Lernfähiges Hypermedia-Information-Retrieval-System

Diplomarbeit
am Lehrstuhl für Kognitive Systeme
Institut für Informatik und praktische Mathematik
Christian-Albrechts-Universität zu Kiel
Prof. Dr. Gerald Sommer

ausgearbeitet von:
Klaus-Peter Herget
Betreuer: Dirk Kukulenz

vorgelegt im Mai 2000

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

Informationen sind in unserer durch elektronische Medien immer enger verbundenen Welt zu einem zentralen Faktor geworden. Da sie Grundlage jeglicher seriöser Entscheidungsfindung sind, ist ihre schnelle Verfügbarkeit, Korrektheit und Vollständigkeit unabdingbar. Durch immer leistungsfähigere Systeme und weltweite Vernetzung schwillt die verfügbare Informationsmenge zu einer Flut an, deren Fülle es immer schwieriger macht, relevante Informationen von unrelevanten zu trennen.

Zur Bewältigung dieser Informationsflut werden im praktischen, EDV-unterstützten Einsatz Informationssysteme unterschiedlicher Art eingesetzt. Man unterscheidet verschiedenartige Datenbanksysteme, Management-Informationssysteme, Expertensysteme und Information-Retrieval(IR)-Systeme. Grundsätzlich müssen diese Systeme die

- Aufnahme und interne Organisation von Daten, Fakten und Wissen und die

- Ausgabe möglichst relevanter Informationen und die dazugehörige Nutzerführung

gewährleisten.

In der Praxis sind formatierte Datenbanken, d.h. Datenbanken, deren interne Darstellung der Daten in einer vordefinierten Metastruktur wie z.B. Tupeln erfolgt, bereits weit verbreitet. Dem hier verwendeten Konzept der *relationalen Datenbanken* liegt eine ausgereifte mathematische Modellierung in Relationen-Algebra und verschiedenen Relationenkalkülen (TRC, DRC) [Ka93] zugrunde. Es können standardisierte Abfragesprachen wie SQL, Alpha oder QUEL eingesetzt werden. Ihre weite Verbreitung ermöglicht eine plattformübergreifende Anwendung.

Um Computer bei der Verwendung von formatierten Datenbanken einsetzen zu können, erfolgte die Entwicklung spezieller Softwarepakete, sogenannter DBMS (Database Management Systems) . Jedes DBMS basiert auf einem speziell für den jeweiligen Einsatzbereich erstellten *logischen Datenmodell.* Man versteht unter einem Datenmodell einen formaler Rahmen zur Beschreibung von Datenstrukturen und Operationen auf Daten [Ka93].

Grundsätzlich besteht ein Datenmodell aus einer Kombination von mindestens drei Komponenten:

1. einer Menge von *Datenstruktur-Typen*;

2. einer Menge von *Operatoren* oder *Inferenzregeln*, die auf alle gültigen Datentypen aus 1) angewendet werden können, und

3. einer Menge von *Integritätsbedingungen*, die implizit oder explizit eine Menge von konsistenten Datenbankzuständen definieren oder verändern.

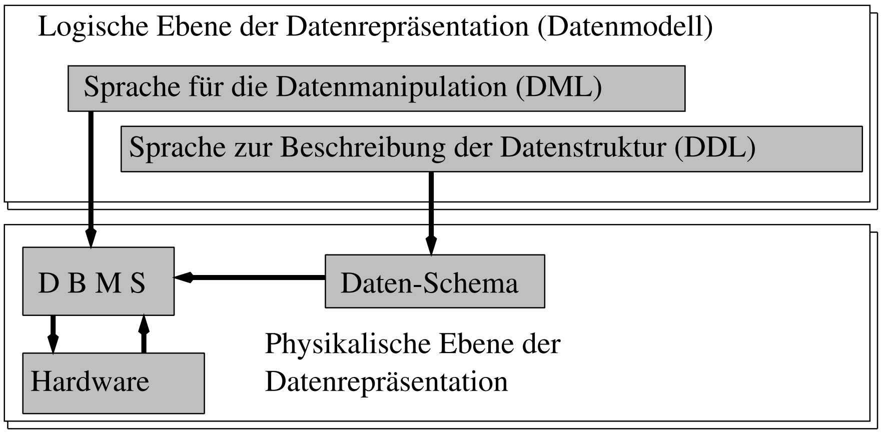

Abbildung 1.1: Logisches Datenmodell.

