützt werden.

- über die visuellen Eigenschaften und räumlichen Beziehungen zwischen den konzeptuellen Objekte, die in einem Multimedia-Objekt vorkommen

  Beispiel: Es sollte möglich sein, z.B. alle Bilder zu suchen, in denen ein Redner vor einer Zuhörermenge abgebildet ist.

- über die tatsächlichen Eigenschaften und die tatsächlichen Beziehungen zwischen den konzeptuellen Objekten, die in einem Multimedia-Objekt vorkommen

  Beispiel: Man sollte nach Bildern suchen können, auf denen eine Person im Nahbereich und ein Flugzeug im Hintergrund abgebildet ist. Dies macht erforderlich, dass das System berücksichtigen kann, dass eine tatsächliche Beziehung zwischen den Objekten im Bild einer bestimmten konzeptuellen Beziehung zwischen den realen Objekten entspricht (z.B. klein $\leftrightarrow$ groß im Bild kann nah $\leftrightarrow$ fern in der Realität entsprechen).

- über das zeitliche Verhalten der konzeptuellen Objekten, die in einem Multimedia-Objekt vorkommen

  Beispiel: Suche Szenen, in denen »etwas« von links nach rechts durch das Bild läuft.

- auch beliebige Kombinationen der Anfragearten sollten möglich sein

  Beispiel: Suche alle Dokumente, die sich mit Gesundheits-Diäten beschäftigen und die auf ihrem Titelbild einen Fruchtkorb mit roten und grünen Äpfeln haben.

Die obigen Anfragemöglichkeiten erfordern natürlich ein sehr leistungsfähiges System und praktisch alle heute existierenden Systeme sind von diesem Ideal noch sehr weit entfernt. Der einzig gangbare Weg besteht derzeit darin, für das konkrete Anwendungsfeld spezifische Anfrageformen zu unterstützen. Die allgemeine Formulierung von beliebigen Anfragen in einfacher Form wird wohl ein Wunschtraum bleiben.

Die Konsequenz aus diesen Überlegungen ist, dass man sich in Abhängigkeit von der konkreten Anwendung gezielt überlegt, welche Anfragemöglichkeiten hier benötigt werden, und wie diese benutzungsfreundlich zur Verfügung gestellt werden können. Dazu wären bei Anfragen nach Bildern z.B. folgende Arten möglich:

- Direkt mit einer Anfragesprache wie SQL auf Basis der Attribute und Beziehungen im Modell gestellte Anfragen. Dies impliziert aber, dass der Benutzer das Schema und die verwendeten Begriffe kennen muss, um eine entsprechende Anfrage korrekt stellen zu können.

- Anfragen mit Hilfe einer Zeichnung (*sketch*), die charakteristische Aspekte der gesuchten Bilder aufzeigt. Der Benutzer skizziert dazu die konzeptuellen Objekte im Bild, sowie deren Art und relative Positionierung zueinander.

- Anfragen mit Hilfe eines Beispielbilds, zu dem dann ähnliche Bilder (hinsichtlich Farbe, Textur, Form, ...) gesucht werden.

- Anfragen, die sich auf Objekte beziehen, die mit den eigentlich gesuchten Bildern verknüpft sind. Ist ein Bild z.B. mit Text verknüpft (etwa mit einer Bildunterschrift oder dem umgebenden Text in einem Abschnitt), so kann sich die Suche per »Textretrieval« auf diesen Text beziehen.

8.7 Zusammenfassung

Der Bereich des Multimedia Information Retrieval betrachtet einerseits die Suche nach einzelnen Medienobjekten (wie Bildern, Audios oder Videos) und andererseits die Suche in und nach strukturierten Multimedia-Dokumenten.

Bei den einzelnen Medienobjekten werden hierzu typische Eigenschaften (*Features*) ermittelt, auf deren Basis Ähnlichkeitsberechnungen durchgeführt werden können. Daneben werden auch Verfahren zur Segmentierung und Konvertierung von Medienobjekten genutzt. Bei strukturierten Multimedia-Dokumenten besteht ein erstes Problem in der Modellierung der Dokumente dabei, die Medienobjekte, die Struktur, Interpretationsobjekte und Realweltobjekte zu berücksichtigen. Die Formulierung von Anfragen muss sich dann auf diese Modellierung beziehen.

Weiterführende Informationen zum Thema finden sich z.B. in [Ste00], [ABH97] oder [Lew01].

9 Information Retrieval und das World Wide Web

Unter Mitwirkung von Dipl. Wirtsch.-Inf. Martin Eisenhardt

Das World Wide Web (WWW) ist zu einer der Hauptinformationsquellen in der Informationsgesellschaft geworden. Viele Informationsanbieter veröffentlichen ihre Informationen inzwischen primär im WWW. Als Beispiele sind Nachrichtenagenturen, Forschungs- und Lehreinrichtungen sowie Wirtschaftsunternehmen zu nennen. Durch diesen Trend zur elektronischen Publikation ist das WWW zu einer unvorstellbaren Größe gewachsen. Schätzungen gehen davon aus, dass momentan weit mehr als 10 Milliarden Dokumente im WWW existieren. Google gibt unter Warum man Google benutzen sollte an über 8 Milliarden URLs im Index zu haben (letzter Abruf: 11.12.2006).

Obwohl das WWW also inzwischen als *die* Informationsquelle anzusehen ist, hat das WWW auch Probleme. Eines der schwerwiegendsten Probleme ist dabei die Tatsache, dass das WWW eine als *anarchisch* einzustufende (Un-)Struktur aufweist. Da es keinerlei zentrale Instanzen gibt, welche die Publikationen im WWW ordnen (wie etwa in einer Bibliothek die Bibliothekare), ist das Auffinden einer bestimmten Information im WWW eine nicht-triviale Aufgabe. Tatsächlich wenden viele Nutzer im WWW mehr Zeit für die Suche nach Information auf als für die Nutzung der gefundenen Information.

Im vorliegenden Kapitel werden wir uns daher mit der Nutzung von IR-Methoden zur Suche im WWW beschäftigen. Wir betrachten dazu folgende Teilaspekte:

9.1 Einführung
9.2 Aufbau einer Suchmaschine
9.3 Begriffsdefinitionen
9.4 Techniken des Crawling
9.5 Typische Funktionen einer Suchmaschine

9.6 Erweiterte Funktionen
9.7 Spezielle Verfahren im Web Information Retrieval
9.8 Probleme bei Suchmaschinen
9.9 Zusammenfassung

9.1 Einführung

Suchmaschine

Eine Möglichkeit, die Suche nach Informationen effizient zu unterstützen, sind *Suchmaschinen*. Sie bieten dem Nutzer eine Schnittstelle an, über die er Suchbegriffe, Phrasen und weitere Suchkriterien wie etwa Dokumenttyp und -größe, Aktualität und Fundort angeben kann. Die Suchmaschine ermittelt aus diesen Angaben dann passende Dokumente. Hierzu wird ein Index genutzt, der über *Crawler* erstellt wurde und Informationen über einen großen Teil der im WWW publizierten Dokumente beinhaltet.

Crawler

Google

Im vorliegenden Kapitel gehen wir auf die Funktionsweise von Suchmaschinen und die von ihnen gebotenen Möglichkeiten ein. Dabei wird als kanonisches Beispiel die Suchmaschine *Google* (vgl. [GGL]) verwendet; diese bietet sowohl die im Folgenden dargestellten typischen Funktionen als auch einige weiterführende Funktionen, die andere Suchmaschinen nicht oder nicht in gleicher Qualität anbieten.

9.2 Aufbau einer Suchmaschine

Eine Suchmaschine besteht aus mehreren Komponenten (vgl. Abbildung 9.1); erst deren Zusammenspiel ermöglicht es, dem Nutzer die gewünschte Funktionalität anzubieten. Übliche Bestandteile einer Suchmaschine sind:

Index

- **Index**

  Der Index stellt die zentrale Komponente einer Suchmaschine dar. In ihm sind alle Informationen über indexierte Dokumente abgelegt. Sowohl die Query Engine als auch der Indexierer greifen auf den Index zu. Der Index wird bei den meisten Suchmaschinen in Form invertierter Listen verwaltet.

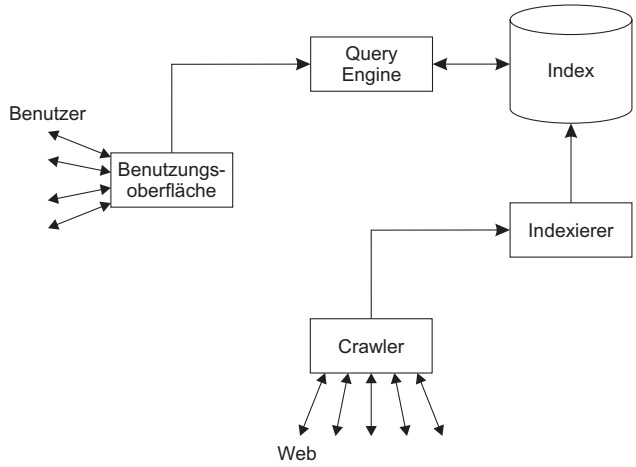


Abbildung 9.1 — Komponenten einer Suchmaschine

- **Indexierer** *Indexierer*

  Der Indexierer ist für die Featureextraktion aus Dokumenten zuständig, die er vom Crawler bezieht. Die extrahierten Features fügt er dann in den Index der Suchmaschine ein, damit sie für die Suche nach relevanten Dokumenten zur Verfügung stehen. Der Indexierer ist damit z.B. für Fragen der Stoppworteliminierung oder der Stammformreduktion zuständig.

- **Crawler** *Crawler*

  Der Crawler ist die Komponente einer Suchmaschine, welche die Dokumente aus dem WWW beschafft. Zu diesem Zweck »crawlt« die auch *Spider* oder *Robot* genannte Komponente – ausgehend von einer Menge von Startseiten (*seed set*) – durch das WWW. Aus HTML-Dokumenten extrahiert der Crawler dabei Hyperlinks, die ihn zu weiteren Dokumenten führen. Auf diese Weise werden nach und nach große Teile des WWW vom Crawler durchwandert.

- **Query Engine** *Query Engine*

  In der Query Engine findet die Anfrageverarbeitung der Suchmaschine statt. Suchanfragen, die der Nutzer über die Benutzungsoberfläche an die Suchmaschine übergeben hat, werden von der Query Engine mit Verfahren des Information Retrieval bearbeitet.

Dabei nutzt die Query Engine den Index. URLs relevanter Dokumente werden an die Benutzungsoberfläche zurückgegeben.

- **Benutzungsoberfläche**

  Die Benutzungsoberfläche dient zur Interaktion des Nutzers mit der Suchmaschine. Über die Benutzungsoberfläche kann der Nutzer seinen Informationswunsch angeben und die ihm von der Suchmaschine als relevant zurückgelieferten Dokumente bzw. die Hyperlinks zu diesen Dokumenten begutachten.

9.3 Begriffsdefinitionen

Obwohl viele der in diesem Kapitel verwendeten Fachbegriffe schon zum Alltagsvokabular gehören, soll ihre Bedeutung an dieser Stelle doch noch einmal erklärt werden, um ein eindeutiges Begriffsverständnis zu gewährleisten:

9.3.1 Internet, Intranet und Extranet

Internet

Das Internet wurde Anfang der 1970er Jahre in den USA entwickelt und hatte vor allem zwei Zielrichtungen:

- Teure Ressourcen wie etwa Großrechner sollten nicht nur den Forschern einer Einrichtung zur Verfügung stehen, sondern auch Mitarbeitern an entfernten Instituten. Durch die Anbindung der Großrechner an einen (damals nur) die USA umspannenden Rechnerverbund konnten diese Ressourcen optimal genutzt werden.

DARPA

- Die Advanced Research Projects Agency ARPA (ARPA und *DARPA* (letzter Abruf: 18.12.2006) bezeichnen die gleiche Organisation; der Originalname war ARPA, sie wurde 1972 in DARPA (Defense Advanced Research Projects Agency) umbenannt, dann 1993 zurück in ARPA und ab 1996 wieder in DARPA.) wollte mit dem Internet erproben, in wieweit sich Rechner- und Kommunikationsnetze derart redundant auslegen lassen, dass sie auch einem massiven Angriff mit Atomwaffen widerstehen können. Daher

ist die Kommunikationsstruktur im Internet auch mehrfach redundant. Für eine Kommunikation zwischen zwei Rechnern stehen im Allgemeinen vielfache Routen zur Verfügung, ein Router wählt nach geeigneten Kriterien (etwa der Netzlast oder den Übertragungskosten) die optimale Route zur Übertragung der Nachrichten.

Das Internet bietet den Nutzern viele Dienste an. Zu Beginn waren dies vor allem *electronic mail* (eMail) und das *File Transfer Protocol* (FTP) zur Übertragung von Dateien zwischen Rechnern. Inzwischen sind noch weitere Dienste hinzugekommen: als Medium zur asynchronen, verteilten Diskussion stehen etwa Newsgroups zur Verfügung. Für die synchrone Kommunikation gibt es beispielsweise den IRC (Internet Relay Chat).

electronic mail, eMail

File Transfer Protocol, FTP

Newsgroups

Internet Relay Chat, IRC

Als »Killer Application« für das Internet hat sich allerdings das World Wide Web (WWW) erwiesen. Das WWW wurde 1989/1990 von *Tim Berners-Lee* am europäischen Kernforschungszentrum *CERN* in Genf entwickelt. Zu Weihnachten 1990 wurde die erste Version des WWW freigegeben. Das WWW bietet dem Benutzer eine (inzwischen) graphisch orientierte Oberfläche zur Darstellung von Hypertext und zur Navigation von Dokument zu Dokument.

World Wide Web, WWW

Inzwischen stellt das Internet bzw. das WWW in vielen Bereichen *das* Kommunikations-, Informations- und Recherchemedium dar. Seit seiner Einführung im Jahr 1990 ist das WWW in nahezu exponentieller Geschwindigkeit gewachsen. Zur Zeit umfasst das WWW nach Schätzungen einige Milliarden Dokumente. Jeden Tag kommen ca. eine Million neue Dokumente hinzu.

Hierbei stellen sich auch schon zwei erste Fragen:

- ***Was* ist ein Dokument?** Sollen nur Textdokumente wie etwa HTML oder Plain Text als Dokumente betrachtet werden, oder sind auch Bilder, Filme, Audio-Dateien, Präsentationen und interaktive Inhalte »Dokumente«?
- **Was ist *ein* Dokument?** Wie im nächsten Abschnitt zum Thema *Hypertext* noch behandelt wird, gibt es gerade bei vielen Dokumenten im WWW das Phänomen, dass sie aus anderen Dokumenten zusammengesetzt sind. Soll man diese Bestandteile als eigenständige Dokumente zählen oder aber nur das Verbunddokument als selbständige Entität begreifen?

Bei unseren weiteren Ausführungen werden wir uns im Hinblick auf das Internet insbesondere auf das WWW beziehen.

Intranet

Intranet

Ein Intranet ist ein abgegrenztes Netz innerhalb einer Organisation, das von außerhalb der Organisation nicht oder nur nach Authentisierung zugänglich ist. Ein Intranet dient dabei vor allem der Information, Kommunikation und der computer-gestützten Kooperation für die Mitglieder innerhalb einer Organisation.

Extranet

Extranet

Ein Extranet ist die Präsentation einer Organisation gegenüber speziellen Zielgruppen: Außenstellen, Tochterfirmen, Vertriebsspartnern, Lieferanten, anderen Unternehmen, Kunden, etc., die zu einem Kreis autorisierter Personen/Unternehmen gehören. Inhalte im Extranet sind somit nicht allgemein zugänglich. Nur gezielte Nutzer erhalten exklusiv Zugang zu Daten/Informationen und Funktionen des Extranet. Ein Extranet kann damit auch für eine exklusive Verbindung unternehmensinterner Computernetze (Intranets) genutzt werden um B2B-Kommunikation (Business-To-Business) zwischen Unternehmen zu erlauben.

9.3.2 Hypertext

Hypertext-Dokumente (vgl. Abbildung 9.2) unterscheiden sich von »normalen« Dokumenten vor allem durch die folgenden beiden Charakteristika:

- Hypertext-Dokumente können eingebettete Dokumente wie etwa Bilder, Sound, interaktive Inhalte oder auch andere Textdokumente enthalten. Dadurch werden Hypertext-Dokumente zu Verbunddokumenten.
- Hypertext-Dokumente können Hyperlinks zu anderen Dokumenten enthalten. Auf einem Hyperlink ist eine Navigations-Operation definiert, die den Benutzer im Allgemeinen zu einem anderen Dokument führt. Hierdurch ist die Vernetzung von Dokumenten möglich.

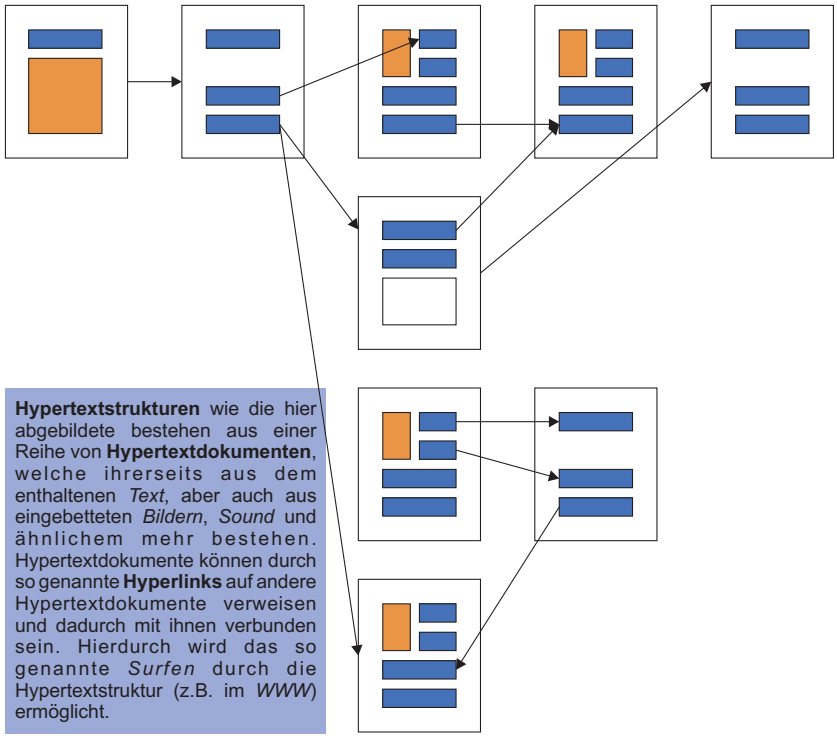


Abbildung 9.2 — Aufbau einer Hypertextstruktur.

Hypertext Markup Language, HTML

Das WWW enthält vornehmlich Hypertext-Dokumente in Form von **HTML-(Hypertext Markup Language)-Dokumenten**. HTML bietet Möglichkeiten zur Formatierung von Dokumenten. So können Überschriften, Listen und Tabellen erstellt werden, Dokumente können Meta-Informationen (etwa Angaben zu Autor, Erstellungs- und Verfallsdatum, Schlüsselwörter, ...) enthalten. Es existieren Konstrukte zur Erstellung von Hyperlinks zu anderen Dokumenten und schließlich können HTML-Dokumente andere (HTML-)Dokumente als eingebettete Dokumente enthalten. HTML stellt somit den Quasi-Standard zur Publikation im Internet dar. HTML-Dokumente sind dabei per se statische Dokumente, können also keine dynamischen Elemente enthalten. Allerdings existieren Technologien, die dynamisch (etwa auf Basis von Informationen in Datenbanken) HTML-Dokumente erstellen können.