Die Strukturregeln sind typischerweise in einer *DDL* (Data Description Language) festgelegt, die Operationen in einer *DML* (Data Manipulation Language) definiert (siehe Abbildung 1.1).

Im Gegensatz zu den oben beschriebenen Datenbank-Systemen verwaltet das hier realisierte Informationssystem semi-strukturierte Daten, d.h., die Darstellung dieser Daten unterliegt nur einer sehr generellen Struktur, wie beispielsweise Sätze oder Abschnitte in Texten.

Die Daten und Informationen stammen aus einem in den letzten Jahren schnell anwachsenden Bereich, dem Hypertext- und Multimedia-Umfeld (zusammen *Hypermedia*). Hier ist es nicht möglich, Strukturen vorzugeben, weil jeder Autor eines

Hypermedia-Dokuments seine eigene interne Gliederung entwirft. Die Informationsgewinnung wird somit erheblich erschwert. Ebenso ist es problematisch, die Relevanz der nach einer Anfrage vom System erhaltenen Daten zu beurteilen.

1.2 Ziel und Aufbau der Arbeit

Das Ziel dieser Arbeit besteht in der Entwicklung eines Informationssystems für eine Hypermedia-Umgebung. Das System soll möglichst flexibel für verschiedene Themenbereiche und unterschiedliche Landessprachen einsetzbar sein. Damit auch ungeübte Nutzer leichten Zugang haben, soll eine möglichst einfache Schnittstelle zum System angeboten werden, die wenige zur Bedienung benötigte Kenntnisse voraussetzt.

Um den Nutzer eines für diesen Bereich konzipierten Informationssystems davon zu entlasten, viel eigenes Know-How mitbringen und langwierige Suchen in einem weitläufigen Informationsfeld durchführen zu müssen, soll dieses System beurteilen können, wie groß die Relevanz gefundener Antworten für eine Anfrage ist. Es ist dann in der Lage, die Antworten in der Reihenfolge ihrer Wichtigkeit auszugeben. Der Nutzer wäre somit nicht gezwungen, alle Ergebnisse selbst anzusehen und zu begutachten.

Im Hypermedia-Umfeld herrscht eine starke Datenfluktuation, weil jeder Anbieter von Informationen uneingeschränkten Zugriff auf zumindest seinen eigenen Bereich hat: Er kann beliebig Daten hinzufügen oder entfernen. Um großen personellen Aufwand und die damit verbundenen Kosten zu vermeiden, muss ein Informationssystem für diesen Bereich weitgehend automatisiert sein.

Die Nutzer-Struktur eines Hypermedia-Informationssystems ist im allgemeinen ebenso vielfältig, wie die durch das System verwalteten Daten. Es ist daher erforderlich, verschiedensten Ansprüchen in der Nutzerführung gerecht zu werden und Informationsbedarf auf unterschiedlichen Ebenen zu decken.

Allein mit den aus den strukturierten Datenbanken bekannten Verfahren und Methoden zur Informationsverwaltung- und Wiedergewinnung lassen sich solche Anforderungen nicht erfüllen. Im hier vorgestellten System kommen daher andere Methoden zum Einsatz. Sie stammen einerseits aus dem Bereich des Information Retrieval und andererseits aus dem Gebiet der künstlichen Intelligenz.

Durch assoziatives Lernen stellt das System Beziehungen zwischen allen im gegebenen Umfeld verwendeten Begriffen her und ermöglicht damit dem Nutzer eine erweiterte Suche auf Basis *synonymer* Zusammenhänge zwischen diesen Begriffen. Das Lernen erfolgt in zwei voneinander unabhängigen Bereichen:

- in Inhalt und Struktur der einzelnen Dokumente und

- in der Struktur der Hypertext-Verweise zwischen den Dokumenten.

Der Umfang des verwaltbaren Bereichs hängt von der Hardwarekapazität des Systems ab und könnte beispielsweise den Web-Server eines großen Unternehmens, einer Universität oder eine Online-Bibliothek u.s.w. umfassen.

Im Bereich der Nutzerführung bietet das System zu einer ursprünglichen Anfrage sinnverwandte Begriffe an, die, falls erforderlich, eine Spezifizierung dieser Anfrage ermöglichen. Der Nutzer erhält also über das Prinzip "Wiedererkennen", welches der assoziativen Funktionsweise des menschlichen Gehirns entgegenkommt, einen schnelleren Zugriff auf die von ihm benötigten Informationen.

Das nun folgende Kapitel beschreibt grundlegende Konzepte und Methoden von Information Retrieval. Daran anschließend erfolgt eine Einführung in den Hypertext- und World-Wide-Web-Bereich, um die damit in Zusammenhang stehenden Fragen bei Aufbau und Design eines Hypermedia Information Retrieval (HIR) Systems zu klären.

Nach der Beschreibung des Systems in Kapitel 4 und 5 erfolgt in Kapitel 6 eine Darstellung verschiedener Auswertungen und Messungen zur Abschätzung seiner Leistungsfähigkeit.

Anschließend wird in Kapitel 7 eine Zusammenfassung und Bewertung der Ergebnisse mit einem Ausblick auf zukünftige Erweiterungs- und Verbesserungsmöglichkeiten diskutiert.

Das letzte Kapitel beschäftigt sich der Implementation des Systems. Der interessierte Leser kann hier die Lösung einiger technischer Probleme bei der Programmierung nachvollziehen.

Kapitel 2

Information Retrieval

Information Retrieval (IR) ist eine seit etwa drei Jahrzehnten etablierte Technik zur Informationsgewinnung aus Texten. IR wurde beispielsweise zur Suche nach Patenten, in Gesetzes-Datenbanken oder für Kataloge von Bibliotheken eingesetzt. Um eine Suche zu ermöglichen, war speziell geschultes Personal erforderlich, dessen Aufgabe darin bestand, eine Schnittstelle zum System zu bilden und den Nutzer interaktiv bei der Informationsgewinnung zu unterstützen.

Durch fortgesetzte Entwicklung von Netzwerk-Technik und explosionsartige Vermehrung verfügbarer Informationen entstand die Notwendigkeit, dem Nutzer zu ermöglichen, seinen Informations-Bedarf selbst zu decken, ohne den schwierigeren und wesentlich teureren Umweg über Experten gehen zu müssen, deren Hilfe bisher für das Auffinden dieser Informationen erforderlich war.

Ein hierbei auftretendes Problem besteht darin, dass die zu durchsuchenden Daten kaum strukturiert sind, weil sie meist in natürlichsprachlicher Form vorliegen. Eine weitere Schwierigkeit liegt in der Formulierung der Suchanfragen. Um den Nutzer möglichst weitgehend davon zu entlasten, sich mit den systeminternen Gegebenheiten und einer Anfragesprache vertraut machen zu müssen, ist es vorteilhaft, auch Suchanfragen in natürlichsprachlicher Form zu ermöglichen. Dies birgt allerdings die Gefahr, dass schon in der Anfrage Missverständnisse auftreten können, wenn der Nutzer nur eine vage oder ungenaue Vorstellung von den von ihm tatsächlich benötigten Informationen hat.

Die Aufgabe der heutigen IR ist es somit, ungenauen Informationsbedarf zu decken, der unpräzise in einer mehrdeutigen Sprache formuliert ist, und diesen mit Informationen vergleicht, die ebenso vage, in einer ebenso mehrdeutigen Sprache formuliert sind. Im Folgenden werden verschiedene Strategien und Methoden vorgestellt, die dieses Problem in Angriff nehmen.

2.1 Retrieval-Methoden

Die in den letzten 30 Jahren gebräuchlichste Form des Retrieval ist das Boolesche Retrieval. Hier formuliert der Nutzer eine aus Schlüsselwörtern oder Index-Termen bestehende Anfrage, die er bei Bedarf mit den aussagelogischen Begriffen **AND, OR** oder **NOT** verbinden kann. In der Kollektion der Dokumente wird nun untersucht, ob ein Dokument den Text der vorgegebenen Anfrage enthält. Erweiterungen zu diesem Prinzip ermöglichen Teilstring- und Wildcardsuche in der Anfrage.

Obwohl dieses Verfahren leicht zu implementieren ist und schnelle Antwortzeiten garantiert, bestehen einige Nachteile:

1. Für einen unerfahrenen Nutzer ist es schwierig, eine effektive logische Anfrage zu formulieren, die seinen Informationsbedarf gezielt beschreibt und die Ausgabe einer geeigneten Menge relevanten Materials garantiert.