9.3.3 Statische und dynamische Dokumente

Zu Beginn des WWW gab es nur statische Dokumente. Ein Beispiel hierfür sind die im vorangegangenen Abschnitt behandelten HTML-Dokumente. Die Menge der Dokumente war zu dieser Zeit eindeutig, abzählbar und endlich. Zwischen den Dokumenten bestand ein zwar chaotisches, aber eindeutiges Beziehungsgeflecht.

Problematisch an statischen Dokumenten ist allerdings, dass ein verhältnismäßig hoher Aufwand betrieben werden muss, damit die statischen Seiten auf aktuellem Stand bleiben. Änderungen an statischen Seiten sind immer mit der Editierung ihres (HTML-)Quellcodes verbunden.

Vorteilhaft an statischen Dokumenten ist hingegen, dass die Anforderungen an einen WWW-Server bei der Auslieferung statischer Dokumente relativ gering sind. Fordert ein Client ein statisches Dokument bei einem Server an, so lädt der Server – nach eventueller vorheriger Prüfung der Zugriffsrechte des Clients – das Dokument vom Hintergrundspeicher und überträgt es zum Client. Das statische Dokument liegt bereits in seiner endgültigen Form vor und muss vom Server nicht nochmals bearbeitet werden. Solche Server skalieren sehr gut; schon auf Basis handelsüblicher PC-Hardware lassen sich WWW-Server realisieren, die bis zu einige Dutzend Zugriffe pro Sekunde bearbeiten können.

Hypertext Transfer Protocol, HTTP

Grundlage des Dokumentenaustauschs zwischen Server und Client ist das **Hypertext Transfer Protocol (HTTP)**, wie es in Abbildung 9.3 schematisch dargestellt wird.

Es wurden eine Reihe von Technologien entwickelt, welche in der Lage sind, dynamisch Inhalte zu erzeugen. Die Dokumente, die ein Nutzer in diesem Fall anfordert, liegen zum Zeitpunkt dieser Anforderung noch nicht vor, sondern werden dynamisch (quasi *on demand*) erstellt. Dabei können Berechnungen, Datenbankabfragen, nutzerabhängige Aktionen und andere Kriterien und Daten in das erstellte dynamische Dokument einfließen.

Mit dynamischen Dokumenten ist es möglich, hochaktuelle Informationen zu liefern (etwa in Buchungs- und Handelssystemen), Inhalte zu transformieren (z.B. dpa-Meldungen optisch aufzubereiten) und personalisierte Inhalte und Portale zu schaffen.

Java Server Pages, JSP
Servlets

Wie bereits erwähnt, existiert eine Vielzahl von (zum Teil konkurrierenden) Technologien, welche die Erstellung dynamischer Inhalte ermöglichen. Als Beispiel seien **Java Server Pages** (JSP), **Servlets**,

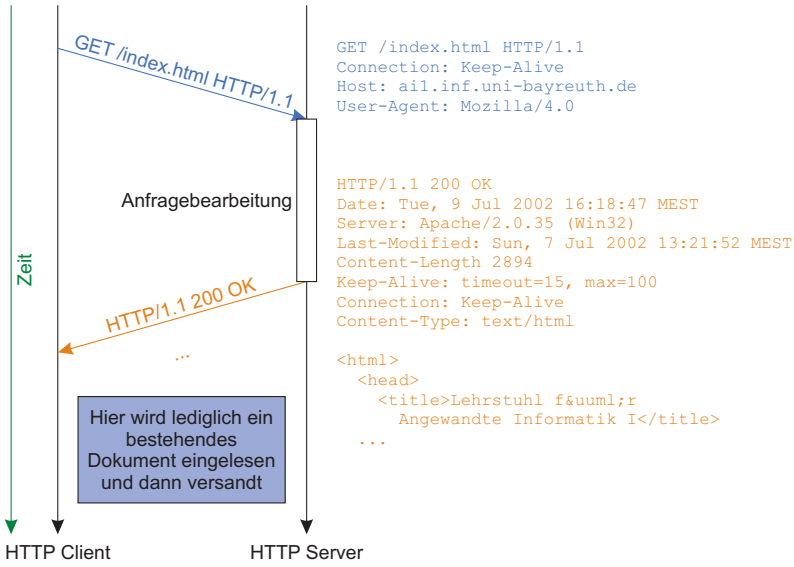


Abbildung 9.3 — Ablauf eines HTTP-Requests, bei dem der Client beim WWW-Server nach einem Dokument nachfragt und dieses dann ausgeliefert bekommt.

das **Common Gateway Interface** (CGI) und **Active Server Pages** (ASP) genannt.

Common Gateway Interface, CGI

Active Server Pages, ASP

Die Auslieferung dynamischer Dokumente stellt höhere Anforderungen an den Server:

- Fordert ein Client ein dynamisches Dokument an, so wird üblicherweise eine Vorlage für dieses Dokument vom Hintergrundspeicher geladen.
- Diese Vorlage enthält Anweisungen (etwa für Datenbankanfragen), welche vom Server ausgeführt werden.
- Liegen die Ergebnisse der einzelnen dynamischen Elemente vor, kann der Server daraus das fertige dynamische Dokument zusammenstellen.
- Dieses wird dann an den Client verschickt.

Um trotz des hohen Aufwands eine angemessene Leistung bzw. einen angemessenen Durchsatz zu erreichen, wird häufig ein mehrstufiger Ansatz verfolgt:

- Ein Server nimmt die Client-Anforderungen entgegen und sendet die Datenbankanfragen u.ä. an dafür spezialisierte Server, die auf einem dedizierten Computer laufen.
- Diese spezialisierten Server übernehmen dann die Auswertung der dynamischen Anweisungen und senden die Ergebnisse an den ersten Server zurück.
- Dieser setzt die Ergebnisse unter Einbeziehung der Dokument-Vorlage zusammen und sendet das Ergebnis an den Client zurück.

Abbildung 9.4 verdeutlicht beispielhaft die Bearbeitung der Anfrage nach einem dynamischen Dokument.

9.3.4 Deep Web und Shallow Web

Shallow Web

Unter *shallow web* versteht man denjenigen Teil des WWW, den man über Hyperlinks erreichen kann. Dabei umfasst das Shallow Web sowohl statische als auch dynamische Dokumente. Nach Schätzungen gibt es momentan mehrere Milliarden Dokumente im Shallow Web. Allerdings dürfte ein gewisser Teil davon aus Duplikaten bestehen. Diese können etwa dadurch entstehen, dass beispielsweise eine Universität für ihre Studenten Teile der Java-API-Dokumentation auf dem WWW-Server der Universität spiegelt.

Deep Web

Das *deep web* hingegen ist nicht über einfache Hyperlinks erreichbar. Um zu Dokumenten des Deep Web zu gelangen müssen z.B. Formulare ausgefüllt werden. Dies kann verschiedene Gründe haben. Immer mehr Dokumente sind in Datenbanken abgelegt, ein Beispiel hierfür sind Nachrichtenagenturen. Häufig gibt es nicht zu allen Objekten in einer Datenbank eine passende Anfrageseite, auf die man über das WWW zugreifen kann. Das *deep web* wächst inzwischen wesentlich schneller als das *shallow web*.

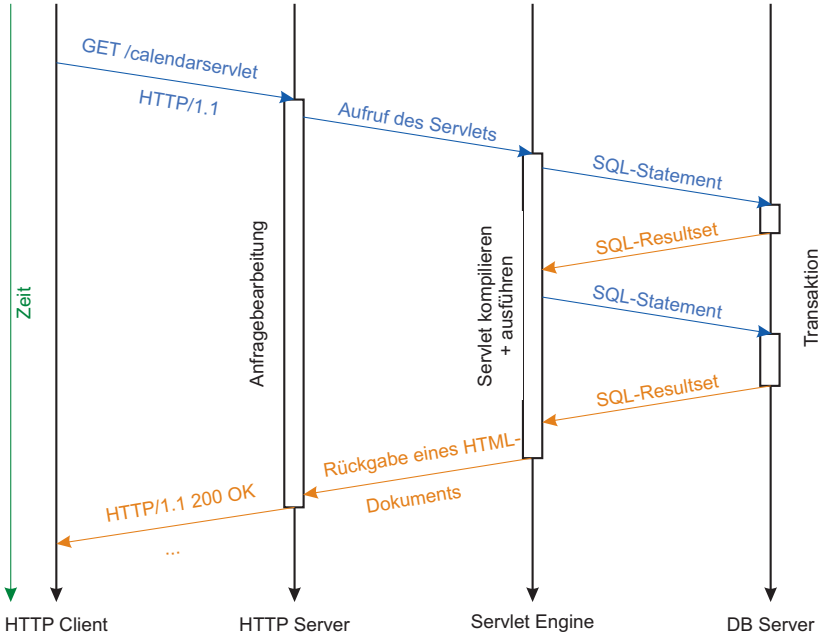


Abbildung 9.4 — Erstellung eines dynamischen Dokuments nach einem dreistufigen Ansatz.

9.3.5 Breiten- und Tiefensuche

Bei der systematischen Durchsuchung des Internet durch einen Crawler werden in bereits bekannten Dokumenten enthaltene Hyperlinks verfolgt, die dann zu eventuell noch nicht bekannten Dokumenten führen, die neu in die Menge der bekannten Dokumente aufgenommen werden können.

Man kann das Geflecht der Beziehungen zwischen Dokumenten im WWW auch als Graph betrachten: die einzelnen Dokumente des WWW sind dabei die Knoten dieses Graphen, die Hyperlinks zwischen den Dokumenten stellen die Kanten dar. Es handelt sich bei diesem Graphen um einen gerichteten Graphen, der mit Sicherheit Zyklen enthält.

Der Vorgang der Durchsuchung des WWW nach Dokumenten kann somit auch als Problem der Graphtraversierung aufgefasst werden. Man möchte dabei idealerweise alle Dokumente des WWW einmal »besu-

chen«. Für dieses Problem gibt es zwei grundsätzliche Strategien, welche die Reihenfolge der Seitenbesuche festlegen:

Tiefensuche

- **Tiefensuche** Jeder besuchte Knoten wird auf Links zu weiteren Knoten untersucht. Wird ein Link gefunden, so wird dieser auf einen Stapel, der nach dem last in first out–Prinzip (LIFO) arbeitet, abgelegt. Ist der aktuelle Knoten vollständig analysiert, dann wird als nächstes der Knoten untersucht, der über den obersten Link auf dem Stapel erreicht wird, welcher also als letzter eingefügt wurde.

Breitensuche

- **Breitensuche** Jeder besuchte Knoten wird auf Links zu weiteren Knoten untersucht. Neu gefundene Links werden am Ende einer Warteschlange eingereiht (first in first out-Prinzip, FIFO). Ist der aktuelle Knoten vollständig analysiert, so wird als nächstes der Knoten untersucht, der über den Link am Kopf der Warteschlange erreicht wird, dessen Einfügezeitpunkt also am längsten zurückliegt.

Die Abbildungen 9.5 und 9.6 zeigen, welche Knoten eines Baumes bei der Traversierung nach Tiefe bzw. Breite nach jeweils 20 Schritten besucht wurden

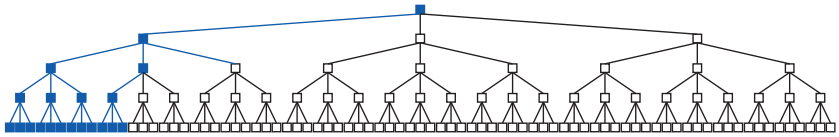


Abbildung 9.5 — Tiefensuche nach 20 Schritten

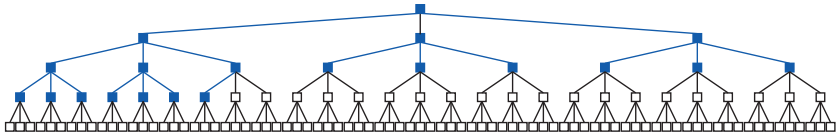


Abbildung 9.6 — Breitensuche nach 20 Schritten

9.3.6 Crawler, Spider, Robot

Crawler

Der Begriff des *Crawlers* ist in der Informatik nicht eindeutig festgelegt.

Wir versuchen uns daher hier an einer Begriffsdefinition:

> »Ein *Crawler* ist ein Programm, welches
>
> - ausgehend von einer nicht-leeren Menge von Start-URLs (WWW- und/oder Filesystem-URLs)
> - das unter der jeweiligen URL erreichbare Dokument liest und
> - dieses Dokument nach Hyperlinks zu weiteren Dokumenten durchsucht.
>
> Zusätzlich kann auf jedem besuchten Dokument eine Operation durchgeführt werden; beispielsweise kann das Dokument indexiert und in die Datenbasis einer Suchmaschine aufgenommen werden.«

Der Name *Crawler* stammt aus dem Englischen (to crawl - kriechen) und bezieht sich darauf, dass ein Crawler systematisch durch den gegebenen URL-Raum »kriecht«. Synonym zum Namen Crawler können im Kontext der Indexierung des WWW auch die Begriffe

- *Spider* (Anspielung auf die Bewegung des Crawlers »wie eine Spinne im Netz des Webgraphen«) und *Spider*
- *Robot* (wegen der automatischen Traversierung und Indexierung) *Robot*

verwendet werden.

Crawler verwenden bei ihrer Suche durch das Netz fast immer das Konzept der Breitensuche. Auf diese Weise werden Dokumente in der Reihenfolge indexiert, in der sie per Hyperlink entdeckt werden. Der vom Crawler verursachte Datenverkehr wird dabei zumindest idealerweise annähernd gleich auf verschiedene WWW-Server verteilt. Crawler verursachen nach diversen Schätzungen etwa 5-15% des gesamten Datenverkehrs im Internet; die gleichmäßige Verteilung dieses Datenverkehrs ist also durchaus wichtig.

9.4 Techniken des Crawling

Im vorliegenden Abschnitt wird genauer auf die Funktionsweise von Crawlern eingegangen, und es werden Techniken und Strategien dargestellt, die von Crawlern eingesetzt werden. Dies geschieht in den folgenden Abschnitten:

9.4.1 Funktionsweise eines Crawlers

9.4.2 Der Robots Exclusion Standard

9.4.3 Indexierung von HTML-Dokumenten

9.4.4 Indexierung anderer Dokumenttypen

9.4.5 Prioritätsgeführtes Crawling

9.4.6 Probleme beim Crawlen

9.4.1 Funktionsweise eines Crawlers

Beim Start eines Crawlers wird diesem eine nicht-leere Menge von Start-URLs vorgegeben, die (im Falle eines WWW-Crawlers) auf Dokumente im WWW verweisen. Der Crawler fordert diese Dokumente von den entsprechenden Servern an und analysiert sie.

Bei dieser Analyse extrahiert der Crawler alle in dem aktuellen Dokument enthaltenen Hyperlinks zu weiteren Dokumenten im WWW. Diese Links nimmt der Crawler in eine persistierbare Warteschlange auf, in der alle bisher gefundenen, aber noch nicht besuchten Links/URLs verwaltet werden. Zusätzlich indexiert der Crawler in aller Regel die besuchten Dokumente und baut aus diesen Informationen einen Index auf, der später von einer Suchmaschine genutzt werden kann.

Durch das beschriebene Verhalten durchsucht der Crawler also – jedenfalls in der Theorie – das gesamte WWW und indexiert alle Dokumente. In regelmäßigen (etwa alle vier Wochen) oder unregelmäßigen (etwa nach einer bestimmten Anzahl indexierter Dokumente) Abständen wird der Crawler angehalten. Der bis dahin erstellte Index wird zur Suchmaschine kopiert und ersetzt oder aktualisiert den dort vorhandenen Index. Ab diesem Zeitpunkt können Benutzer der Suchmaschine auf den neuen Index zurückgreifen und auf den neu indexierten Dokumenten suchen. Aus

diesem Grund hinken Suchmaschinen dem aktuellen Stand des WWW auch immer mit einer gewissen Verzögerung (einige Tage bis einige Monate) hinterher. Nach der Übertragung des Index wird der Crawler von neuem gestartet.

Betreiber von Websites können ihre Website bei einem Crawler anmelden und diesen dadurch zur Indexierung auffordern (siehe z.B. die entsprechende Seite bei Google, letzter Abruf: 18.12.2006). Dies führt dazu, dass die Website indexiert wird, bevor der Crawler sie über Hyperlinks von anderen Websites erreicht. Auch Websites, auf die noch gar nicht verwiesen wird, können so indexiert werden. Die Liste der neu angemeldeten Websites wird üblicherweise an den Crawler übertragen, wenn dieser angehalten wurde; einige Crawler können auch während eines Crawls neue URLs in ihre Warteschlangen aufnehmen. Zwischen der Anmeldung einer Website und ihrer erstmaligen Indexierung durch einen Crawler können daher von einigen Tage bis zu einigen Monaten vergehen.

9.4.2 Der Robots Exclusion Standard

Crawler besuchen standardmäßig alle über Hyperlinks erreichbaren Dokumente einer Website. Aus verschiedenen Gründen ist dies nicht immer erwünscht:

- Manche Administratoren möchten etwa den von Crawlern verursachten Datenverkehr auf der eigenen Website minimieren.
- Manche Crawler haben/hatten die schlechte Angewohnheit, während der Indexierung einen Server zu »hammern«, also alle paar Zehntel- oder gar Hundertstelsekunden ein weiteres Dokument von diesem Server anzufordern. Dies hat den gleichen Effekt wie eine Denial-of-Service-(DoS)-Attacke: der Server hat mit der Bedienung des Crawlers (bzw. des Angreifers bei einer DoS-Attacke) so viel Last, dass andere Clients entweder gar nicht oder nur sehr schleppend bedient werden können.
- Wünschenswert ist manchmal auch, dass vor allem die Einstiegsseiten einer Website indexiert werden, damit Besucher, welche über eine Suchmaschine die Website finden, auch auf der Einstiegsseite einsteigen und nicht tief unten in der Hierarchie.

```
# Alle Crawler sollen von bestimmten Bereichen der
    Website ausgeschlossen werden:

User-agent: *
Disallow: /cgi-bin/
Disallow: /tmp/
Disallow: /private/

# Ein einzelner Crawler soll ausgeschlossen werden:

User-agent: BadBot
Disallow: /
```

Abbildung 9.7 — Beispiele für den Inhalt einer `robots.txt`-Datei

- Manche Bereiche einer Website werden zudem zwar nicht als geheim eingestuft, sollen aber dennoch nicht per Suchmaschine gefunden werden.

Robots Exclusion Standard

Um den Administratoren von Websites eine Möglichkeit zu geben, das Verhalten von Crawlern auf der eigenen Website zu beeinflussen, wurde der Robots Exclusion Standard geschaffen (letzter Abruf: 12.12.06).

Durch eine Datei namens *robots.txt* im Hauptverzeichnis des WWW-Servers kann das Verhalten der Crawler beeinflusst werden. Beispiele für den Inhalt einer solchen Datei gibt Abbildung 9.7 wieder.

Über den *User-agent* kann ein einzelner Crawler identifiziert werden, ein Stern steht dabei für alle Crawler. Im Folgenden können dann in der *Disallow*-Klausel einzelne Verzeichnisse angegeben werden, die dem entsprechenden Crawler verwehrt bleiben soll. Möchte man allen Crawlern Zugriff auf sämtliche Bereiche einer Website geben, so gibt man als User-agent ein * an und lässt die Disallow-Klausel leer.