2. Es gibt keine Kontrollmöglichkeit über die Anzahl der nach einer Anfrage ausgegebenen Dokumente. Damit besteht die Gefahr, den Nutzer mit einer allzu großen Menge an Information zu überlasten oder die Antwortzeit des Systems unakzeptabel zu verlängern.

3. Alle gefundenen Dokumente werden als gleich relevant behandelt und in einer beliebigen Reihenfolge ausgegeben. Der Nutzer ist so gezwungen, nochmals zu suchen, weil die für ihn wichtigen Dokumente zufällig im Anfrage-Ergebnis verteilt sind.

4. Unterschiedliche Teile der Anfrage können nicht verschieden gewichtet werden. Somit besteht keine Möglichkeit, Konzepte mit feineren Unterschieden in der Anfrage zu formulieren.

Eine alternative Retrieval-Methode misst die Ähnlichkeit jedes Dokuments mit der Anfrage und listet die Dokumente in der Rangfolge der ermittelten Ähnlichkeiten auf. Um Ähnlichkeiten zwischen Dokumenten und Anfragen abzuschätzen, kann man die einzelnen Index-Terme mit Gewichtungen versehen. Eine populäre und effektive Methode, automatisch Term-Gewichte zu generieren, ist die in Kapitel 4.3 beschriebene tf*idf-Strategie. Vorteile bieten sich hier durch die Entlastung des Nutzers von der Notwendigkeit, eine booleschen Anfrage zu erstellen und die Möglichkeit, die Menge der auszugebenden Dokumente zu kontrollieren.

Als eine weitere Strategie ist das Relevanz-Feedback-Verfahren [Sal90] zu nennen, das man zusätzlich zur Gewichtung von Termen und Ähnlichkeitsmessungen von Dokumenten einsetzt. In diesem Verfahren wird das Urteil des Nutzers über die

Relevanz (oder Nicht-Relevanz) eines Dokuments bezüglich seiner Anfrage an das IR-System zurückgegeben. Das System verwendet diese Information zu einer Neuberechnung der Termgewichte, die eine Veränderung in Reihenfolge der Wichtigkeit der Dokumente bewirkt. Die Grundidee bei dieser Vorgehensweise besteht in der Annahme, die Nutzeranfrage spezifiziere bei Beginn des Vorhabens den Informationsbedarf des Nutzers nur unvollständig. Nach der Präsentation einiger Dokumente könne der Nutzer sein Informationsziel deutlicher erkennen oder klarer formulieren.

Eine weitere Methode zur Verwendung von Relevanz-Informationen besteht darin, Nutzer-Anfragen automatisch oder manuell zu modifizieren [Ha92]. Man geht davon aus, dass theoretisch eine "optimale Suchanfrage" existiert, zu deren Formulierung allerdings die Mengen von relevanten und nicht relevanten Dokumenten bekannt sein müssen. Diese Mengen sind in der Praxis jedoch unbekannt [Pan87]. Daher wird bei Relevanz-Feedback-Verfahren zur Modifikation der Suchanfrage die ursprüngliche Eingabe des Nutzers verändert und präzisiert, um die optimale Suchanfrage zu approximieren.

Für die Modifikation der Suchanfrage sind zwei Ziele zu nennen. Die Suchanfrage soll Dokumenten

- die als relevant eingestuft wurden, ähnlicher gemacht werden;
- die als nicht relevant eingestuft wurden, unähnlicher gemacht werden.

Eine modifizierte Suchanfrage unterscheidet sich von der ursprünglichen daher insofern, als sie eine hohe Ähnlichkeit mit den als relevant beurteilten Dokumenten aufweist, hingegen eine geringe mit den als irrelevant beurteilten.

Um die aufgezeigten Vorgaben zu erfüllen, wird im hier beschriebenen System die Anfrage durch Terme erweitert, die aus einer durch das System generierten Wissensbasis stammen.

Zunehmende Beachtung findet eine *konzeptionelle* oder *wissens-basierte* IR, die in Systemen wie SCISOR [Ja90] oder FERRET [Ml91] realisiert ist. Hier beantwortet man die Nutzeranfrage nicht mit einem Dokumenten-Reihenfolge, sondern durch eine mit Hilfe semantischer Textanalyse der Dokumente generierte Antwort. Systeme, die auf konzeptioneller IR basieren, operieren typischerweise nur in sehr begrenzten Bereichen. Auch hier sind zur Erstellung der Wissens-Datenbank eingehende Fachkenntnisse über diese Bereiche erforderlich.

Auf der Basis des probabilistischen Modells setzten Turtle und Croft Inferenz-Netzwerke ein [Tu91], [Tu92]. Diese basieren auf einem Dokumenten-Netzwerk, einem Anfragen-Netzwerk und einem Inferenz-Mechanismus, der für den Retrieval eingesetzt wird. Im Dokumenten-Netzwerk beispielsweise werden Knoten, die aus Bezeichnern, Index-Termen oder konzeptrepräsentierenden Knoten bestehen können, mit gewichteten Verweisen verbunden. Ähnlich ist die Darstellung des Anfragen-Netzwerks konzipiert. Das Retrieval-Verfahren zur Bewertung eines Dokuments führt einen Bayesschen Inferenzprozess auf den Netzwerken aus. Realisiert sind diese Strategien in beispielsweise als öffentliche Suchmaschinen in den Archiven der US-Legislative eingesetzten Systemen wie INQUERY [Cr95] oder THOMAS [Cr97].

Ein weiteres Beispiel für den Einsatz neuronaler Netze im Information Retrieval bilden Hopfield-Netze [Ho82], die es ermöglichen, Wissen und Informationen in Neuronen (Knoten) und gewichteten Synapsen (Links) zu speichern. Durch ein Konvergenzverfahren, dass das Netz in einen stabilen Zustand versetzt, können verschiedene Klassifizierungs- und globale Optimierungsvorgänge durchgeführt werden.

Alle Inferenzsysteme haben ein gemeinsames Problem: Die kombinatorische Explosion von möglichen Inferenzen. Da Systemkapazitäten begrenzt sind, kann man i.a. nicht alle möglichen Inferenz-Ressourcen auswerten. Als Folge davon muss der durch das System bearbeitete Bereich eingeschränkt werden, entweder durch Kontrolle der Inferenzen, oder durch Verkleinerung des Term-Vorrates.

Beim Einsatz neuronaler Netze in einem WWW-Umfeld ist ebenfalls zu beachten, dass die Antwortzeit des Systems in vom Nutzer akzeptierten Grenzen liegen muss. Weil die Anzahl der Iterationen zur Stabilisierung eines neuronalen Netzes vorher nicht bekannt ist, kann es hier zu Zeitproblemen kommen. Beispielsweise erhält H. Chen [Ch94] bei der Verwaltung einer Wissensbasis von 14.000 Termen und 80.000 Gewichten mit einem modifizierten Hopefield-Netz durchschnittliche Antwortzeiten von 24.5 Sekunden. Eine solche Verzögerung nimmt der Nutzer im WWW-Bereich i.A. nicht mehr hin.

Die Wissensbasis des hier entwickelten Volltext-Systems enthält ca. 100.000 Terme und 10.000.000 Gewichte. Die Antwortzeiten eines Systems dieses Umfangs mit neuronalem Netzwerk wären keinesfalls mehr akzeptabel. Weiterhin ist es nicht in jedem Einsatzfeld gewährleistet, das bei einer solchen Anzahl von Begriffen die Benutzungshäufigkeit des Systems durch Internet-Anfragen für ein effektives Training ausreicht.

Ein neuronales Netz erfordert weiterhin eine Vorgabe von Trainings-Zielen. Das hier entwickelte System soll jedoch nur die tatsächlich vorhandenen Informationen aufnehmen, und keine von außen gegebenen Wertungen einbeziehen.

Aufgrund der oben genannten Einschränkungen wurden im hier entwickelten System keine neuronalen Netze eingesetzt.

2.2 Komponenten eines IR-Standard-Systems

Ein IR-System besteht aus einer Menge von Dokumenten, einer Menge von Suchanfragen und einem Mechanismus, der aus der Menge der Dokumente die zu einer Suchanfrage relevanten Dokumente bestimmt [Sal87, S. 11].

Für ein IR-System notwendige Teile sind:

- Informationserschließung (Indexierung)
  Die Informationserschließung erfasst den Inhalt der Dokumente und beschreibt ihn durch Merkmale, die das Auffinden und Unterscheiden der Dokumente ermöglichen. Dieser auch Indexierung genannte Prozess generiert eine Menge von Index-Termen, die zur internen Darstellung von Dokumenten und Anfragen geeignet sind. Diese soll den Vergleich der gespeicherten Daten miteinander und den Aufbau einer Metrik ermöglichen.