Alternativ können diese Angaben auch als *meta*-Tag im Header eines HTML-Dokuments enthalten sein, wie Abbildung 9.8 zeigt. Hierbei wird dann auch noch zwischen den beiden Optionen *INDEX* bzw. *NOINDEX* und *FOLLOW* bzw. *NOFOLLOW* unterschieden. Mit *INDEX* bzw. *NOINDEX* kann die Indexierung des HTML-Dokuments erlaubt bzw. untersagt werden, mit *FOLLOW* bzw. *NOFOLLOW* kann das Verfolgen der in diesem HTML-Dokument enthaltenen Hyperlinks zugelassen bzw. verwehrt werden.

```
<html>
  <head>
    <title>Lehrstuhl f&uuml;r Medieninformatik</title>
    <META NAME="ROBOTS" CONTENT="NOINDEX,NOFOLLOW">
  </head>
  <body>
    <p>Herzlich willkommen auf den Seiten des
       Lehrstuhls f&uuml;r Medieninformatik an der
       Universit&auml;t Bamberg.
  </body>
</html>
```

Abbildung 9.8 — Beispiel für `meta`-Tags im Header einer HTML-Datei. Das hier gezeigte Tag führt dazu, dass ein Crawler dieses HTML-Dokument nicht indexiert und auch keine Hyperlinks auf dieser Seite verfolgt.

Allerdings implementieren nicht alle Crawler diese Funktionalität, so dass es nach wie vor ratsam ist, eine Datei *robots.txt* zu verwenden.

Problematisch an diesem Ansatz ist, dass die Crawler diesen Standard nicht implementieren und sich auch nicht an ihn halten müssen. Nichts hindert einen Crawler daran, trotz einer restriktiven *robots.txt* dennoch – eigentlich »verbotene« – Bereiche einer Website zu indexieren. Allerdings gehen nahezu alle bekannten Crawler mit diesem Standard konform – schon deshalb, weil Administratoren ansonsten mit anderen Mitteln den Zugang der Crawler zur Website sperren könnten, etwa, indem das komplette IP-Subnetz, aus dem der Crawler stammt, für den Zugriff auf die Website gesperrt wird.

9.4.3 Indexierung von HTML-Dokumenten

Hypertext Markup Language, HTML

HTML-Dokumente folgen in Syntax und Grammatik einem festgelegten Standard, nämlich dem HTML-Standard in der aktuellen Version 4.01. Erst dadurch werden sie zu gültigen HTML-Dokumenten. Für die Indexierung eines HTML-Dokuments ist es nun vorteilhaft, wenn man die grammatische Struktur des Dokuments analysiert. So möchte man im Allgemeinen nur diejenigen Teile des Dokuments in den Index aufnehmen, welche der Nutzer am Bildschirm auch sehen kann, also keine Steuerbefehle, Kommentare, ...

HTML-Parser

Zu diesem Zweck verwendet man einen HTML-Parser, der ein HTML-

Dokument einliest und intern die Struktur des Dokuments repräsentiert. Danach kann man daran gehen, gezielt die Teile des Dokuments zu indexieren, an denen man interessiert ist. So kann man beispielsweise nur die Überschriften und den normalen Fließtext eines Dokuments indexieren, nicht aber den in Tabellen und Kommentaren enthaltenen Text.

Ein interessanter Nebeneffekt des Einsatzes eines HTML-Parsers ist, dass der Parser das HTML- Dokument automatisch auch auf die darin enthaltenen Hyperlinks untersucht – schließlich sind die hierfür vorgesehenen Konstrukte Bestandteil der HTML-Grammatik und des HTML-Sprachumfangs. Der Crawler kann also einfach den eingesetzten Parser nach einer Liste aller im Dokument vorkommenden Hyperlinks fragen und die erhaltenen URLs in seine Warteschlange einfügen.

Zur Implementierung eines solchen Crawlers mit integriertem HTML-Parser kann man auf eine Reihe frei verfügbarer HTML-Parser zurückgreifen. Diese findet man vornehmlich für objektorientierte Programmiersprachen wie C++ und Java, aber auch für interpretierte Skriptsprachen wie Perl.

Häufig gibt es einen eigentlichen, vorgefertigten HTML-Parser, welcher die einzelnen Tags aus einem HTML-Dokument extrahiert und einen Syntaxbaum aus dem Dokument ableitet. Zusätzlich existiert dann eine abstrakte Klasse, in der Callback-Methoden definiert sind, die das Verhalten bzw. die Reaktion auf einzelne Tags definiert. Von dieser abstrakten Klasse leitet man dann seine eigene Klasse ab und überschreibt die Callback-Methoden durch eigene Versionen, welche das gewünschte Verhalten implementieren. Hierdurch wird es möglich, beispielsweise nur auf Tags, die einen Hyperlink enthalten, mit einer Indexierungsaktion zu reagieren.

Im Fall der Programmiersprache Java sind die beiden eben besprochenen Klassen bereits in der mitgelieferten Standardbibliothek enthalten:

- `javax.swing.text.html.parser.ParserDelegator` als eigentlicher Parser und
- `javax.swing.text.html.HTMLEditorKit.ParserCallback` als abstrakte Klasse mit den vordefinierten Callback-Methoden für einzelne Tags.

Link Extraction

Von besonderem Interesse bei der Indexierung von HTML-Dokumenten ist die *Link Extraction*. So wird der Vorgang der Extraktion von Hyperlinks aus Dokumenten genannt. Für die eigentliche Extraktion kann das eben beschriebene Verfahren mit der Überschreibung entsprechender Callback-Methoden verwendet werden. In der Callback-Methode, die für Hyperlink-Tags eines HTML-Dokuments aufgerufen wird, muss der vorliegende Hyperlink dann allerdings meist noch nachbearbeitet werden.

Link Extraction

Die Hyperlinks eines Dokuments sind nach der eigentlichen Extraktion durch den Parser eventuell noch nicht in ihrer einfachsten, »kanonischen« Form. Darüber hinaus muss vermieden werden, dass sich der Crawler in unendlichen virtuellen Hierarchien verfängt wie sie z.B. von Content Management Systemen erzeugt werden (vgl. zu Content Management Systemen z.B. contentmanager.de, letzter Abruf: 18.12.2006).

Content Management System, CMS

Kanonische Form von URLs

Ein Beispiel für den Fall, dass eine URL noch nicht in ihrer einfachsten Form ist, zeigt die folgende URL:

Uniform Resource Locator, URL

http://www.spiegel.de/./kultur/../wirtschaft/0,1518,204638,00.html

Solche Formen von URLs können durch den Verlauf des Crawls entstehen, indem Hyperlinks gefolgt wird, die z.B. ein Dokument im übergeordneten Verzeichnis (..) referenzieren. Üblicherweise ersetzen solche *relativen Links* einfach das letzte Element des bisher verfolgten Pfades und haben daher URLs wie die oben abgebildete zur Folge.

Die Überführung einer solchen URL in ihre kanonische Form ist ein verhältnismäßig einfaches Problem:

- *./* verweist auf das aktuelle Verzeichnis auf dem Webserver und kann daher ersatzlos gestrichen werden.
- *../* verweist hingegen auf das dem aktuellen Verzeichnis übergeordnete Verzeichnis. Dies führt dazu, dass neben *../* auch das vorhergehende Element der URL gestrichen wird.

Somit kann folgende Überführung der oben angegebenen URL in ihre kanonische Form vorgenommen werden.

http://www.spiegel.de/./kultur/../wirtschaft/0,1518,204638,00.html

⇓

http://www.spiegel.de/wirtschaft/0,1518,204638,00.html

In der Standardbibliothek der Programmiersprache Java gibt es die Klasse `java.net.URL`, die eine URL repräsentiert und eine kanonische Form dieser URL erzeugen kann.

Unendliche virtuelle Hierarchien

Insbesondere bei dynamisch erstellten Dokumenten kann es vorkommen, dass der erzeugte URL-Raum unendlich ist. So kann beispielsweise an die URL jeder dynamisch erzeugten Seite ein Zeitstempel angehängt werden, etwa *?DATE=20030123092724*. Damit existieren zu jedem solcherart dynamisch erzeugten Dokument unendlich viele Varianten – und jede Sekunde kommt eine weitere Variante hinzu.

Im Vergleich zum vorher betrachteten Problem ist dieses schon wesentlich schwieriger zu lösen. Ein naiver Ansatz könnte etwa darin bestehen, dass der Betreiber eines Crawlers diesen anhält, sobald er – so zu sagen »händisch« – bemerkt, dass dieser sich in einer unendlichen virtuellen Hierarchie befindet. Dem Crawler wird dann eine Regel hinzugefügt, dass er keine Dokumente mehr indexieren soll, die unterhalb des Topverzeichnisses dieser unendlichen virtuellen Hierarchie liegen. Damit wird nun allerdings keine einzige Variante eines dynamisch erzeugten Dokuments dieser Hierarchie mehr indexiert, so dass der Crawler in diesem Bereich mehr oder weniger »blind« ist.

Ein komplexerer Ansatz verfolgt eine aufwändigere Strategie: soll eine neue URL zur Menge der noch zu durchsuchenden URLs hinzugefügt werden, so wird zunächst geprüft, ob es schon »sehr ähnliche« URLs gibt, die sich etwa nur durch einen Zeitstempel voneinander unterscheiden. Unterscheiden sich dann auch noch die von diesen sehr ähnlichen URLs referenzierten Dokumente nicht oder nur marginal (etwa durch eine angezeigte Uhrzeit), werden die URLs als verschiedene Varianten ein und derselben URL aufgefasst. Es wird dann nur eine Variante indexiert.

9.4.4 Indexierung anderer Dokumenttypen

Für die Indexierung eines Dokuments, welches einem gewissen Dokumenttyp entspricht und dessen Syntax und Grammatik folgt, gibt es grundsätzlich zwei verschiedene Ansätze:

- Wie im Falle von HTML bereits angesprochen, kann man einen Parser entwickeln, welcher die Syntax und Grammatik des jeweiligen Dokumenttyps versteht und einen Syntaxbaum aus dem analysierten Dokument ableitet. Auf diesem Syntaxbaum kann man dann operieren und etwa einzelne Tags oder Bereiche des Dokuments für die Indexierung heranziehen.

  Parser

  Für einige Dokumenttypen existieren solche Parser, teilweise sind sie auch frei verfügbar. Falls für einen bestimmten Dokumenttyp ein solcher Parser nicht existiert oder aufgrund eines beschränkten Budgets nicht beschafft werden kann, muss ein eigener Parser für diesen Dokumenttyp entwickelt werden. Je nach Komplexität von Syntax und Grammatik der jeweiligen Sprache kann dies unterschiedlich aufwändig sein. In der Regel ist die Erstellung eines korrekten und performanten Parsers für eine Sprache ein nichttriviales Problem.

- Alternativ kann ein Dokument auch durch einen *Filter* geschickt werden. Dieser Filter transformiert das Dokument dann aus dem Originalformat in ein anderes, meist einfacher zu parsendes Format. Häufig ist dieses Zielformat *Plain Text*.

  Filter

  Plain Text

Besonders wünschenswert für einen Crawler ist es, wenn er neben HTML auch XML, PDF, PS, diverse Office-Formate und Plain Text indexieren kann. Dabei ist die Indexierung von Plain Text am simpelsten: Plain Text muss nicht geparst werden, sondern kann direkt indexiert werden. Daher werden komplexere Formate vor der Indexierung auch häufig in Plain Text überführt; dabei geht zwar die Struktur-Information (Überschriften, Fettdruck, Tabellen, ...) des Originalformats verloren – allerdings ist man meist ohnehin eher an den im Dokument verwendeten Begriffen und nicht so sehr an der logischen Struktur interessiert.

XML (eXtensible Markup Language) verbreitet sich immer mehr als ein allgemeines Format zur Ablage von Daten und Dokumenten. Dabei können XML-Parser ebenso leicht implementiert werden wie HTML-Parser;

eXtensible Markup Language, XML

für Java etwa existieren gleich mehrere konkurrierende Packages, die XML-Parser anbieten.

Portable Document Format, PDF

PDF (Portable Document Format) ist ein sehr verbreitetes Format zum plattformunabhängigen Austausch von Dokumenten. Hierfür existieren zum einen Parser, welche die Syntax und Grammatik eines PDF-Dokuments analysieren können. Wesentlich einfacher ist aber die Konvertierung nach Plain Text und die Indexierung dieses transformierten Dokuments.

PostScript, PS

PS (PostScript) ist ein Format, welches insbesondere in der Druckvorstufe und bei wissenschaftlichen Publikationen eine Rolle spielt. Mit PS-Dokumenten kann ebenso wie mit PDF-Dokumenten verfahren werden.

Für diverse Office-Formate existieren ebenfalls Plain Text-Filter, welche ein gegebenes Office-Dokument unter Verlust der Formatierung in eine Plain Text-Repräsentation überführen, welche dann indexiert werden kann.

9.4.5 Prioritätsgeführtes Crawling

Einige Websites sind für Benutzer besonders interessant. Hierzu gehören neben Online-Magazinen auch die Online-Ausgaben von Printmedien, Wirtschaftsdienste u.ä. Diese Websites zeichnen sich üblicherweise durch eine hohe Aktualisierungsrate aus. Sie ändern ihren Inhalt sehr häufig, teilweise sogar minütlich oder stündlich.

Daher ist solchen Websites auch seitens eines Crawlers erhöhte Aufmerksamkeit zu schenken. Der Crawler überprüft in bestimmten Intervallen, ob sich ein bereits indexiertes Dokument seit dem letzten Besuch geändert hat. Falls dies so ist, wird das Intervall zwischen zwei Besuchen für dieses Dokument verringert (außer es unterschreitet bereits ein Mindestmaß). Hat sich das Dokument hingegen nicht verändert, so wird das Besuchsintervall für dieses Dokument belassen oder sogar erhöht.

Durch dieses Vorgehen kann erreicht werden, dass ein Crawler Seiten auch in Abhängigkeit ihrer Aktualisierungsrate wiederholt besucht und indexiert. Durch die Einstellung des Besuchsintervalls bleibt der Index also auch für solche Seiten aktuell, die sich häufig ändern.

9.4.6 Probleme beim Crawlen

Beim Crawlen durch das WWW können sich einige Probleme ergeben, die an dieser Stelle angesprochen werden sollen. Dies sind vor allem (aber nicht ausschließlich):

9.4.6 Frames

9.4.6 JavaScript und andere Scriptelemente

9.4.6 Imagemaps

Frames

Frame

Ein besonderes Problem beim Crawling im WWW stellen Dokumente und Websites dar, die Frames verwenden. Frames sind Konstruktionselemente, mit denen der Anzeigebereich eines WWW-Browsers in mehrere Bereiche (die »Frames«) aufgeteilt werden kann. In einem speziellen Dokument werden die Definitionen der Frames vorgenommen, dabei wird auch jedem einzelnen Frame ein Dokument zugewiesen, welches in diesem Frame angezeigt werden soll. Es kann also in jedem Frame unabhängig von den anderen Frames ein anderes Dokument geladen und angezeigt werden. Als zusätzliche Erschwernis kommt hinzu, dass der Klick des Besuchers auf einen Hyperlink in einem Frame das Laden des referenzierten Dokuments in einen anderen Frame auslösen kann.

Für einen Crawler ist es nun im Allgemeinen kein Problem, aus der Definition der Frames die URLs der einzelnen Dokumente abzuleiten und diese dann einzeln zu indexieren. Das eigentliche Problem ergibt sich bei der Suche nach Dokumenten.

Wenn der Crawler nämlich ein Dokument indexiert, das eigentlich in einem Frame liegt, so wird im Index die URL zum Dokument und nicht zu dem die Frames definierenden Dokument abgelegt. Ist dieses Dokument nun in einem Suchergebnis enthalten, so führt die dem Suchenden angebotene URL zum »nackten« Dokument ohne die das Dokument eigentlich umgebende Framestruktur. Dies ist fast immer unerwünscht und kann den Nutzen des Dokuments stark einschränken, wenn etwa erst durch die Framestruktur ein Kontext gegeben wird, der die Einordnung des Dokuments ermöglicht.

Ein naiver Ansatz auf Seiten des Crawlers besteht darin, nicht die URL des indexierten Dokuments im Index abzulegen, sondern die URL der

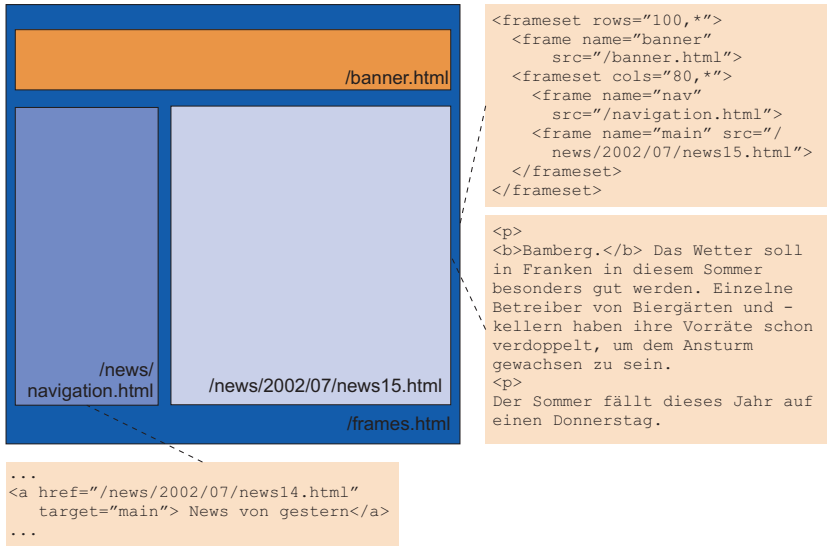


Abbildung 9.9 — Beispiel für ein Frameset.

umgebenden Framestruktur. Dies funktioniert allerdings nur so lange, wie es innerhalb der Framestruktur für jeden Frame nur ein Dokument gibt. Sobald aber ein »Navigationsframe« existiert, welcher Hyperlinks enthält, die verschiedene Dokumente im »Hauptframe« anzeigen, versagt dieser Ansatz.

Die Lösung dieses Problems kann zur Zeit nicht auf Seiten des Crawlers und der Suchmaschine erfolgen. Statt dessen sollten Webdesigner ein kurzes JavaScript in ihre Dokumente einfügen: dieses fragt bei einem Aufruf der Seite ab, ob das Dokument alleine oder in der Framestruktur angezeigt wird; bei Bedarf wird dann die Framestruktur nachgeladen und angezeigt.

Obwohl dieses Vorgehen sehr einfach und gleichzeitig sehr effektiv ist, wird es leider nicht immer angewendet. Viele Designer sind sich der beschriebenen Probleme nicht bewusst oder nehmen sie als Preis für ein »schöneres« Design der Websites in Kauf. Einige Designer allerdings gehen auch andere Wege:

Komplexe Dokumente werden häufig nicht mit Frames, sondern über Design-Tabellen entworfen. Da alle Bestandteile dieser Tabellen und damit auch des Dokuments innerhalb ein und desselben HTML-Dokuments

liegen, wird so die oben geschilderte Problematik umgangen. Das visuelle Ergebnis für den Benutzer einer Website ist mindestens genau so gut wie mit Frames, eher sogar noch besser, da sich mit Design-Tabellen Dokumente noch besser layouten lassen.

JavaScript und andere Scriptelemente

JavaScript

Durch das Einbinden kurzer JavaScripts (zu JavaScript siehe z.B. http://de.selfhtml.org/javascript/index.htm, letzter Abruf 18.12.2006) können Autoren ihre Dokumente aufwerten und Zusatznutzen anbieten. Durch JavaScripts lassen sich auch einige Aufgaben automatisieren: so kann beispielsweise die Erzeugung von Hyperlinks zu vorangegangenen und nachfolgenden Dokumenten sowie zu übergeordneten Hierarchiestufen per JavaScript gelöst werden.

Dies führt aber auch zu Problemen: die so erzeugten Links werden erst durch die JavaScript-Laufzeit-Umgebung im Browser des Clients erzeugt und in das Dokument geschrieben. Sie sind also im eigentlichen Dokument (noch) nicht vorhanden. Die meisten Crawler verfügen jedoch nicht über die Fähigkeit JavaScripts auszuführen und können solchen Links daher auch nicht folgen. (Das Ausführen aller JavaScripts würde dabei auch ein Effizienz-Problem für den Crawler bedeuten.)

Sitemap

Eine Lösung dieses Problems muss wiederum auf Seiten der Autoren der Website gesucht werden. Ein relativ simpler und doch effektiver Ansatz besteht in der Erstellung einer Sitemap durch die Autoren. Sitemaps sind dabei spezielle HTML-Dokumente, die in einer (eventuell kategorisierten) Auflistung alle Dokumente einer Website aufführen und verlinken; dabei ist die Sitemap selbst über normale Hyperlinks zu erreichen. Die Erstellung solcher Sitemaps kann automatisch erfolgen, etwa indem ein spezielles Tool im Filesystem des WWW-Servers alle HTML-Dokumente unterhalb des Wurzelverzeichnisses des WWW-Servers in die Sitemap aufnimmt. Damit kann ein Crawler dann alle Dokumente einer Website erreichen, selbst wenn der »normale« Benutzer diese über dynamisch per JavaScript erzeugte Links ansteuern würde.

Imagemaps

Imagemap

Ein ähnlich gelagertes Problem stellen so genannte Imagemaps dar. Diese sind Grafiken, die an verschiedenen Stellen für Mausklicks sensibel sind und mit der Ausführung einer Operation reagieren. Eine solche

Operation ist typischerweise die Weiterleitung des Nutzers zu einem anderen Dokument. Man unterscheidet des Weiteren zwischen server-side und client-side Imagemaps; bei diesen wird server- bzw. client-seitig entschieden, wie auf einen Klick des Nutzers auf eine bestimmte Stelle der Imagemap reagiert werden soll.

Heutige Crawler können Hyperlinks in Imagemaps zum Teil nicht verfolgen. Die Lösung dieses Problems geschieht analog zum Vorgehen bei JavaScript. Wiederum erstellen die Autoren der Website eine Sitemap – wenn diese nicht bereits wegen der JavaScript-Problematik vorliegt. Hierdurch wird sichergestellt, dass Crawler alle Dokumente einer Website über normale Hyperlinks erreichen können.

9.5 Typische Funktionen einer Suchmaschine

Nachdem wir uns bisher ausführlich mit der Crawl-Komponente einer Suchmaschine beschäftigt haben, kommen wir nun zur Suchfunktionalität auf dem durch das Crawling erstellten Index. Suchmaschinen bieten hierzu eine Reihe von Standardfunktionen an. Nahezu jede aktuelle Suchmaschine verfügt über diese Funktionen. Zu beachten ist, dass sich mit der Zeit die Ansicht darüber, was man als »Standard« bezeichnet, verändert; sobald eine bestimmte Funktion genügend lange verfügbar ist oder eine Reihe von Suchmaschinen diese Funktion anbieten, wird sie auch von allen anderen erwartet.

Wir wollen die verschiedenen Funktionen oder Funktionsbereiche im Folgenden gemäß der untenstehenden Gliederung betrachten:

9.5.1 Einfache Suche mit Suchbegriffen

9.5.2 Suche nach Phrasen

9.5.3 Einschränkung des Suchergebnisses

9.5.4 Web Directories

9.5.1 Einfache Suche mit Suchbegriffen

Alle Suchmaschinen bieten dem Nutzer die Möglichkeit an, nach Dokumenten im WWW zu suchen, welche einen oder mehrere Suchbegriffe

enthalten. Das Portal der Suchmaschinen enthält hierfür ein Eingabefeld, in das die Suchbegriffe eingegeben werden können. Im Falle von Google ist das Hauptportal der deutschsprachigen Variante der Suchmaschine unter `www.google.de` zu finden. Abbildung 9.10 zeigt dieses Portal zu Google. In das Textfeld auf der Startseite können Suchbegriffe eingegeben werden. Die Suchbegriffe werden dabei implizit konjunktiv (mit UND) verknüpft.

Abbildung 9.10 — Eingabe von einfachen Suchbegriffen in Google

Zur Suche nach relevanten Dokumenten nutzen Suchmaschinen im Prinzip Methoden, wie sie aus dem Information Retrieval bekannt sind. Insbesondere das Boolesche Retrieval und das Vektorraummodell sind hier zu nennen. Viele Suchmaschinen haben die genannten Methoden weiterentwickelt. Google etwa verwendet eine PageRank™ genannte Technik, die bei der Ermittlung der Relevanz eines Dokuments für eine Anfrage unter anderem auch die Linkstruktur zwischen den Seiten in Betracht zieht. Dieses und verwandte Verfahren werden im Kapitel 9.7 vorgestellt.

PageRank™

Ist die Suchmaschine mit der Anfragebearbeitung fertig, wird eine Ergebnisrepräsenation generiert. Dies ist in den meisten Fällen ein HTML-Dokument, das in einer Auflistung Hyperlinks zu den für relevant befundenen Dokumenten enthält. Zusätzlich enthalten diese Ergeb-

Ergebnisrepräsenation

nisrepräsentationen häufig weitere Informationen wie etwa den Titel der Dokumente oder kurze Textauszüge. Abbildung 9.11 zeigt die Ergebnisrepräsentation von Google.

Abbildung 9.11 — Ergebnispräsentation von Google

Andere Suchmaschinen wie z.B. Vivísimo (letzter Abruf: 18.12.2006) ordnen die gefundenen Dokumente zusätzlich Kategorien zu, um so insbesondere bei umfangreicheren Ergebnissen den Überblick zu erleichtern (siehe Abbildung 9.12).

9.5.2 Suche nach Phrasen

Phrasen

Viele Suchmaschinen bieten die Möglichkeit, nicht nur nach Suchbegriffen zu suchen, sondern auch nach Phrasen, also zusammenhängenden Termgruppen. Meist kann die Suche nach Phrasen genutzt werden, indem die Suchphrase in Anführungszeichen gesetzt wird. Die Suchmaschine sucht dann nur nach Dokumenten, welche die gesuchte Phrase enthalten (vgl. Abbildung 9.13).

Vorkommensort

Um diese Funktion anbieten zu können, muss der Index der Suchmaschine neben der grundlegenden Information, welcher Begriff in welchen Dokumenten vorkommt (etwa in einer invertierten Liste), zusätzlich Informationen über die Termpositionen innerhalb der Dokumente enthal-

Abbildung 9.12 — Ergebnispräsentation von Vivísimo

ten.

9.5.3 Einschränkung des Suchergebnisses

Anhand bestimmter Kriterien lässt sich die von der Suchmaschine zurückgegebene Menge als relevant befundener Dokumente weiter einschränken. Übliche Kriterien zur Einschränkung sind:

Datum, Einschränkung der Suche

- **Datum / Aktualtität der Dokumente**

  Nur Dokumente, die in einem angegebenen Zeitfenster indexiert bzw. neu in den Index aufgenommen wurden, können in das Suchergebnis aufgenommen werden. Selbst relevante Dokumente werden nicht aufgenommen, wenn sie etwa zu alt sind.

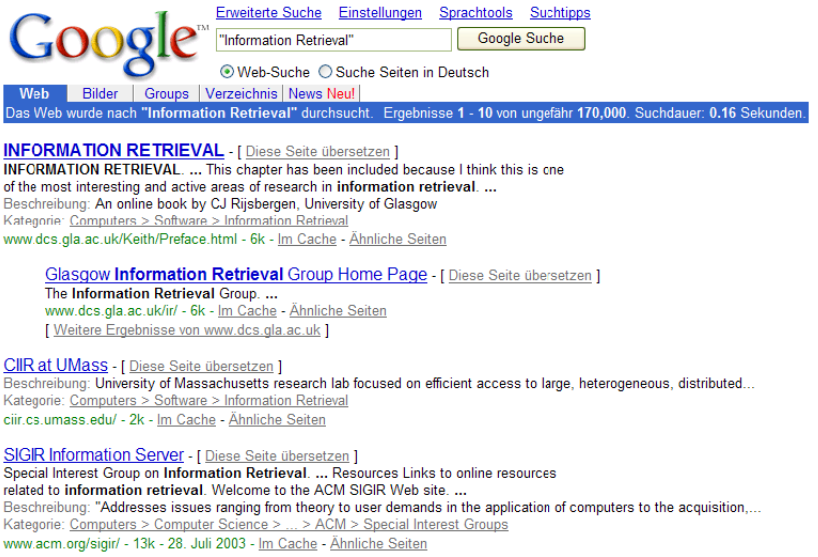


Abbildung 9.13 — Die Suche nach Phrasen mit Google.

Dateityp, Einschränkung der Suche

- **Dateityp**

  Nur Dokumente in einem bestimmten Dokumentformat werden – bei Relevanz – in das Ergebnis aufgenommen. So kann man seine Suche z.B. auf Dokumente im PDF-Format einschränken.

Domain, Einschränkung der Suche

- **Domain**

  Bei der Anfrageverarbeitung werden nur solche Dokumente betrachtet, die von einer bestimmten Domain oder einer Gruppe von Domains stammen. Da das System des Domain Name Service (DNS) (vgl. hierzu z.B. www.linuxfibel.de, letzter Abruf: 18.12.2006) den Namensraum der WWW-Server hierarchisch partitioniert, können auch ganze Teilbäume des Namensraums (also der Domain) ausgewählt werden (etwa `uni-bamberg.de`). Genauso können bestimmte Domains ausgeschlossen werden.

Land, Einschränkung der Suche

- **Land / TLD (Top Level Domain)**

  Eine Sonderform der Einschränkung nach Domain ist die Beschränkung des Suchergebnisses auf Dokumente aus einem bestimmten Land. Den meisten Ländern ist eine zweistellige Top Level Domain (TLD) zugeteilt worden (z.B. `.de` im Falle von Deutschland). Wird

die Suche nun auf Dokumente beschränkt, die aus einer Domain stammen, welche unterhalb der TLD `.de` angesiedelt ist, befinden sich nur Ergebnisse aus Deutschland (bzw. von Servern der deutschen TLD) im Suchergebnis.

Zu beachten ist, dass es zusätzlich zu den Länder-TLDs auch einige generische TLDs gibt. Zu nennen sind hier insbesondere `.com`, `.org`, `.net` und `.edu`. Diese waren ursprünglich für bestimmte WWW-Anbieter in den USA vorgesehen (etwa für kommerzielle Firmen, Non-Profit-Organisationen, Netzbetreiber und Bildungsinstitutionen); inzwischen können Domains unterhalb dieser TLDs aber auch an WWW-Anbieter aus anderen Staaten vergeben werden.

- **Sprache**

  Sprache, Einschränkung der Suche

  Einige Suchmaschinen nutzen Indexierer, die ein Dokument nicht nur auf die enthaltenen Terme hin untersuchen, sondern auch (nach heuristischen Verfahren) eine Einschätzung abgeben, in welcher Sprache dieses Dokument verfasst ist. Dabei können neben den im Dokument verwendeten Begriffen auch andere Kriterien wie etwa die Domain berücksichtigt werden.

  Mit diesen Informationen im Index kann die Suche nach Dokumenten nun auch auf Dokumente eingeschränkt werden, die in einer bestimmten Sprache verfasst worden sind.

- **Position der Suchbegriffe im Dokument**

  Position, Einschränkung der Suche

  Es kann angegeben werden, an welcher Position im Dokument sich die gesuchten Begriffe befinden sollen. So kann die Suche etwa auf Dokumente eingeschränkt werden, welche die gesuchten Terme im Titel tragen.

Abbildung 9.14 zeigt die Möglichkeiten zur Einschränkung von Suchergebnissen, die Google bietet. Es sind zusätzlich zu diesen Kriterien sicherlich noch andere Kriterien denkbar. In manchen Situationen könnte beispielsweise eine Einschränkung der Größe des gesuchten Dokuments sinnvoll sein.

9.5.4 Web Directories

Viele Suchmaschinen bieten zusätzlich zur Möglichkeit der Suche auch ein Webverzeichnis oder Web Directory an. In diesem Verzeichnis ist eine

Web Directory

Abbildung 9.14 — Google bietet viele Möglichkeiten, die Suche nach bestimmten Kriterien einzuschränken

große Zahl von Websites hierarchisch kategorisiert. Innerhalb der einzelnen Kategorien wird auf noch speziellere Subkategorien sowie auf eine Reihe von Websites verwiesen, die thematisch in die gerade betrachtete Kategorie einzuordnen sind (vgl. Abbildung 9.15).

Die Nutzung eines Verzeichnisses bietet sich dann an, wenn sich das Informationsbedürfnis des Nutzers eindeutig einer Kategorie oder Subkategorie zuordnen lässt. Insbesondere ist es auch möglich, zunächt innerhalb des Directories zu einer Subkategorie zu browsen und seine Suche dann auf Websites und Dokumente innerhalb dieser Kategorie und ihrer Subkategorien zu beschränken. (Vergleiche hierzu auch Abschnitt 6.1.3.)

9.6 Erweiterte Funktionen

Einige Suchmaschinen bieten Optionen an, welche über die eben vorgestellten Möglichkeiten hinausgehen. Diese speziellen Fähigkeiten sollen in diesem Abschnitt vorgestellt werden.

Abbildung 9.15 — Das Web Directory von Google

9.6.1 Suche im Internet außerhalb des WWW

9.6.2 Suche nach Bildern

9.6.3 Metasuchmaschinen

9.6.1 Suche im Internet außerhalb des WWW

Das WWW stellt nur einen kleinen Teil des Internets dar – wenn auch einen sehr wichtigen und populären. Neben dem WWW gibt es weitere Dienste, die im Internet angeboten werden, darunter so wichtige wie eMail, FTP (das *File Transfer Protocol* zum Übertragen von Dateien zwischen Rechnern) und das NNTP (*Network News Transport Protocol*).

eMail
FTP
Network News Transport Protocol, NNTP

Auf dem NNTP baut ein weltweiter Nachrichtendienst auf, der Foren zu verschiedensten Themen enthält. Die einzelnen Foren sind in einem hierarchischen Namensraum angeordnet, dessen Namensbestandteile von vorne nach hinten gelesen werden (also umgekehrt zum DNS). Die Newsgroup `comp.lang.java` etwa ist eine Newsgroup, die sich mit *computer science* (Informatik) befasst, und zwar mit *languages* (Sprachen), insbesondere mit der Programmiersprache Java.

Newsgroup

In die Newsgroups können Artikel eingestellt werden, welche Diskussionsbeiträge bzw. Fragen zu bestimmten Themen enthalten. Veröffentlicht werden diese Artikel auf einem lokalen NNTP-Server (z.B. auf dem Newsserver der Universität); die einzelnen Newsserver stehen weltweit in Kontakt und synchronisieren sich miteinander. Die Summe aller NNTP-Server wird auch als Usenet bezeichnet. Da insbesondere Computerspezialisten weltweit diese Möglichkeit zur Kommunikation nutzen, hat man gute Chancen, auf gestellte Fragen qualifizierte Antworten zu erhalten.

Somit stellen Newsgroups eine reiche Informationsquelle dar. Google indexiert diese Newsgroups (zumindest einen großen Teil davon) und bietet die Möglichkeit, in den Newsgroups nach Artikeln zu suchen, die für das Informationsbedürfnis des Nutzers relevant sind. Abbildung 9.16 zeigt die Benutzerschnittstelle von Google zur Suche in Newsgroups.

Google, Inc. hat im Jahr 2001 von der Firma Deja.com neben anderen Rechten komplette Archive von Beiträgen im Usenet gekauft und archiviert seitdem auch die neu hinzukommenden Beiträge. Diese Archive wurden indexiert und somit auch für das Suchen erschlossen. Darüber hinaus betreibt auch Google selbst NNTP-Server, die sich täglich mit anderen NNTP-Servern synchronisieren; auf diese Weise werden auch die aktuellen News-Artikel zeitnah in den Index aufgenommen und können für die Suche genutzt werden. Zur Zeit stehen Google mehr als eine Milliarde Usenet-Artikel zur Verfügung, die bis zum Jahr 1981 zurückreichen (Stand: 14.12.2006).

Da viele Beiträge in Newsgroups von Experten verfasst sind, stellt dieser Index innerhalb von Google eine Quelle für hochwertige Information dar. Viele Probleme, die ein Nutzer nicht ohne weiteres selbst lösen kann, sind bereits Gegenstand von Diskussionen in einer Newsgroup gewesen. Häufig kann man dort auch die Lösung seines Problems finden.

Mailing-Liste

Eine weitere viel genutzte Informationsquelle im Internet sind Mailing-Listen. Diese werden – meist von größeren Organisationen wie etwa GNU oder Firmen – angeboten, um vielen Abonnenten der jeweiligen Mailing-Liste aktuelle Informationen zukommen zu lassen. Dies können Produktinformationen, aber auch Diskussionsbeiträge zu technologischen, politischen und anderen Entwicklungen sein.

Die einzelnen Mitteilungen einer Mailing-Liste erhalten Nutzer jedoch nur, wenn sie diese auch abonniert haben, so dass sie keinen Zugriff auf Mitteilungen haben, die vor ihrer Registrierung versandt wurden. Genauso kann es problematisch sein, wenn ein Nutzer sein eMail-Archiv

Abbildung 9.16 — Google Interface für die Suche in Newsgroups

verliert (etwa durch Rechnerschaden) oder verkleinern möchte. Aus diesem Grund gibt es Mailing-List-Archive, die meist über WWW zugänglich sind, oft aber auch als komprimierte Textdatei o.ä. zum Download bereitstehen.

Einige Suchmaschinen indexieren auch solche Mailing-List-Archive und machen deren Inhalt so der Suche nach relevanter Information im Internet zugänglich.

9.6.2 Suche nach Bildern

Bei den bisher vorgestellten Angeboten von Suchmaschinen handelte es sich ausschließlich um Möglichkeiten zur Suche nach textueller Information – wenn auch das Format der textuellen Information von Plain Text über HTML bis hin zu Office-Formaten und PDF reicht.

Das Internet (und insbesondere das WWW) beinhaltet allerdings noch andere Medientypen, welche ebenfalls indexiert und damit für die Suche erschlossen werden können. Insbesondere die Suche nach Bildern ist

Bildsuche im Internet

dabei von Interesse. Das WWW enthält eine Vielzahl von Abbildungen: Fotographien, Logos, graphische Elemente u.v.m. Ein besonderer Service einer Suchmaschine ist es daher, den Nutzer bei der Suche nach Bildern im WWW zu unterstützen.

Hierzu gibt es zwei deutlich zu unterscheidende Ansätze:

- **Google und andere allgemeine Suchmaschinen**

  Allgemeine Suchmaschinen sind auf die Suche nach textueller Information spezialisiert. Aufbauend auf ihren Fähigkeiten in diesem Anwendungsbereich bieten einige Suchmaschinen auch die Suche nach Bildern an. Der Benutzer gibt bei der Suche einige Suchbegriffe an, die er in der Umgebung der zu findenden Bilder vermutet, beispielsweise den Namen einer Person oder auch den Namen einer Blume. Die Suchmaschine findet während der Anfrageverarbeitung dann in einem speziellen Bildindex alle Bilder, in deren Umgebung (etwa in der Bildunterschrift oder im Titel des Dokuments) die gesuchten Begriffe vorkommen.

Abbildung 9.17 — Das Ergebnis einer Bildsuche in Google

Das Suchergebnis wird nach üblichen Verfahren des Information Retrieval erstellt und gerankt. Zu beachten ist, dass die Suche nicht auf den Bildern selbst, sondern auf den Textdokumenten, welche die Bilder enthalten, durchgeführt wird. Die zugrunde liegende Annahme ist, dass Bilder durch Begriffe, die in ihrer Umgebung vorkommen, gut beschrieben werden. Im Index von Google sind momentan etwa 880 Millionen Bilder enthalten (Stand: 12.12.06).