- Abfragesprache
  In diesem Teil wird dem Nutzer eine Schnittstelle zur Verfügung gestellt, die es ihm ermöglicht, Anfragen für seine Informationssuche zu formulieren. Die Suchanfragen werden ausgewertet und analysiert, um die dafür relevanten Dokumente bestimmen zu können.

- Informationsausgabe
  Hier erfolgt die Darstellung der Antwort des Systems auf die Suchanfrage des Nutzers.

Einer der wichtigsten und am schwierigsten zu realisierenden Teile eines solchen Systems ist die Informations-Erschließung, das Indexieren von Dokumenten. Welche Methoden und Strategien man bisher dafür entwickelt hat, beschreibt der nächsten Abschnitt.

2.3 Indexierung

Der Vergleich oder auch approximative Vergleich von Zeichenketten [RY98] reicht für die Realisierung eines IR-Systems nicht aus. Daher transformiert man Nutzeranfragen und Dokumente üblicherweise in interne Darstellungen, die beim Retrieval miteinander verglichen werden können. Dieser sogenannte Indexierungs-Prozess generiert für die interne Darstellung eines Dokuments oder einer Anfrage eine Menge von Indexierungs-Termen, deren kombinierte semantische Bedeutung in bestimmter Weise den Inhalt eben dieses Dokuments oder dieser Anfrage repräsentiert.

Ein Hauptproblem der Indexierung liegt in der Auswahl geeigneter Index-Terme. Der ANSI-Standard für die Verwendung von Indexen [ANS92] legt in Abschnitt 3 die

Funktionen eines Index fest. Es heißt dort: "The function of an index is to provide users with an efficient and systematic means for locating documents, portions of documents, or descriptions of documents that may address information needs or requests. An index should therefore:

- Identify documents that treat particular topics or possess particular features.
- Discriminate between major and minor treatments of particular topics or manifestations of particular features.
- Provide access to topics or features by means of the terminology of users.
- Link terms representing equivalent concepts and indicate relationships among terms representing related concepts.
- Provide for the combination of terms to facilitate the identification of particular types or aspects of topics or features and to eliminate unwanted types or aspects.

Der Indexierungssprache fallen somit 3 Hauptaufgaben zu:

1. Repräsentation des Inhalts der einzelnen Dokumente,
2. möglichst effiziente Unterscheidung und Abgrenzung der einzelnen Dokumente voneinander und
3. Verbindung der Dokumente miteinander, die thematisch zusammengehören.

Grundsätzlich besteht die Möglichkeit, Texte manuell zu indexieren. Hier kann man nicht nur einzelne Wörter, sondern ganze Phrasen für die Indexierung zu verwenden. Die Ausdruckskraft von Phrasen ist sicher stärker als die der Einzelterme, solche Verfahren sind jedoch nur in kleinen Domains anwendbar. In einem größeren Umfeld ist dies nicht möglich, weil manuelle Methoden erhebliche Nachteile in der Zeit- und Kosten-Komponente mit sich bringen.

2.4 Automatische Indexierung

Die Methoden der automatischen Indexierung versuchen den Prozess der Informationserschließung, also die Zuordnung von Deskriptoren zu den einzelnen Dokumenten, vollständig zu automatisieren, so dass die Funktion eines Indexierers entbehrlich ist. Dabei gibt es zwei unterschiedliche Ansätze:

1. den dokumentorientierten Ansatz [Sal75, Sal87, S.65]. Hier werden die Deskriptoren direkt aus den Dokumenten gewonnen (Gewinnung von Stichwörtern).

2. den begriffsorientierten Ansatz [Lu89, S.1]. Bei dieser Methode existiert ein verbindliches Indexierungs-Vokabular, eine Art Wörterbuch. Aus dem Wortschatz dieses Wörterbuches werden den Dokumenten die Deskriptoren zugeordnet (Zuordnung von Schlagwörtern).

Beim dokumentorientierten Ansatz sind mehrere Möglichkeiten denkbar: (1) Die Deskriptoren können nur aus dem Titel des Dokuments, (2) aus dem Abstrakt (Zusammenfassung), oder (3) aus dem Volltext des Dokuments selbst gewonnen werden.