- **Spezialisierte Systeme**

  Neben allgemeinen Suchmaschinen, die mit ihren »hauseigenen« Mitteln auch eine Bildsuche ermöglichen, existieren Systeme, die auf das Image Retrieval im Web spezialisiert sind (vgl. hierzu auch Kapitel 8). Als Beispiel soll an dieser Stelle das System Viper von der Universität Genf genannt werden (http://viper.unige.ch, letzter Abruf: 18.12.2006). Es steht stellvertretend für Systeme des Content Based Image Retrieval (CBIR). Viper basiert dabei auf Gift, dem GNU Image Finding Tool (http://www.gnu.org/software/gift/, letzter Abruf: 18.12.2006).

  Gift, GNU Image Finding Tool

  Solche Systeme erlauben als Anfrage nicht einen oder mehrere Begriffe, sondern ein Beispielbild – also eine *Query by Example* (QbE). Das System sucht ausgehend vom Beispielbild ähnliche Bilder in seiner Datenbank und präsentiert das Ergebnis dem Benutzer. Dieser kann über ein Relevance Feedback seine Suche verfeinern und nach mehreren Iterationen (hoffentlich) zu einem Suchergebnis gelangen, das seinem Informationsbedürfnis entspricht. Viper arbeitet dabei allerdings derzeit nicht auf dem WWW sondern auf einer Beispielkollektion von Bildern (vgl. Abbildung 9.18).

  Query by Example

9.6.3 Metasuchmaschinen

Neben den bislang beschriebenen »einfachen« Suchmaschinen existieren so genannte *Metasuchmaschinen*. Während einfache Suchmaschinen nur die Suche auf ihrem eigenen Index erlauben (und damit u.U. sehr große Bereiche des WWW gar nicht durchsucht werden), ermöglichen Metasuchmaschinen die parallele Suche auf mehreren Indexen. Metasuchmaschinen transformieren dazu die an sie gestellten Anfragen in das Anfrageformat anderer (»einfacher«) Suchmaschinen und leiten diese transformierten Anfragen dann an diese Suchmaschinen weiter.

Metasuchmaschine

Viele Metasuchmaschinen lassen dem Anwender dabei die Wahl, zu welchen Suchmaschinen eine Anfrage weitergeleitet werden soll. Darüber hinaus gibt es oft Funktionen, um in mehreren Suchmaschinen gefundene Dokumente als einen Treffer anzeigen zu lassen (vgl. Abbildung 9.19).

Die Motivation hinter Metasuchmaschinen besteht darin, dass eine einzelne Suchmaschine nur einen gewissen, meist relativ geringen Teil des

WWW in ihrem Index hält. Es wird angenommen, dass die von den einzelnen Suchmaschinen indexierten Teile des WWW nur teilweise überlappen, dass also einzelne Dokumente nur von einer oder einigen wenigen Suchmaschinen indexiert wurden. Die Suche über die Metasuchmaschine entspricht einer Vereinigung der Menge der in den einzelnen Indexen enthaltenen Dokumente, so dass insgesamt ein größerer Teil des WWW abgedeckt werden kann. Problematisch sind dabei aber einerseits die Eliminierung von Duplikaten im Ergebnis (Dokumente, die von mehreren Suchmaschinen gefunden wurden) und andererseits die Erstellung eines übergreifenden Rankings für die Ergebnisdokumente.

9.7 Spezielle Verfahren im Web Information Retrieval

Eine alleinige Anwendung klassischer Techniken des Information Retrieval (also z.B. des Vektoraummodells oder des Booleschen Retrieval) erscheint im Kontext des WWW und der Suche im WWW aus verschiedenen Gründen nicht sinnvoll:

Precision Recall Qualität

- Im Web Information Retrieval wird auf einer »Dokumentenkollektion« operiert, die in Bezug auf ihre Größe andere Kollektionen bei weitem in den Schatten stellt. Allein schon aus diesem **Umfang** der Dokumentenkollektion ergibt sich, dass bei einer Suche Precision häufig wichtiger sein wird als Recall.
- Die Dokumente im Web sind von sehr unterschiedlicher **Qualität**. Ein Ranking allein aufgrund der in einem Dokument vorkommenden Begriffe berücksichtigt dies nur unzureichend. Statt dessen wären zusätzliche Informationen über die »Qualität« der Dokumente für das Ranking wichtig.
- Typische **Anfragen** an eine Suchmaschine für das Web bestehen nur **aus ein bis drei Begriffen**. Selbst wenn man nur Dokumente betrachtet, die alle Anfragebegriffe enthalten, erhält man sehr viele Treffer. Unter diesen erscheint ein Ranking allein aufgrund der Vorkommenshäufigkeit von Begriffen recht problematisch.

Hypertext-Dokumente

- Hinzu kommt, dass die im WWW abrufbaren Dokumente zu einem großen Teil **Hypertext-Dokumente** sind – zwischen ihnen bestehen also in Form von Hyperlinks gerichtete Beziehungen. Diese

Beziehungen sagen implizit etwas über die Bedeutung von Dokumenten aus. Sie können in Analogie zu wissenschaftlichen Artikeln als »Zitate« betrachtet werden.

- Die Dokumente im Internet werden **von vielen verschiedenen Personen** gelesen. Es erscheint daher denkbar, Urteile der Leser zu Dokumenten zu erfassen und diese Urteile bei der Ermittlung von Ergebnislisten zu berücksichtigen (»15 von 17 Lesern fanden diesen Artikel hilfreich«).
- Schließlich steckt im Internet auch ein hohes **kommerzielles Potential**. Wird ein Dokument über eine Suchmaschine gefunden, so kann dies für den Anbieter des Dokumentes sehr wichtig sein (nur wer gefunden wird kann z.B. etwas verkaufen). Es liegt daher nahe, auch an kommerziell gesteuerte Ergebnislisten zu denken.

Aus diesen Gründen werden im Web Information Retrieval verfeinerte Methoden und Verfahren eingesetzt. Im Folgenden werden wir zunächst auf den PageRank™-Algorithmus von Google eingehen, der versucht die Popularität von Webseiten als Indikator für ihre Qualität zu messen. Anschließend werden wir uns mit »bezahlten Links« beschäftigen und schließlich kurz auf Bewertungssysteme eingehen:

9.7.1 PageRank-Algorithmus

9.7.2 Sponsored Links

9.7.3 Bewertungssysteme

9.7.1 PageRank™-Algorithmus

PageRank™

Grundidee des PageRank™-Algorithmus ist es, die Hyperlink-Struktur des WWW zu nutzen, um die Qualität und Relevanz der Dokumente im WWW besser einschätzen zu können.

forward link
back link

PageRank geht dabei davon aus, dass es für ein bestimmtes Dokument im WWW zwei Arten von Hyperlinks gibt: die *forward links*, die von diesem Dokument aus auf andere Dokumente zeigen, und die *back links*, welche von anderen Dokumenten auf das gerade betrachtete Dokument zeigen. Eine erste Arbeitshypothese ist dabei, dass die Qualität eines Dokuments umso höher ist, je mehr Backlinks es aufzuweisen hat. Dies bedeutet

nämlich im Allgemeinen, dass eine Reihe von Autoren im WWW das Dokument für so gut halten, dass sie von ihren Dokumenten aus auf dieses Dokument verlinken. Die Anzahl der Backlinks auf ein Dokument kann also als ein grobes Maß für die Qualität dieses Dokuments dienen.

Allerdings sind nicht alle Backlinks gleich zu bewerten. Kommt ein Backlink von einer qualitativ hochwertigen Seite, ist ihm sicherlich mehr Gewicht beizumessen als einem Backlink aus den »Niederungen« des WWW. Daher berücksichtigt PageRank auch die Qualität der Dokumente, von denen aus die Backlinks auf das gerade zu bewertende Dokument zeigen.

Problematisch an der Verwendung von Backlinks zur Bewertung von Dokumenten ist die Tatsache, dass nicht sichergestellt werden kann, dass alle Backlinks eines Dokuments bekannt sind; um dies zu erreichen, müssten der Suchmaschine bzw. ihrem Crawler alle Dokumente im WWW bekannt sein. Da das WWW mehrere Milliarden Dokumente enthält und täglich einige Millionen neue hinzukommen sowie einige Millionen bereits existente Dokumente geändert werden, ist dieses Unterfangen zum Scheitern verurteilt.

Berechnung des PageRank eines Dokuments

Nach dieser einleitenden Beschreibung soll nun gezeigt werden, wie der *PageRank* eines Dokuments im WWW berechnet wird. Die Ausführungen orientieren sich an `http://pr.efactory.de/d-index.shtml` (letzter Abruf: 11.12.2006).

Der ursprüngliche PageRank-Algorithmus wurde von Lawrence Page und Sergey Brin mehrfach beschrieben [BP98]. Er hat die folgende Form:

$$\mathcal{PR}(A) = (1-d) + d \cdot \sum_{i=1}^{n} \frac{\mathcal{PR}(T_i)}{C(T_i)} \tag{9.1}$$

Hierbei ist:

- $PR(A)$ der PageRank einer Seite A,
- $PR(T_i)$ der PageRank der Seiten T_i, von denen ein Link auf die Seite A zeigt,
- $C(T_i)$ die Gesamtanzahl der Links auf Seite T_i und

- d ein Dämpfungsfaktor (*damping factor*), der zwischen 0 und 1 liegt.

Das PageRank-Verfahren bewertet damit grundsätzlich nicht Websites in ihrer Gesamtheit, sondern basiert ausschließlich auf der Beziehung einzelner Webseiten zueinander. Der PageRank einer Seite A bestimmt sich dabei rekursiv aus dem PageRank derjenigen Seiten, von denen ein Link auf die Seite A zeigt.

Der PageRank der Seiten T_i , die auf eine Seite A verlinken, fließt nicht gleichmäßig in den PageRank von Seite A ein. Der PageRank einer Seite T_i wird stets anhand der Anzahl $C(T_i)$ der von Seite T_i ausgehenden Links gewichtet. Das bedeutet, je mehr ausgehende Links eine Seite T_i hat, umso weniger PageRank gibt sie an Seite A weiter. Eine Seite T_i verteilt ihren PageRank $PR(T_i)$ damit quasi über ihre ausgehenden Links.

Der anhand der Anzahl der jeweils ausgehenden Links gewichtete PageRank der Seiten T_i wird nun über alle Backlinks zur Seite A addiert. Dies hat zur Folge, dass jeder zusätzliche eingehende Link für eine Seite A den PageRank dieser Seite A erhöht.

Schließlich wird die Summe der gewichteten PageRanks der Seiten T_i mit dem Dämpfungsfaktor d, der stets zwischen 0 und 1 liegt, multipliziert. Hierdurch wird das Ausmaß der Weitergabe des PageRanks von einer Seite auf eine andere verringert.

Das Random Surfer Modell

Lawrence Page und Sergey Brin beschreiben in ihren Veröffentlichungen eine sehr einfache, intuitive Begründung des PageRank-Algorithmus. Sie betrachten das PageRank-Verfahren als ein Modell zur Abbildung von Benutzer-Verhalten. Hierzu führen sie einen »Zufalls-Surfer« ein, der von einer Webseite zur nächsten jeweils beliebige Links verfolgt, ohne dabei auf die Inhalte zu achten.

Der Zufalls-Surfer befindet sich mit einer bestimmten Wahrscheinlichkeit auf einer Website, die sich aus deren PageRank herleiten lässt. Die Wahrscheinlichkeit, dass der Zufalls-Surfer nun einem bestimmten Link folgt, ergibt sich einzig und allein daraus, aus wie vielen Links er die Auswahl hat. Aus diesem Grund fließt der PageRank einer verlinkenden Seite stets nach der Anzahl ihrer ausgehenden Links gewichtet in die PageRank Berechnung der verlinkten Seiten ein.

Die Wahrscheinlichkeit, dass der Zufalls-Surfer auf eine Seite gelangt, ist also die Summe der Wahrscheinlichkeiten, mit der er von einer verlinkenden Seite den entsprechenden Link verfolgt. Nun wird allerdings die Wahrscheinlichkeit, mit der der Zufalls-Surfer auf eine Seite gelangt, mit dem Faktor d gedämpft. Dies hat im Rahmen des Random Surfer Modells den Hintergrund, dass der Zufalls-Surfer nicht unendlich viele Links verfolgt. Nach einer bestimmten Zeit wird er gelangweilt und ruft eine beliebige andere Webseite auf.

Random Surfer Modell

Die Wahrscheinlichkeit, mit der der Zufalls-Surfer die Verfolgung von Links nicht abbricht und somit weiterklickt, wird durch den Dämpfungsfaktor d angegeben, der abhängig von der Höhe der Wahrscheinlichkeit einen Wert von 0 bis 1 annimmt. Je höher d ist, um so wahrscheinlicher ist es, dass der Zufalls-Surfer Links verfolgt. Da der Zufalls-Surfer nach dem Abbruch der Link-Verfolgung eine beliebige Seite aufruft, geht die Wahrscheinlichkeit mit der er dies tut, mit dem Wert $(1 - d)$ als Konstante in die Berechnung des PageRanks einer jeden Seite ein.

Berechnung des PageRank

Berechnung des PageRank

Die Formel zur Berechnung des PageRank ist rekursiv formuliert: zur Berechnung von $PR(A)$ ist die Kenntnis der PageRanks $PR(T_i)$ aller Dokumente notwendig, welche auf das Dokument A verweisen. In der Praxis beginnt man mit der Berechnung des PageRanks eines beliebigen Dokuments und fährt nach einer bestimmten Reihenfolge mit der Berechnung der PageRanks der anderen Dokumente fort. Sind die PageRanks für alle Dokumente berechnet, so beginnt man mit dem nächsten Durchgang; fällt die Änderung der PageRanks von einer Iteration zur nächsten unter einen zu wählenden Schwellwert, geht man von der Konvergenz des Algorithmus aus und beendet die Berechnung.

In der Praxis treten dabei noch einige andere Probleme auf:

Dangling Links

- **Dangling Links**

  Aufgrund der Anzahl der im WWW publizierten Dokumente können selbst kommerziell betriebene Suchmaschinen nicht alle Dokumente indexieren. Daher werden immer eine gewisse Anzahl von in Dokumenten gefundenen Hyperlinks ins »Leere« zeigen; die von ihnen referenzierten Dokumente existieren zwar im WWW, konnten allerdings nicht in die aktuelle Indexierung des WWW durch die Suchmaschine aufgenommen werden.

Zur Lösung dieses Problems bedient man sich einer ebenso einfachen wie groben Methode: alle Hyperlinks im Index, die auf nicht im Index enthaltene Dokumente verweisen, werden für die Berechnung des PageRank ignoriert – eventuell werden sie sogar aus dem Index gelöscht.

- **Rank Sinks** *Rank Sinks*

  Es kann Fälle geben, in denen eine Reihe von Dokumenten im WWW zyklisch aufeinander verweisen, von diesen Dokumenten aber keinerlei Links auf andere Dokumente ausgehen. Wird nun von irgendeinem Ort des WWW aus auf ein Dokument dieses Zyklus verwiesen, so »fließt« über diesen Backlink Gewicht in den Zyklus ein, so dass die in diesem Zyklus enthaltenen Dokumente einen höheren PageRank erhalten. Da diese Dokumente sich wiederum zyklisch beeinflussen, steigen mit jeder Iteration des PageRank-Algorithmus die PageRanks der beteiligten Dokumente; das »herumgereichte« Gewicht (das initial von einem Dokument außerhalb des Zyklus stammte) kann nirgendwo »abfließen«. Eine solche Hyperlinkstruktur wird im Kontext von PageRank auch als *rank sink* bezeichnet, da ausgehend von der externen Quelle (*source*) Gewicht in diese Senke (*sink*) einfließt, aber niemals wieder abfließen kann.

  Aus den Erfahrungen mit eben solchen zyklischen Hyperlinkstrukturen ging der Ansatz hervor, dem Fluss der Gewichte von Dokument zu Dokument eine natürliche Grenze zu setzen. Diese besteht dadurch, dass eine *decay*- oder auch *aging*-Komponente eingeführt wird. Diese sorgt dafür, dass das Gewicht eines Dokuments nur über eine begrenzte Zahl von Hyperlinks weitergereicht werden kann, indem bei jedem Übergang über einen Hyperlink ein gewisser Bruchteil abgezogen wird.

Eine recht gelungene Beschreibung zur Berechnung der PageRank-Werte findet sich auch bei `www.ArtikelWeb.de` in dem entsprechenden `Eintrag` (letzter Abruf: 12.12.06).

Manipulation des PageRank

Das Konzept des PageRank hat in den ersten Jahren seiner Verwendung auch deshalb so gut funktioniert, weil der Anbieter einer Seite nur mit sehr hohem Aufwand den PageRank seiner Seite manipulieren konnte. Durch »Link-Farmen« und das gegenseitige Verlinken von Webseiten un- *Manipulation* *Link-Farm*

ter befreundeten Institutionen ergeben sich aber auch hier Möglichkeiten der Manipulation. So sind z.B. ausgehende Links von universitären Seiten recht begehrt, weil diese durch die Regeln von Google leicht einen hohen PageRank erreichen können, der dann über den ausgehenden Link weitergeleitet werden kann.

Auf der Wikipedia-Seite zum PageRank (letzter Abruf: 12.12.06) wird diese Kritik wie folgt formuliert:

Die hohe wirtschaftliche Bedeutung des PageRanks spornt viele Unternehmen an, die eigene Position auf der Suchergebnisseite durch »Tricks« zu verbessern. Hier herrscht zurzeit keine Chancengleichheit mehr, denn jeder kann sich eine Position gezielt erkaufen. Es hat sich ein ganzer Geschäftszweig der *Suchmaschinenoptimierung* daraus entwickelt. Die Nachteile von PageRank im Überblick:

- Finanziell Starke können sich Backlinks erkaufen, und werden in Suchergebnissen höher positioniert. Gezielte Manipulation wird seit geraumer Zeit betrieben.
- Statt qualitativ hochwertigem Inhalt entscheiden oft die finanziellen Möglichkeiten über die Reihenfolge der Suchergebnisse.
- Webmaster sehen oft im PageRank das einzige Bewertungskriterium für den Linktausch. Der Inhalt der verlinkten Seiten gerät in den Hintergrund, entscheidend ist nur noch der PageRank.

Trotz dieser berechtigten Kritik ist der PageRank nach wie vor als ein durchaus sinnvolles Maß für die Popularität einer Webseite anzusehen, das allerdings nicht gefeit gegen Manipulation ist.

Berücksichtigung des PageRank bei der Erstellung des Rankings

Ergebnisranking

Zur Bestimmung des Ergebnisrankings bei einer Anfrage liegen nun zwei Kriterien vor:

1. Die Einschätzung der inhaltlichen Ähnlichkeit z.B. auf Basis des Booleschen Retrievals oder des Vektorraummodells und
2. der PageRank der Seiten als Maß für ihre Popularität.

Um aus diesen beiden Kriterien nun ein Ranking zu erstellen wäre zunächst eine Verrechnung beider Kennzahlen denkbar: also etwa *inhaltliche Relevanz zur Anfrage multipliziert mit dem PageRank-Wert*. Da aufgrund der einfachen Anfragen und der hohen Zahl der Dokumente im WWW in der Regel sehr viele Dokumente existieren, die alle Anfrageterme enthalten, bietet sich als Alternative an, nur die Dokumente zu berücksichtigen, die alle Anfrageterme enthalten, und diese nach ihrem PageRank zu ordnen.

Daneben ist es im Hinblick auf die inhaltliche Ähnlichkeit natürlich auch noch möglich, Vorkommen der Suchterme in der URL, in Überschriften oder in Fettdruck höher zu gewichten als Vorkommen im normalen Text. Die Details dieser Gewichtungen versuchen die Suchmaschinenbetreiber natürlich geheim zu halten. Einen kompakten Überblick hierzu gibt auch die Seite zum `Website-Marketing` der `lb medien` (letzter Abruf: 12.12.06).