Beim begriffsorientierten Ansatz besteht das Problem der Erzeugung des verbindlichen Vokabulars. Weil der Aufbau und die Strukturierung dieses Wörterbuches nicht triviale Aufgaben sind, nimmt man für die Zwecke der Indexierung beim begriffsorientierten Ansatz zur Vereinfachung an, dass das Wörterbuch mit dem entsprechenden Vokabular bereits besteht [Lu89, S.1].

Die zur automatischen Indexierung im dokumentenorientierten Ansatz verwendeten Verfahren lassen sich in folgender Weise klassifizieren:

1. Statistische Verfahren:
   Die statistischen Verfahren unterstellen, dass die Bedeutung der einzelnen Begriffe in einem Dokument eng mit der Häufigkeit ihres Auftretens zusammenhängt. Wichtig sind hier die Arbeiten von Luhn [Luh57], von Maron und Kuhns [MK60], von Jones [SJ71] und von Salton [SY73,Sea76].

2. Probabilistische Verfahren:
   Bei den probabilistischen Verfahren wird die Wahrscheinlichkeitstheorie zur mathematischen Modellierung des Retrieval-Prozesses benutzt: Man wertet eine Anzahl verschiedener Phänomene aus, die Terme und Dokumente betrifft. Beispiele dafür sind Gemeinsamkeiten beim Auftreten von Termen, aus Wörterbüchern und Thesauri entnommene Term-Relationen oder Vorabwissen über statistische Term-Verteilungen in Dokumenten [Rij77]. Wichtige Arbeiten stammen von Bookstein und Swanson [BS74], von Harter [Har75a] und von Dillon und Federhart [DF82,DF84].

3. Linguistische Verfahren:
   Linguistische Verfahren versuchen, mit Hilfe einer morphologischen, syntaktischen und semantischen Analyse der Dokumente die Deskriptoren zu gewinnen. Wichtige Arbeiten sind die von Dillon und Gray [DG83], sowie das Projekt PHRASE [Ear72,Ear73].

Um einen möglichst hohen Automatisierungsgrad zu ermöglichen, verwendet das hier beschriebene System den dokumentenorientierten Ansatz mit statistischen Verfahren zur Indexierung. Probabilistische und linguistische Verfahren erfordern einen wesentlich höheren Aufwand für die Informationserschließung, insbesondere bei Verwendung mehrerer Landessprachen in der Dokumenten-Sammlung, weil hier erheblich mehr Vorwissen einzubringen ist.

2.5 Intelligente IR-Systeme

Ein wichtiger Schritt bei der Evolution Intelligenter IR-Systeme besteht in der Loslösung der Relation zwischen Anfrage und durch das System erhaltener Antwort. Ein Datenbank-System ermittelt alle gespeicherten Strukturen, die exakt zur Anfrage passen. Ein IIR sollte in der Lage sein für die Anfrage relevante Informationen zu finden und zu beurteilen. Nicht nur die Syntax der Anfrage sollte Ausgangspunkt für den Retrieval-Prozess sein, sondern auch deren Bedeutung.

In der Literatur existiert keine eindeutige Definition für intelligente IR-Systeme, Smith [Sm80], Knorz [Kn83] und Lustig [Lu85] beschreiben einige Merkmale folgendermaßen: Ein intelligentes IR-System liegt vor, wenn es eines oder mehrere der folgenden Eigenschaften besitzt:

- Das intelligente IR-System integriert verschiedenartige Formen der Dialogunterstützung. Ein Experte benötigt sicher andere Hilfestellungen, als ein ungeübter Nutzer. Aufgrund einer Analyse der Recherchenhäufigkeit könnte das System beispielsweise eine Nutzer-Klassifikation vornehmen und daraufhin unterschiedliche Nutzer-Schnittstellen anbieten.

- Das intelligente IR-System verfügt über ein speziell entwickeltes Expertensystem, welches die Funktion eines qualifizierten Informationsvermittlers ausüben und das Systemverhalten mit Hilfe einer Erklärungskomponente erläutern kann. Sie stellt beispielsweise dar, welche Recherche-Syntax vorliegt, welche Datenbank bei einer bestimmten Problemstellung anzusprechen ist, oder wie bei einer Suche vorzugehen ist.