9.7.2 Sponsored Links

Auf eine Suchanfrage durch den Nutzer hin präsentiert eine Suchmaschine das Suchergebnis üblicherweise in Form einer tabellarischen Auflistung der Verweise auf die gefundenen relevanten Dokumente. Diese Auflistung ist dabei nach absteigenden RSVs geordnet. Dabei wird die Gesamttabelle normalerweise auf mehrere Ergebnisseiten aufgeteilt, mit etwa zehn bis 25 gefundenen Dokumenten pro Seite. Dies geschieht zum einen, um dem Nutzer bereits sehr schnell die relevantesten Dokumente für seine Anfrage anzeigen zu können, da die Ladezeit einer solchen stark verkürzten Ergebnispräsentation wesentlich geringer ist als das Anzeigen der kompletten Liste (mit eventuell mehreren hunderttausend gefundenen relevanten Dokumenten). Zum anderen wird die Navigation durch die Liste auch wesentlich einfacher, da der Nutzer sich nun von Seite zu Seite durch das Ergebnis vorarbeiten kann.

Retrieval Status Value, RSV

Die Position eines gefundenen relevanten Dokuments in dieser Ergebnispräsentation ist dabei ein wesentlicher Einflussfaktor für die Wahrscheinlichkeit dafür, dass der Nutzer das Dokument liest bzw. die WWW-Seite besucht. Für viele Firmen ist inzwischen das WWW zum wichtigsten Medium der Kommunikation mit ihren Kunden geworden. Für solche Firmen ist es daher essentiell, dass möglichst viele WWW-Nutzer ihre Seiten besuchen.

Nun kann es sein, dass eine Firma *F, Inc.* ein Produkt *P* verkauft, aber in einer Suchmaschine bei Anfragen nach diesem Produkt oder der entsprechenden Produktgruppe nicht auf der ersten Ergebnisseite erscheint. Viele Suchmaschinen bieten für solche Unternehmen die Möglichkeit an, einen so genannten *sponsored link* zu kaufen. Beispielsweise kann *F, Inc.* einen *sponsored link* kaufen, so dass immer dann, wenn ein Nutzer nach *P* oder der entsprechenden Produktgruppe sucht, eine kurze Anzeige oder auch nur ein Link zur Website von *F, Inc.* auf der ersten Seite der Ergebnispräsentation eingebunden wird.

sponsored link

Dabei ist der Erwerb von *sponsored links* nicht auf solche Suchbegriffe beschränkt, die direkt im Namen des Käufers oder dessen Produkt vorkommen. Aus Sicht der Firma kann es auch sinnvoll sein, *sponsored links* für Begriffe zu erwerben, die in den generellen Kontext der vom Unternehmen angebotenen Produkte oder Dienstleistungen fallen.

Während die *sponsored links* etwa bei Google noch als solche erkannt werden können (siehe Abbildung 9.20), bieten manch andere Suchmaschinen eine Dienstleistung, die etwa *pay-for-performance* (z.B. bei YaHoo Search Marketing, letzter Abruf: 18.12.2006) genannt wird. Hierbei wird – wie bei den *sponsored links* – für Suchbegriffe gezahlt, bei denen ein Unternehmen möglichst weit oben im Suchergebnis stehen möchte. Dabei stehen einzelne Unternehmen, die sich für die gleichen Suchbegriffe interessieren, in direkter Konkurrenz: dasjenige Unternehmen, das am meisten zahlt, wird auch am weitesten oben angezeigt.

pay-for-performance

Aus Nutzersicht ist der Hauptunterschied zwischen *sponsored links* (etwa bei Google) und *pay-for-performance*, dass Dokumente, die aufgrund von *pay-for-performance* weit oben im Suchergebnis auftauchen, nicht oder nur schwer von »echten« Suchergebnissen unterschieden werden können.

Im Beispiel in Abbildung 9.21 weißt lediglich die Markierung »Partnerlink« auf die bezahlten Links hin.

Dass eine solche Editierung von Suchergebnissen gegen Geld eine zweischneidige Angelegenheit ist, dürfte einleuchtend sein. Natürlich lassen sich auf diese Weise für die Betreiber von Suchmaschinen weitere Einnahmequellen erschließen, auf der anderen Seite wird dem Benutzer – zumindest bei *pay-for-performance* – ein Suchergebnis präsentiert, von dem er eigentlich erwartet, es sei aufgrund der Indexierung des WWW entstanden; tatsächlich ist dieses Suchergebnis aber das Produkt eines marktwirtschaftlichen Prozesses.

Eine solche Editierung von Suchergebnissen kann letztendlich dazu führen, dass schwächere Unternehmen, nicht profitorientierte Unternehmen

oder Einzelpersonen im Suchergebnis einer Suchmaschine wesentlich weiter unten aufgeführt werden, als sie es eigentlich – aufgrund des Inhalts und der Qualität ihrer Dokumente – verdient hätten. Dadurch verschiebt sich zwangsläufig die Wahrnehmung der Nutzer hinsichtlich der Relevanz bestimmter Dokumente bzw. deren Autoren. Eine solche Entwicklung birgt Gefahren für die im Internet und WWW sonst so hoch eingeschätzte Demokratie und Vielfalt. Andererseits steht eine umfassende Bewertung von *sponsored links* nach den Kriterien von Recall und Precision noch aus. Man könnte hier auch vermuten, dass sich durchaus positive Effekte ergeben könnten, denn jemand, der bereit ist für eine vordere Platzierung in der Ergebnisliste z.B. einen Euro zu bezahlen, wird dies sicher nur dann dauerhaft tun, wenn der Anfragende seine Seite und auch sein Angebot für relevant hält. Sonst wird es kaum zu einem Folgegeschäft für den Anbieter des *sponsored links* kommen.

9.7.3 Bewertungssysteme

Eine Möglichkeit, die Beurteilung der Qualität einer Webseite durchzuführen, besteht in der direkten Beurteilung durch die Leser. Solche Systeme gibt es bereits in vielen Web-Shops oder Portal-Seiten. Dabei können z.B. einzelne Nutzer Produkte beurteilen und andere Benutzer können dann entscheiden, ob diese Beurteilung für sie hilfreich war.

Z.B. bittet Amazon (letzter Abruf: 18.12.2006) Kunden Rezensionen zu Büchern zu schreiben. Um die Qualität dieser Rezensionen wiederum beurteilen zu können, können andere Kunden dann auf die Frage »War diese Rezension für Sie hilfreich?« mit *Ja* oder *Nein* antworten.

Vergleichbare Systeme könnten schon bald die Suche im Internet insgesamt verbessern. Erste Ansätze hierzu sind in den Toolbars der verschiedenen Suchmaschinenanbieter bereits gelegt (vgl. z.B. Informationen zur `Google Toolbar`, letzter Abruf: 18.12.2006).

9.8 Probleme bei Suchmaschinen

Die Verwendung von Suchmaschinen zur Suche nach relevanten Informationen im WWW hat viele Vorteile und verkürzt die Zeit bis zum Auffinden relevanter Dokumente dramatisch. Allerdings enstehen aus der Benutzung von Suchmaschinen auch einige Probleme, die in diesem Abschnitt betrachtet werden sollen:

9.8.1 Staatliche und nicht-staatliche Zensur

9.8.2 Veraltete Indexe

9.8.3 Weblogs und Co.

9.8.1 Staatliche und nicht-staatliche Zensur

Zensur

Eine Thematik, die der Thematik der *sponsored links* nicht unähnlich ist, betrifft die Möglichkeit von staatlicher und nicht-staatlicher Zensur. Es ist für die Betreiber von Suchmaschinen möglich, einzelne Dokumente, Dokumente bestimmter Autoren oder Organisationen oder komplette Bereiche des WWW absichtlich aus ihrem Index zu entfernen bzw. gar nicht erst aufzunehmen.

Eine solche Vorgehensweise führt – in noch schärferem Maße als bei den *sponsored links* – dazu, dass die in den entfernten Dokumenten vorliegenden Informationen den WWW-Nutzern (zumindest über diese Suchmaschine) nicht mehr zugänglich sind. Natürlich kann der Betreiber einer Suchmaschine die betreffenden Dokumente nicht aus dem WWW löschen bzw. den Zugang zu den entsprechenden Servern sperren. Da allerdings die meisten Informationen im WWW heutzutage nicht durch das Verfolgen von Hyperlinks bis zum Ziel, sondern über Suchmaschinen gefunden werden, stellt die Entfernung eines Dokuments aus dem Index einer Suchmaschine eine effektive Möglichkeit dar, den Zugriff auf dieses Dokument zumindest zu erschweren und hinreichend unwahrscheinlich zu machen.

In einigen Ländern – besonders im arabischen und asiatischen Raum – sind einige Suchmaschinen (darunter auch Google) entweder nicht oder nur eingeschränkt zugreifbar. Staatliche Organe leiten den gesamten Internetverkehr dieser Länder über zentrale Router und können an diesen Stellen den Zugriff auf bestimmte Bereiche des Internet sperren. Suchmaschinen sind in solchen Ländern nur dann zugelassen, wenn ihr Index von bestimmten, den jeweiligen Machthabern ungelegenen Inhalten gesäubert wurde. Damit steht den Nutzern in diesen Ländern nur ein eingeschränktes Informationsangebot im WWW zu Verfügung; die jeweilige Regierung hat eine effektive Möglichkeit zur Zensur des WWW.

9.8.2 Veraltete Indexe

Das WWW umfasst zur Zeit mehrere Milliarden Dokumente. Es ist für keine Suchmaschine möglich, das WWW vollständig zu erfassen; die besten Suchmaschinen decken zur Zeit nach Schätzungen etwa ein Drittel des WWW ab. Aber selbst einen solchen, im Vergleich zum WWW kleinen Index können die Suchmaschinen nicht ständig aktuell halten. Große Suchmaschinen schaffen es, täglich mehrere Millionen oder sogar hunderte Millionen von Dokumenten zu indexieren und auf Veränderungen zu überprüfen. Damit dauert es durchschnittlich immer noch mehrere Wochen, bis die Änderung eines Dokuments oder die Publikation eines Dokuments von einer Suchmaschine entdeckt wird.

Die Sicht auf das WWW (oder auch andere Teile des Internet), die eine Suchmaschine einem Benutzer bietet, ist also immer ein veraltetes Abbild des WWW. Den aktuellen Stand des WWW können traditionelle Suchmaschinen nicht abbilden.

9.8.3 Weblogs und Co.

Weblog

Seit einiger Zeit ist es in Mode gekommen, so genannte *Weblogs* (kurz: Blog) zu führen. Diese Weblogs können in etwa mit einem elektronischen Tagebuch verglichen werden, das im WWW veröffentlicht wird. Ein Weblogger (oder kurz: Blogger) dokumentiert Tag für Tag seine Erlebnisse, Gedanken und Gefühle und teilt sie auf diesem Wege der Welt mit. Häufig sind in solchen Weblogs auch Listen mit Hyperlinks zu Dokumenten oder Websites enthalten, die der Blogger für interessant, amüsant oder aus anderem Grund für wichtig hält. Solche Weblogs sind aus einem einfachen Grund für Suchmaschinen ein Problem. Auf der einen Seite gibt es ziemlich viele Weblogs – man geht von einigen Millionen aus, zumal es Portale gibt, die einem die Einrichtung eines Weblogs ohne großen Aufwand und ohne Kosten ermöglichen. In jedem dieser Weblogs kommt es täglich oder beinahe täglich zu Änderungen oder zur Publikation neuer Dokumente, so dass das WWW momentan vor allem auch an dieser Stelle enorm schnell wächst. Auf der anderen Seite hält sich die Leserschaft solcher Weblogs in Grenzen: üblicherweise gibt es – vom Autor abgesehen – keine oder nur eine Handvoll Leser. Dies deutet darauf hin, dass ein jeweiliges Weblog nur von einem sehr kleinen Kreis von Menschen als relevant und lesenswert bewertet wird. Gleichzeitig ist es in Blogger-Kreisen üblich, in seinem Blog auf die Weblogs von Freunden und Bekannten zu verlinken. Dadurch ist die Zahl der Backlinks (etwa

PageRank™

bei der Berechnung des PageRank™) von Weblogs häufig hinreichend groß, um in den Suchergebnissen zu vielen Anfragen sehr weit vorne aufzutauchen.

Den Betreibern von Suchmaschinen stellt sich nun eine wichtige Frage: sollen sie ihre Suchmaschinen täglich eine Vielzahl neuer oder geänderter Dokumente in Weblogs indexieren lassen, die offenbar nur von wenigen Menschen für relevant befunden werden, oder sollen sie ihre Suchmaschine vom Indexieren solcher Weblogs abhalten?

Bei Google etwa hat man für den Bereich des WWW, der sich aus Weblogs zusammensetzt, eine eigene Kategorie eingeführt, also parallel zur Suche im WWW, in Newsgroups und zur Bildersuche auch einen eigenen Bereich zur Suche in Weblogs (www.google.de/blogsearch, letzter Abruf: 18.12.2006).

Neben der schieren Anzahl publizierter, für die Allgemeinheit jedoch wenig relevanter Dokumente besteht noch ein anderer Grund, der gegen eine Aufnahme von Weblogs in den allgemeinen Index von Suchmaschinen spricht. Weblogger reagieren nämlich auf wichtige Ereignisse wesentlich schneller als andere Publizierer im WWW. So publizierten Blogger bereits wenige Minuten nach dem Beginn des Irak-Krieges im Jahr 2003 erste Berichte, während andere Websites (etwa Nachrichtenagenturen, Magazine, Zeitungen, ...) noch einige Zeit mit ersten Berichten auf sich warten ließen. Nimmt man nun Weblogs in den allgemeinen Index einer Suchmaschine auf, so werden zu wichtigen Themen vor allem Weblogs als relevante Quellen in Suchergebnissen aufgeführt. Aufgrund der großen Zahl von Weblogs erscheinen die qualitativ eigentlich hochwertigeren Ressourcen von Nachrichtenagenturen u.ä. erst sehr weit unten im Suchergebnis.

Es ist sicherlich keine einfache Frage, wie eine Suchmaschine mit Blogs umgehen sollte. Zusammenfassend lässt sich aber sagen, dass man Weblogs entweder nicht in den allgemeinen Index einer Suchmaschine aufnehmen oder aber den RSV von Dokumenten aus Blogs durch eine geeignete Maßnahme reduzieren sollte.

Retrieval Status Value, RSV

9.9 Zusammenfassung

In diesem Kapitel wurde der grundsätzliche Aufbau einer Suchmaschine für das WWW vorgestellt, der Crawling-Prozess wurde dargestellt,

typische und weitergehende Funktionen von Suchmaschinen wurden besprochen, ihre Vorteile und die eventuell auftretenden Probleme wurden diskutiert. Als Beispiel für ein Verfahren des Web Information Retrieval wurde der PageRank™-Algorithmus von Google behandelt, inklusive einiger Probleme bei dessen Anwendung und Ansätzen zur Lösung dieser Probleme. Der letzte Abschnitt des Kapitels beschäftigte sich schließlich mit einigen grundsätzlichen Problemen, die aus der Verwendung bzw. während der Benutzung von Suchmaschinen entstehen.

Abschließend sei noch darauf verwiesen, dass es im Web eine Vielzahl guter Informationsquellen zu Suchmaschinen und deren Hintergründen gibt. An dieser Stelle können wir nur einige wenige aufführen:

- Die Suchfiebel (letzter Abruf: 18.12.2006) ist eine deutschsprachige Seite mit Hintergrundinformationen zu Suchmaschinen und Tipps für Suchende und Webmaster.
- Zum Umfeld der Suchfibel gehört auch das Suchlexikon (letzter Abruf: 18.12.2006), das einen sehr umfassenden Überblick über verschiedenste Suchienste gibt.
- SearchEngineWatch (letzter Abruf: 18.12.2006) ist eine englischsprachige Seite, die sich mit Neuigkeiten, die Suchmaschinen betreffen, befasst.
- Search Engine World (letzter Abruf: 18.12.2006) bietet in Form von Foren viele interessante Informationen zu Suchmaschinen.
- Suchmaschinentricks (letzter Abruf: 18.12.2006) wendet sich primär an Webadministratoren, die dafür sorgen wollen, dass ihre Seiten in den Rankings der Suchmaschinenbetreiber weit oben landen.

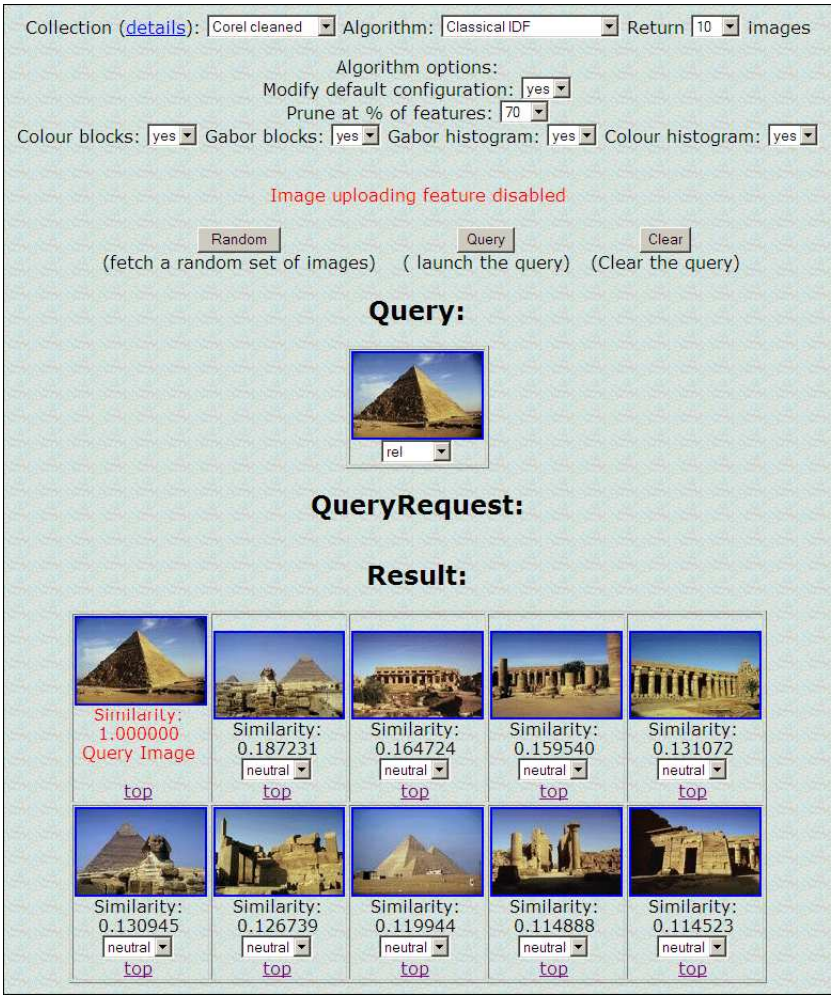


Abbildung 9.18 — Das Ergebnis einer Bildsuche in Viper. Viper liefert zu einem Anfragebild eine Reihe von Bildern aus der Kollektion, die diesem ähneln

Abbildung 9.19 — Die Metasuchmaschine *MetaGer* bietet eine Metasuche über (meist) deutschsprachige Suchmaschinen.

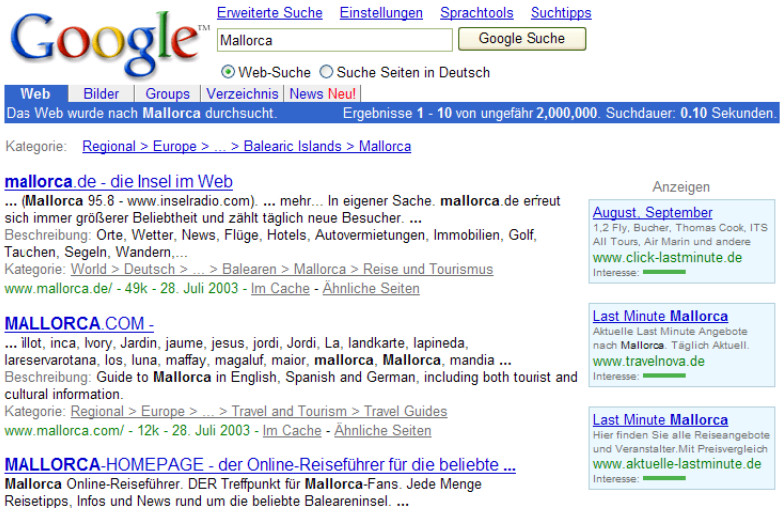


Abbildung 9.20 — Sponsored links bei Google: Bei der Suche nach dem Begriff »Mallorca« werden neben den relevanten Dokumenten (links) auch Links zu den Websites von Unternehmen angezeigt (rechts), die einen sponsored link zum Suchbegriff erworben haben.

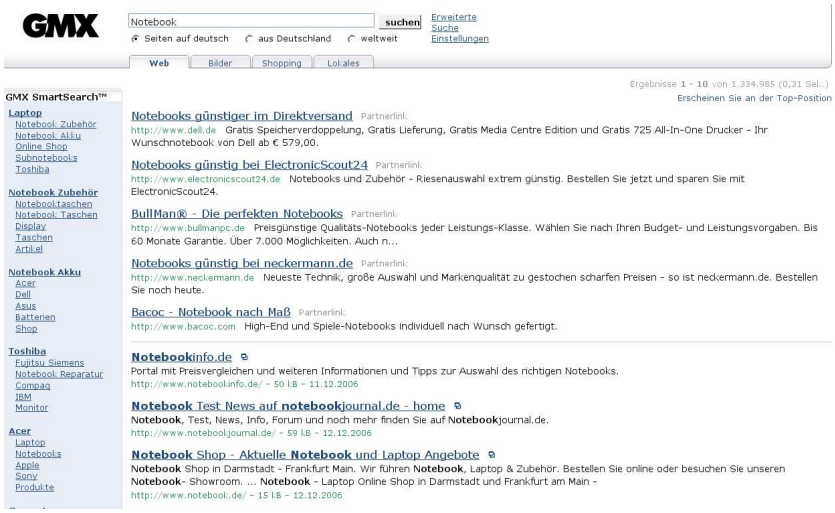


Abbildung 9.21 — Ergebnis einer Suche nach »Notebook« bei **GMX**

10 Neuere Entwicklungen und innovative Anwendungen

Neuere Entwicklungen und innovative Anwendungen von Information Retrieval Systemen

Wir haben im vorliegenden Kurs Modelle, Methoden, Algorithmen und Anwendungen des Information Retrieval betrachtet. Leider kann aber ein Kurs des gegebenen Umfangs nicht alle Aspekte abdecken. An dieser Stelle soll daher nur ganz kurz als Perspektive auf drei weitere aktuelle Forschungsfelder des Information Retrieval hingewiesen werden:

- **Cross-Language Information Retrieval**
  Das WWW und auch viele andere Dokumentenkollektionen enthalten Dokumente in verschiedenen Sprachen. Wenn auf einer solchen Kollektion nun eine Anfrage in einer bestimmten Sprache gestellt wird, dann sind verschiedene Szenarien denkbar: Entweder der Anfragende ist nur an Dokumenten in der Sprache interessiert, in der auch die Anfrage gestellt wurde oder aber er wünscht Dokumente in verschiedenen Sprachen im Ergebnis. Aus einer solchen Situation ergeben sich unmittelbar Aufgabenstellungen wie die Bestimmung der Sprache, in der eine Anfrage oder ein Dokument geschrieben ist oder die »Übersetzung« von Anfragen oder Dokumenten.

- **Geographisches Information Retrieval**
  Im klassischen Information Retrieval werden Suchanfragen gestellt, die sich auf den Inhalt von Dokumenten beziehen. Das geographische IR bezieht daneben eine räumliche Komponente ein, mit der nur solche Webseiten gefunden werden sollen, die einen bestimmten geographischen Kontext besitzen. Dazu müssen bestehende IR-Systeme um entsprechende Komponenten erweitert werden, die den geographischen Kontext einer Webseite erkennen, geographische Anfragen verarbeiten und bei der Ergebnispräsentation die räumlichen Zusammenhänge darstellen können.

- **Verteiltes Information Retrieval in Peer-to-Peer-Netzen**
  Peer-to-Peer-Netze (P2P-Netze) erfreuen sich zunehmender Beliebtheit zur verteilten und administrationsarmen Verwaltung größerer Dokumentenkollektionen. Ein wichtiges Problem ist dabei die Suche und insbesondere die inhaltsbasierte Suche nach Dokumenten in P2P-Netzen. Dabei sind Fragen der Replikation von Dokumenten auf verschiedenen Peers ebenso zu betrachten wie die Frage, an welche Peers eine Anfrage weitergeleitet werden sollte um möglichst gute Anfrageergebnisse zu erzielen.

Diese Bereiche machen deutlich, dass im Feld des Information Retrieval viele neue und wichtige Herausforderungen bestehen. Dies betrifft die Suche nach Textdokumenten ebenso wie die Suche nach multimedialen Inhalten.

Literaturverzeichnis

[ABH97] Apers, P. (Hrsg.) ; Blanken, H. (Hrsg.) ; Houtsma, M. (Hrsg.): *Multimedia Databases in Perspective*. Springer-Verlag, 1997 8.7

[AC97] Analyti, A. ; Christodoulakis, S.: Content-Based Querying. In: *Multimedia Databases in Perspective*. Springer-Verlag, 1997, S. 145–180 8.33, 8.34

[ACL05] Allan, James ; Carterette, Ben ; Lewis, Joshua: When will information retrieval be »good enough«? In: *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*. New York, NY, USA : ACM Press, 2005. – ISBN 1–59593–034–5, S. 433–440 2.5

[BK98] Bässmann, Henning ; Kreyss, Jutta: *Bildverarbeitung Ad Oculos*. 3., vollst. überarb. u. erw. Aufl. Springer-Verlag, 1998. – Taschenbuch 8.3.3

[BL85] Buckley, C. ; Lewit, A.: Optimization of inverted vector searches. In: *In Proceedings of the 8th International ACM SIGIR Conference on Research and Development in Information Retrieval*. New York, 1985, S. 97–105 5.5.1, 5.5.1, 5.5.2

[BL98] Berners-Lee, Tim: *A roadmap to the Semantic Web*. Version: September 1998. http://www.w3.org/DesignIssues/Semantic.html 3.7.2

[BL00] Berners-Lee, Tim: *Semantic Web on XML*. Version: Dezember 2000. http://www.w3.org/2000/Talks/1206-xml2k-tbl 3.4

[BL04] Berners-Lee, Tim: *Design Issues. Architectural and philosophical points.* Version: 1997-2004. http://www.w3.org/DesignIssues/ 3.7.2

[BLFM98] Berners-Lee, Tim ; Fielding, Roy ; Masinter, Larry: *RFC2396: Uniform Resource Identifiers (URI): Generic Syntax.* Version: August 1998. http://www.ietf.org/rfc/rfc2396.txt 3.7.2

[BP98] Brin, Sergey ; Page, Lawrence: The Anatomy of a Large-Scale Hypertextual Web Search Engine. In: *Computer Networks and ISDN Systems* 30 (1998), Nr. 1-7, 107-117. citeseer.ist.psu.edu/brin98anatomy.html 9.7.1

[BPSM+04] Bray, Tim ; Paoli, Jean ; Sperberg-McQueen, C. M. ; Maler, Eve ; Yergeau, François: *Extensible Markup Language (XML) 1.0 (Third Edition).* Version: 4. Februar 2004. http://www.w3.org/TR/REC-xml/ 3.7.2

[Buc89] Buchanan, Brian: *Bibliothekarische Klassifikationstheorie.* München : Saur, 1989 6.1, 6.4

[BY92] Baeza-Yates, Ricardo: String Searching Algorithms. In: Frakes, William B. (Hrsg.) ; Baeza-Yates, Ricardo (Hrsg.): *Information Retrieval: Data Structures and Algorithms.* Englewood Cliffs, NJ, USA : Prentice-Hall, 1992, S. 219–240 4.2

[BYRN99] Baeza-Yates, Ricardo ; Ribeiro-Neto, Berthier: *Modern Information Retrieval.* Addison-Wesley-Longman, 1999 7.1, 7.1

[Can86] Canny, J.: A computational approach to edge detection. In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* 8 (1986), November, Nr. 6, S. 679–698 8.3.3

[CMK66] Cleverdon, Cyril ; Mills, Jack ; Keen, Michael: *Factors Determining the Performance of Indexing Systems: ASLIB Cranfield Research Project.* Bd. Volume 1: Design. Cranfield : ASLIB Cranfield Research Project, 1966 2.4

[Cro90] Crouch, J.C.: An approach to the automatic construction of global thesauri. In: *Information Processing and Management* 26 (1990), Nr. 5, S. 629–640 3.2

[DDF+90] Deerwester, Scott ; Dumais, Susan ; Furnas, Goerge ; Landauer, Thomas ; Harshman, Richard: Indexing by latent semantic analysis. In: *Journal of the American Society for Information Science* 41 (1990), Nr. 6, S. 391–407 7.3.2

[DK01] Dodge, Martin ; Kitchin, Rob: *Atlas of Cyberspace*. Addison Wesley, 2001. – (ISBN 0-201-74575-5) 6.3.3, 6.4

[DRA98] Dugad, R. ; Ratakonda, K. ; Ahuja, N.: Robust Video Shot Change Detection. In: *IEEE Workshop on Multimedia Signal Proc.*, 1998, S. 376–381 8.5.1

[DS04] Dean, Mike ; Schreiber, Guus: *OWL Web Ontology Language Reference*. Version: Februar 2004. http://www.w3.org/TR/owl-ref/ 3.7.2

[FB91] Fuhr, Norbert ; Buckley, Chris: A probabilistic learning approach for document indexing. In: *ACM Transactions on Information Systems* 9 (1991), Juli, Nr. 3, S. 223–248. – Special Issue on Research and Development in Information Retrieval 7.1

[FBY92] Frakes, William B. ; Baeza-Yates, Ricardo: *Information Retrieval: Data Structures and Algorithms*. Englewood Cliffs, NJ 07632, USA : Prentice-Hall, 1992 1.1.2, 4.6

[FDST98] Faudemay, P. ; Durand, G. ; Seyrat, C. ; Tondre, N.: Indexing and retrieval of multimedia objects at different levels of granularity. In: *Proc. SPIE - Multimedia Storage and Archiving Systems III* Bd. 3527. Boston, November 1998, S. 112–121 8.31

[FEF04] F. Estrada, A. J. ; Fleet, D.: *SIFT matlab tutorial: http://www.cs.toronto.edu/ jepson/csc2503/tutSIFT04.pdf*. 2004. – 9.10.2007 8.18, 8.19, 8.22, 8.23, 8.25

[Fer03] Ferber, Reginald: *Information Retrieval – Suchmodelle und Data-Mining-Verfahren für Textsammlungen und das Web*. Heidelberg : dpunkt.verlag, 2003

[Fil81] Fill, Karl: *Einführung in das Wesen der Dezimalklassifikation*. 4. Aufl. Berlin : Beuth, 1981 6.1.2

[Fra92] Frakes, William B.: Stemming algorithms. 1992, S. 131–160 3.6

[FTZ04] Fang, Hui ; Tao, Tao ; Zhai, ChengXiang: A formal study of information retrieval heuristics. In: *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, 2004. – ISBN 1–58113–881–4, S. 49–56 7.2

[Fuh96] Fuhr, Norbert: Skriptum Information Retrieval. Lecture Notes on Information Retrieval / Universtität Dortmund. 1996. – Vorlesungsskript 2.2.2, 4.2.4

[Gev01] Gevers, Theo: Color-Based Retrieval. In: Lew, Michael S. (Hrsg.): *Principles of Visual Information Retrieval*. London : Springer, 2001 (Advances in Pattern Recognition), S. 11–49 8.3.1

[GGL] *Google Website*. `www.google.de` 9.1

[Gul88] Gulbins, J.: *UNIX: Eine Einführung in Begriffe und Kommandos*. 3. Springer-Verlag, 1988 (Springer Compass) 3.3.3

[Har96] Harter, Stephan: *Online Information Retrieval. Concepts, Principles, and Techniques*. Orlando, FL : Academic Press, 1996 1.1.2

[Hen96] Henrich, A.: Adapting a Spatial Access Structure for Document Representations in Vector Space. In: *CIKM '96, Proceedings of the Fifth International Conference on Information and Knowledge Management*. Rockville, Maryland, USA : ACM, 1996, S. 19–26 5.5

[Hil98] Hildebrand, Lars: Unscharfe Farbverarbeitung. In: *Tagungsband zum 18. Workshop Interdisziplinäre Methoden der Informatik*. Dortmund : Universität Dortmund, 1998 8.3.1

[HTP+00] Hersh, William ; Turpin, Andrew ; Price, Susan ; Chan, Benjamin ; Kramer, Dale ; Sacherek, Lynetta ; Olson, Daniel: Do batch and user evaluations give the same results? In: *SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*. New York, NY, USA : ACM Press, 2000. – ISBN 1–58113–226–3, S. 17–24 2.5

[ID89] Inselberg, Alfred ; Dimsdale, Bernard: Visualizing multi-variate relations with parallel coordinates. In: *Proceedings of the third international conference on human-computer interaction, Vol.1 on Work with computers: organizational, management, stress and health aspects*, 1989, S. 460–467 6.10

[JJC04] Jeremy J. Carroll, Graham K.: *Resource Description Framework (RDF): Concepts and Abstract Syntax*. Version: Februar 2004. http://www.w3.org/TR/rdf-concepts/ 3.7.2

[KJP77] Knuth, Donald E. ; Jr., James H. M. ; Pratt, Vaughan R.: Fast Pattern Matching in Strings. In: *SIAM Journal on Computing* 6 (1977), Nr. 2, S. 323–350 4.1

[KM97] Kowalski, Gerald J. ; Maybury, Mark T.: *Information Retrieval Systems: Theory and Implentation*. Boston / Dordrecht / London : Kluver Academic Publishers, 1997 6.4

[Knu73] Knuth, D. E.: *The Art of Computer Programming III: Sorting and Searching*. Reading, Massachusetts : Addison-Wesley, 1973 4.2.3

[Kom75] Komitee Terminologie und Sprachfragen (KTS) der Deutschen Gesellschaft für Dokumentation e.V. (DGD) (Hrsg.): *Terminologie der Information und Dokumentation*. München : Verlag Dokumentation, 1975 3.5

[Kuh77] Kuhlen, Rainer: *Experimentelle Morphologie in der Informationswissenschaft*. München : Verlag Dokumentation, 1977 3.3.1, 3.3.1

[Kuh90] Kuhlen, Rainer: Zum Stand pragmatischer Forschung in der Informationswissenschaft. In: Herget, J. (Hrsg.) ; Kuhlen, R. (Hrsg.): *Pragmatische Aspekte beim Entwurf und Betrieb von Informationssystemen. Proceedings des 1. Internationalen Symposiums für Informationswissenschaft*. Konstanz : Universitätsverlag Konstanz, 1990, S. 13–18 1.1.1

[Kun99] Kunze, John: *Encoding Dublin Core in HTML*. Version: 1999. http://www.ietf.org/rfc/rfc2731.txt. – (Stand 1.12.2003) 3.7.1

[LAN87] *Langenscheidts Groß es Schulwörterbuch Englisch-Deutsch*. 17. Berlin und München : Langenscheid KG, 1987 1.1.1

[Lew01] Lew, Michael S. (Hrsg.): *Principles of Visual Information Retrieval*. London : Springer, 2001 (Advances in Pattern Recognition) 8.7

[LJZ01] Lu, Lie ; Jiang, Hao ; Zhang, HongJiang: A robust audio classification and segmentation method / Microsoft Research. 2001. – Tech. Rep. 8.4.1, 8.4.1, 8.4.3

[Lov68] Lovins, Julie B.: Developing of a stemming algorithm. In: *Mechanical Translation and Computational Linguistics* 11 (1968), März, Nr. 1 3.3.4

[Low04] Lowe, David G.: Distinctive Image Features from Scale-Invariant Keypoints. In: *Int. Journal of Computer Vision* 60 (2004), Nr. 2, S. 91–110. – ISSN 0920–5691 8.3.4, 8.20, 8.21, 8.24

[Lus86] Lustig, Gerhard: *Linguistische Datenverarbeitung*. Bd. 5: *Automatische Indexierung zwischen Forschung und Anwendung*. Hildesheim : Olms, 1986 3.4.2

[Mei97] Meiss, Brigitte: *Information Retrieval und Dokumentenmanagement im Multimedia-Zeitalter*. Frankfurt am Main : Deutsche Gesellschaft für Dokumentation, 1997 (Reihe Informationswissenschaft der DGD) 6.1

[MH04] McGuinness, Deborah L. ; Harmelen, Frank van: *OWL Web Ontology Language Overview*. Version: Februar 2004. http://www.w3.org/TR/owl-features/ 3.7.2

[MM97] Mayfield, James ; McNamee, Paul: N-Grams vs. Words as Indexing Terms. In: *TREC-6 Conference Notebook Papers*. Gaithersburg, Maryland, USA, 1997 3.6, 5.6

[MRC91] Mackinlay, Jock D. ; Robertson, George G. ; Card, Stuart K.: The perspective wall: detail and context smoothly integrated. In: *Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology*, 1991, S. 173–179 6.9

[OMK91] Ogawa, Y. ; Morita, T. ; Kobayashi, K.: A fuzzy document retrieval system using the keyword connection matrix and a learning method. In: *Fuzzy Sets and Systems* 39 (1991), S. 163–179 4.5

[OW02] Ottmann, Thomas ; Widmayer, Peter: *Algorithmen und Datenstrukturen*. Spektrum Akademischer Verlag, 2002 4.1, 4.2.1

[Por80] Porter, M. F.: An Algorithm for Suffix Stripping. In: *Program* 14 (1980), Nr. 3, S. 130–137 3.3.4

[PSHH04] Patel-Schneider, Peter F. ; Hayes, Patrick ; Horrocks, Ian: *OWL Web Ontology Language Semantics and Abstract Syntax*. Version: Februar 2004. http://www.w3.org/TR/owl-semantics/ 3.7.2

[PZM96] Pass, G. ; Zabih, R. ; Miller, J.: Comparing images using color coherence Vectors. In: *ACM Conference on Multimedia*. Boston, MA, USA, 1996 8.3.1

[Ras92] Rasmussen, Edie: Clustering Algorithms. In: Frakes, William B. (Hrsg.) ; Baeza-Yates, Ricardo (Hrsg.): *Information Retrieval: Data Structures and Algorithms*. Englewood Cliffs, NJ, USA : Prentice-Hall, 1992, S. 419–442 6.2, 6.4

[RF96] Rusch-Feja, Diann: *Metadata-Tags zur Erschließ ung von Internetquellen: Metadata-Elemente des Dublin Core*. Version: 1996. http://www.mpib-berlin.mpg.de/DOK/metatagd.htm. – (Stand 1.12.2003) 3.7.1, 3.6

[Rij79] Rijsbergen, C.J. van: *Information Retrieval*. 2. London : Butterworths, 1979. – Online Ausgabe 1.3, 2.2

[RJ76] Robertson, S. E. ; Jones, K. S.: Relevance weighting of search terms. In: *Journal of the American Society for Information Sciences* 27 (1976), Mai-Juni, Nr. 3, S. 129–146 7.1

[RMC91] Robertson, George G. ; Mackinlay, Jock D. ; Card, Stuart K.: Cone Trees: animated 3D visualizations of hierarchical information. In: *Proceedings of the SIGCHI con-*

ference on Human factors in computing systems: Reaching through technology, 1991, S. 189–194 6.11

[RWHB98] Robertson, Stephen E. ; Walker, Steve ; Hancock-Beaulieu, Micheline: Okapi at TREC-7: automatic ad hoc, filtering, VLC and interactive track. In: *The Seventh Text REtrieval Conference (TREC-7), National Institute of Standards and Technology (NIST)*, 1998, S. 199–210. – http://trec.nist.gov/pubs/trec7/t7_proceedings.html 7.2

[Sal71] Salton, G.: *The SMART Retrival System - Experiments in Automatic Document Processing*. Englewood Cliffs, NJ, USA : Prentice Hall Inc., 1971 5

[SB88] Salton, G. ; Buckley, C.: Term-weighted approaches to automatic text retrieval. In: *Inf. Proc. & Mngmt.* 24 (1988), Nr. 5, S. 513–523 5.2, 5.2

[SB90] Salton, G. ; Buckley, C.: Improving retrieval performance by relevance feedback. In: *J. Amer. Soc. for Information Sci.* 41 (1990), Nr. 4, S. 288–297 5.4, 5.4, 7.1

[SBM96] Singhal, Amit ; Buckley, Chris ; Mitra, Mandar: Pivoted document length normalization. In: *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Experimental Studies*, 1996, S. 21–29 5.3, 5.3

[Sch02] Schmitt, Ingo: Retrieval in Multimedia-Datenbanksystemen. In: *Datenbank Spektrum* (2002), Nr. 4, S. 28–35 8.2

[SGR97] Stickel, Eberhard (Hrsg.) ; Groffmann, Hans-Dieter (Hrsg.) ; Rau, Karl-Heinz (Hrsg.): *Gabler Wirtschaftsinformatik-Lexikon*. Dr. Th. Gabler Verlag, 1997 1.1.1, 6.1, 6.3.1

[Shn98] Shneiderman, Ben: *Designing the User Interface - Strategies for Effective Human-Computer Interaction*. 3. Reading, Massachusetts, USA : Addison-Wesley, 1998 6.3.2, 6.3, 6.3.3, 6.4

[SHS99] Seaborn, M. ; Hepplewhite, L. ; Stonham, J.: Fuzzy Colour Category Map for Content Based Image Retrieval. In: Pridmore, Tony (Hrsg.) ; Elliman, Dave (Hrsg.): *Electronic Proceedings of the Tenth British Machine Vision Conference*, http://www.bmva.ac.uk/bmvc/1999/index.html, 1999 8.3.1

[SL68] Salton, G. ; Lesk, M.E.: Computer evaluation of indexing and text processing. In: *Journal of the ACM* 15 (1968), January, Nr. 1, S. 8–36 5

[SL01] Sebe, Nicu ; Lew, Michael S.: Texture Features for Content-Based Retrieval. In: Lew, Michael S. (Hrsg.): *Principles of Visual Information Retrieval*. London : Springer, 2001 (Advances in Pattern Recognition), S. 51–85 8.3.2

[SM83] Salton, G. ; McGill, M.J.: *Information Retrieval-Grundlegendes für Informationswissenschaftler*. McGraw-Hill Book Company GmbH, 1983 1.4.2

[SP93] Schmidt, Michael ; Pfeifer, Ulrich: Eignung von Signaturbäumen für Best-Match-Anfragen. In: Knorz, Gerhard (Hrsg.) ; Krause, Jürgen (Hrsg.) ; Womser-Hacker, Christa (Hrsg.): *Information Retrieval '93: Von der Modellierung zur Anwendung, Proceedings der 1. Tagung »Information Retrieval«*. Regensburg : Universitätsverlag Konstanz, September 1993 (Schriften zur Informationswissenschaft 12), S. 125–138. – ISBN 3-87940-473-9 5.5

[Ste00] Steinmetz, Ralf: *Multimedia-Technologie - Grundlagen, Komponenten und Systeme*. 3. überarbeitete Auflage. Berlin : Springer, 2000 8.1, 8.3.1, 8.3.3, 8.7

[Str94] Strzalkowski, Tomek: Robust text processing in automated information retrieval. In: *Proceedings of the Fourth ACL Conference on Applied Natural Language Processing (13-15. Oktober 1994, Stuttgart)*, Association for Computational Linguistics, Oktober 1994 3.4.1

[Sun90] Sunday, Daniel: A Very Fast Substring Search Algorithm. In: *Communications of the ACM (CACM)* 33 (1990), Nr. 8, S. 132–142 4.1

[SW95] Sturges, Julia ; Whitfield, T.W. A.: Locating basic colours in the Munsell Space. In: *Color Research and Application* 20 (1995), Nr. 6, S. 364–376 8.8

[SWY75] Salton, G. ; Wong, A. ; Yang, C.S.: A vector space modell for automativ indexing. In: *Communications of the ACM (CACM)* 18 (1975), November, Nr. 11, S. 613–620 5

[VH98] Voorhees, Ellen M. ; Harman, Donna: Overview of the Seventh Text REtrieval Conference (TREC-7). In: *Proceedings of the Seventh Text REtrieval Conference (TREC-7)*. Gaithersburg, Maryland, USA : NIST Special Publication 500-242, 1998, S. 1–24 2.3.1

[VH01] Veltkamp, Remco C. ; Hagedorn, Michiel: State of the Art in Shape Matching. In: Lew, Michael S. (Hrsg.): *Principles of Visual Information Retrieval*. London : Springer, 2001 (Advances in Pattern Recognition), S. 87–119 8.3

[Vic70] Vickery, B.C.: *Techniques of information retrieval*. London : Butterworth, 1970 2.4

[W3C] W3C, World Wide Web Consortium (Hrsg.): *Extensible Markup Language (XML)*. http://www.w3.org/XML/ 3.7.2

[WBKW96] Wold, E. ; Blum, T. ; Keislar, D. ; Wheaton, J.: Content-based classification, search and retrieval of audio. In: *IEEE Multimedia* 3 (1996), Nr. 2, S. 27–36 8.4.1, 8.4.1

[ZKS93] Zhang, H. J. ; Kankanhalli, A. ; Smoliar, S.W.: Automatic Partitioning of Full Motion Video. In: *Multimedia Systems* 1 (1993), Januar, S. 10–28 8.5.1, 8.32

Index

n-gram, 121

Abbildung auf die Grund- und Stammform, 100
Abgleich mit externen Quellen, 71
Abstraktionsrelation, 119
Active Server Pages, ASP, 357
ad hoc task, 80
ADI-Kollektion, 86
Algorithmus von Porter, 109
Algorithmus, Lovins, 105
Algorithmus, regelbasiert, 105
Anfangsadresse, 147
Anfragebearbeitung, 31
Anfrageerweiterung, 70, 100, 119
Anfrageformulierung, 38
Anfragesprache, 38
Ansatz, nutzerorientierter, 72
Avarage Zero-Crossing Rate, 329
Average Link, 238, 246

B-Baum, 149
back link, 387
Begriff, 93
Benutzbarkeit, 90
Berechnung des PageRank, 390
Beschreibungsvektor, 46
Best Match, 34
Best Match Suche, 35
Bestandsrelation, 119
Beugungsform, 99
Bildsuche im Internet, 383
Binary Independence Retrieval, 263
Binary Large Object, BLOB, 336
Binärsignaturen, 152
Bitscheiben, 163
Bitscheibenorganisation, 163
Blocknummer, 147
BM 25, 280
Boolescher Ausdruck, 38
Boolesches Retrieval, 41, 139, 170, 185
Breitensuche, 360
Browsing, 32, 217, 245, 247
Buckley & Lewit, Algorithmus nach, 211

CACM-Kollektion, 86
CISI-Kollektion, 86, 87
Clique-Technik, 236, 237
Cluster, 232, 249, 259
  Aktualisierung, 236
  Erzeugung, 236
Cluster-Analyse, 232
Cluster-Ansätze, 217
Clusterbildung
  agglomerative Verfahren, 235, 237
  hierarchische Verfahren, 235, 237
  nichthierarchische Verfahren, 235
  teilende Verfahren, 235, 237

Clusterverfahren, 32
CMY-Modell, 297
Common Gateway Interface, CGI, 357
Complete Link, 238, 246
Content Management System, CMS, 367
Coordination Level Match, 176
Cosinus-Maß, 234
Cosinusmaß, 194, 285
Coverage, 89
Coverage of the Collection, 89
Crawler, 350, 351, 360

Dangling Links, 390
Darmstädter Indexierungsansatz, 112
DARPA, 352
Darstellung des Wissens, 25
Dateikennung, 147
Dateityp, Einschränkung der Suche, 378
Daten, 18
Datenbank, 33
Datum, Einschränkung der Suche, 377
DBMS, 33
Deduktion, 37
Deep Web, 358
Deklination, 99
Dendogramm, 239, 245
Deskriptor, 115, 116, 118
Determinante, 321
deterministisch, 37
Dezimalklassifikation, 223
  Verknüpfung einzelner Klassen, 226
Dice-Koeffizent, 234
disjunktive Anfrage, 156
Dokument-Source-Methode, 69
Dokumentenrepräsentation, 40
Domain, Einschränkung der Suche, 378
Dublin Core, 122

Effizienz, 90
Effort, 89
Eigenwert, 321
Einfügen im S-Baum, 167
Einheitsvektoren, 48
electronic mail, eMail, 353, 381
Energie, 327
Ergebnisqualität, 60, 90
Ergebnisranking, 392
Ergebnisrepräsenation, 375
Erkennung von Musik, 331
Erkennung von Sprache, 332
Erkennung von Stille, 331
Euklidischer Abstand, 234
Evaluierung, 59
Evaluierungsmaße, 59
Exact Match, 33
Extensible Markup Language (XML), 126
eXtensible Markup Language, XML, 369
Extraktion, 20
Extranet, 354

F-Measure, 67
Facettierung, 225, 246, 248
false drops, 151, 156
Faltung, 317
Faltung eines Bildes, 311
Farbhistogramm, 300
Farbmodelle, 296
Feature, lokal, 317
Feature-Extraktion, 295
Feature-Vektor, 324
Fehlerwahrscheinlichkeit, 158
File Transfer Protocol, FTP, 353, 381

Filter, 369
Flexionsform, 98
Form of Presentation, 89
forward link, 387
Frame, 371
Funktionsumfang, 59
Fuzzy-Set, 179
Fuzzy-Set Modell, 178

Gift, GNU Image Finding Tool, 385
Google, 350
Gradient, 323
Grauwertübergangsmatrix, 305
Grund- und Stammformreduktion, 102
Grund- und Stammformreduzierung, 95
Grundform, formale, 102
Grundform, lexikographische, 102

Hash-Funktion, 150
Hauptkrümmung, 321
Hesse-Matrix, 321
high precision task, 80
Hilfsdatenstruktur, 205
Hilfsdatenstruktur, Realisierung der, 209
Hintergrundspeicher, 144
Histogramm, 323
Hits, 65
Homogenität, 306
Homograph, 117
HSV-Modell, 298
HTML-Parser, 365
Hypertext Markup Language, HTML, 355, 365
Hypertext Transfer Protocol, HTTP, 356

Ide (dec hi), 201
Ide (regular), 201
IDF, 48, 281
Imagemap, 373
Index, 350
Indexierer, 351
Indexierungssprache, 115
Indexierungsvokabular, 187
Indexstruktur, 140
Induktion, 37
inexakter Filter, 151
Inferenz, 36
Information, 18, 19
Information Filtering, 32
Information Retrieval, 11, 18
Information Retrieval Systeme, 17
Informationsvisualisierung, 249, 259
  1-dimensionale lineare Darstellung, 251
  2-dimensionale Darstellung, 251
  3-dimensionale Darstellung, 254
  baumartige Darstellung, 258
  Darstellung als Netzwerk, 258
  multidimensionale Darstellung, 254
  zeitliche Darstellung, 254
Informationswunsch, 32, 38, 53, 199
inhaltsbasierte Suche, 15
INSPEC-Kollektion, 86, 212
Internationale Patentklassifikation, 220
Internet Relay Chat, IRC, 353
Interpretationsobjekte, 342
Intranet, 354
invertierte Liste, 172, 235

Invertierte Listen, 140
IR-Modell, 137

Jaccard-Koeffizent, 234
Java Server Pages, JSP, 356
JavaScript, 373

Kataloge im Internet, 227
Key Frame Extraktion, 335
Key-Frame, 333
Keypoint, 318
Klassifikation, 31, 217, 218, 259
    automatische, 259
    Einordnung hierarchischer, 229
    Einsatzgebiete, 231
Klassifikationsschemata, 28
Klassifikationssysteme
    monohierarchisch, 232
    polyhierarchisch, 232
Klassifkationen, 248
Konjugation, 99
konjunktive Anfrage, 156
Konnotation, 116
Kontrast, 305, 319
Konvertierung, 295
Konzept, 93
Korrekturfaktor, 190
Kriterienkataloge, 88

Land, Einschränkung der Suche, 378
Latent Semantic Indexing, 282
Laufzeiteffizienz, 60, 61
Lautstärke, 328
Link Extraction, 367
Link-Farm, 391
Literal, 130

Mailing-Liste, 382
Makrobewertung, 72
Manipulation, 391
Matchingalgorithmus, 96
Medien-Objekt, 336
Medlars-Kollektion, 86
Mehrwortgruppe, 95, 110
Metasuchmaschine, 385
Mikrobewertung, 73
Misses, 65
Mittelwertbildung, 71
Modell, 37, 137
monothetisch, 37
Morphem, 117
Morphologische Zerlegung, 117
Mosaicing, 335
Motion Detection, 334
Multimedia, 30, 291
Multimedia-Datenbanken, 335
Multimedia-Dokument, 336
Multimedia-Objekt, 336
Munsell-Modell, 299

NEAR-Operator, 175
Network News Transport Protocol, NNTP, 381
Newsgroup, 381
Newsgroups, 353
Noise, 65
Normierung, 188
Normierung, Anpassung der, 197

Okapi, 280
Oktave, 318
Ontologie, 126
Optical Character Recognition, OCR, 295
Optical Flow, 334

PageRank™, 375, 387, 398
Parser, 111, 369
Partial Match, 34
Pattern Matching, 27, 172
Pattern-Matching, 138, 139, 151

pay-for-performance, 394
Phrasen, 376
Pixeldifferenz, 320
Plain Text, 369
Platzbedarf, 144
Polysem, 117
Polysemkontrolle, 117
polythetisch, 38
Pool, 83
Pooling, 70, 71
Pooling Method, 83
Portable Document Format, PDF, 370
Position, Einschränkung der Suche, 379
Postkoordination, 115
PostScript, PS, 370
Precision, 55, 63, 89, 90, 96, 100, 101, 110, 111, 120, 386
Presentation, 90
probabilistisch, 37
Probabilistisches Information Retrieval, 52
probabilistisches Lernen, 279
Proof, 127
Prozess des Wissenstransfers, 22
Präkombination, 114
Präkoordination, 114
Pseudo-Relevance Feedback, 275
Pulse Code Modulation, 325

Qualität von IR-Systemen, 26
Quasi-Synonym, 116
Query by Example, 385
Query Engine, 351

Random Surfer Modell, 390
Rank Sinks, 391
Ranking, 45, 74
RDF-Schema, 132
Reallocation-Technik, 236, 246
Realweltobjekt, 342
Recall, 55, 63, 88, 90, 96, 100, 110, 111, 120, 386
Recall-Punkt, 75
Recall/Precision-Graph, 56, 67
Region Growing, 315
Registrierungsdaten, 341
regulärer Ausdruck, 103
Rejected, 65
Relevance Feedback, 24, 40, 200, 266
Relevanz, 62
Relevanzbeurteilung, manuell, 69
Relevanzbeurteilung, repräsentative Stichprobe, 69
Relevanzeinstufung, binäre, 62
Relevanzskala, 56
Repräsentation, 19, 94
Repräsentation des Informationsbedürfnisses, 24
Resource Description Framework (RDF), 126, 127
Response, 90
Retrieval, 21
Retrieval Status Value, RSV, 393, 398
Retrieval-Status-Wert, 267
RGB-Modell, 296
Robot, 361
Robots Exclusion Standard, 364
Rocchio, 202

Satzform, 111
Schnittmenge der Listen, 173
Schnittmenge mehrerer Listen, 143
Schreibweisenvariante, 116
Segmentierung, 295, 308, 332
(teil-)automatisch, 309

inhaltsbasierte, 330
kantenorientierte, 311
manuell, 309
punktorientierte, 310
regionenorientierte, 315
Seiten, 147
Semantic Web, 125
Semantic Web, Schichtenarchitektur, 125
Semantische Zerlegung, 118
Servlets, 356
Shallow Web, 358
Shot-Detection, 333
Sichtweise, systemorientierte, 73
SIFT, 316
Signatur, 150, 172
Signatur, digitale, 127
Signaturen, 149
Signaturgewicht, 153
Single Link, 238, 246
Singularwertzerlegung, 282
Sitemap, 373
Skalarprodukt, 46, 186, 193
Skalenraum, 317
slope-Wert, 198
Sobel-Operatoren, 313
Speicherplatzeffizienz, 60
Spider, 361
Splitten eines Knotens, 168
sponsored link, 394
Sprache, Einschränkung der Suche, 379
Sprachstil, 116
Spur, 321
SQL, 60
SQL-Anfrage, 36
Stammform, 102
Star-Technik, 236, 243
Stoppwort, 41, 95
Stoppworteliminierung, 95
Strukturierte Dokumente, 29
Suche im S-Baum, 166
Suchmaschine, 350
Suchmuster, 138
Superimposed Coding, 153
Synonym, 95, 120
Synonymkontrolle, 116

Tasks, 80
Taylorreihe, 319
Term, 41, 93
Term-Term-Matrix, 180
Terminologische Kontrolle, 115
Texturanalyse, 303
tf·idf-Formel, 191
tf·idf-Formel, Variante der, 195
Thesaurus, 28, 95, 116, 118, 233
automatische Erzeugung, 243
Thesaurusrelationen, 118
Tiefensuche, 360
Time Lag, 89
Tonhöhe, 329
TopDocs, 205
TopDocs, Realisierung des Arrays, 210
topics, 80, 81
TREC-Kollektion, 79, 143, 195
Trennschärfe, 48, 190
Trunkierung, 103
Trust, 127

UND-Verknüpfung, 143
Unicode, 126
Uniform Resource Identifier (URI), 126, 128
Uniform Resource Locator, URL, 367
Usability, 90

Vektorraummodell, 45, 96, 185, 281

Vektorraummodell, Basisalgorithmus, 204
Vektorraummodell, Implementierung des, 204
Vereinigung mehrerer Listen, 142
Vereinigungsmenge der Listen, 174
Verschlagwortung, 28
Verschlüsselung, 127
Visualisierung, 249
Vorkommenshäufigkeit, 171, 188
Vorkommensort, 171, 376
Vorzugsbenennung, 115

Ward's Methode, 239, 246
Water-Inflow, 315
Web Directory, 379
Web Ontology Language (OWL), 126
Weblog, 397
Windowing, 326
Wissen, 19
Wissensrepräsentation, 19
World Wide Web, WWW, 353
Wort, 93
Wortabstand, 111
Wortsuche, 139
Wörterbuch, 109

XML, 18

Zensur, 396
Zentroid Methode, 239, 246
Zipf's Law, 144
Zufallssystem, 68

Ähnlichkeitsmatrix, 234, 246
Ähnlichkeitsmaß, 96, 194
Ähnlichkeitsmaße, 233
Äquivalenzklasse, 116, 117

Überlagerung, 166

überlagerungsfähige Signaturen, 152