- Die Repräsentation der Daten eines intelligenten IR-Systems ist an Techniken der Künstlichen Intelligenz, wie z.B. an Frames oder semantische Netze angelehnt. Diese Form der Darstellung kann den gesamten Datenbestand oder nur bestimmte Teile betreffen, wie beispielsweise den Thesaurus.

- Eine natürlichsprachliche Nutzer-Schnittstelle erleichtert den Zugang zum System wesentlich und befreit den Nutzer von den syntaktischen Zwängen einer formalen Retrieval-Sprache. Ist das System sehr weit entwickelt, so kann die gesamte Recherche natürlichsprachlich ablaufen.

- Ist ein IR-System in irgendeiner Form lernfähig, so gilt es als intelligent.

- Kann ein IR-System durch die Interpretation einer Retrieval-Sitzungstheorie verbesserte Konzepte zur weiteren Unterstützung bei Retrieval-Anfragen erkennen, ist es ebenfalls als intelligent einzustufen.

Das in dieser Arbeit entwickelte System ist als intelligent zu bezeichnen, weil die Datengewinnung für die Thesauri Techniken der KI verwendet. Es lernt durch Hinzugabe von Dokumenten zunehmend semantische Verbindungen zwischen den verwendeten Termen herzustellen, indem die Gewichtungen der Ähnlichkeiten zwischen den Termen den neuen Verhältnissen nach der Zugabe angepasst werden.

Weiterhin bietet das System dem Nutzer die Möglichkeit einer natürlichsprachlichen Anfrage. Er kann in der hier realisierten Installation seinen Informationsbedarf in zwei unterschiedlichen Landessprachen, deutsch und englisch, formulieren. Um die Anpassung an eine andere Sprache zu erreichen, muss lediglich die Stoppwortliste mit den entsprechenden Wörtern aus dieser Sprache erweitert werden.

Die Dialogunterstützung erfolgt durch Vorschläge seitens des Systems, die eine *semantische* Erweiterung der Anfrage erlauben und auf diese Weise eine tiefergehende Erschließung der vorhandenen Informationen ermöglichen.

Kapitel 3

Hypertext und Hypermedia

Ende der 80iger Jahre begann die Verbreitung der ersten Generation von Hypertext-Systemen [Ni90], [Co87], [Mau98]. Vom Anfang ihrer Verfügbarkeit an schenkte die Gemeinde der IR-Forscher diesen Systemen eine große Beachtung. Weil Hypertext-Systeme auch für den PC-Bereich vorhanden waren, vereinfachte sich die Bereitstellung von Software-Tools zur Entwicklung von Prototypen für verschiedenste Anwendungsbereiche erheblich. Noch wichtiger für die IR-Forschung war die nun entstandene Möglichkeit der Durchführung von Experimenten, die aufgrund der teuren und zeitaufwendigen Entwicklungs-Erfordernisse für ad-hoc Software bis dato nicht vorhanden war.

Man bezog einige dieser Prototypen in die Entwicklung von Information Retrieval ein: Ein Beispiel für einen solchen Prototyp ist das von Frisse [Fr88] entwickelte System für medizinische Informatik. Andere Systeme wie GUIDE oder HyperCard waren zunächst für den Apple Macintosh verfügbar und wurden später auf den IBM PC portiert.

Dieses Kapitel stellt nun die charakteristischen Merkmale von Hypertext und Hypermedia-Systemen vor und untersucht deren Nützlichkeit für das Management von Informationen im Information Retrieval. Am Anfang steht eine Einführung in die grundsätzlichen Konzepte und brauchbaren Eigenschaften von Hypertext und Hypermedia, das Hauptaugenmerk richtet sich jedoch auf den Kontext von IR-Applikationen.

3.1 Hypertext

Das menschliche Denken arbeitet nicht in streng linearer Weise, es tendiert dazu, Assoziationen zu bilden. Wenn wir an etwas denken, stellen wir in diesem Sinn eine gedankliche Verbindungen zu etwas Anderem her, das in irgendeiner Weise damit zu tun hat. Daher bewegen wir uns schnell von einem Thema zu einem anderen, welches damit in Relation steht. Genau diese Möglichkeit, die Modellierung nichtlinearer Assoziationen zwischen Informationsfragmenten, bietet das Hypertext-Paradigma.

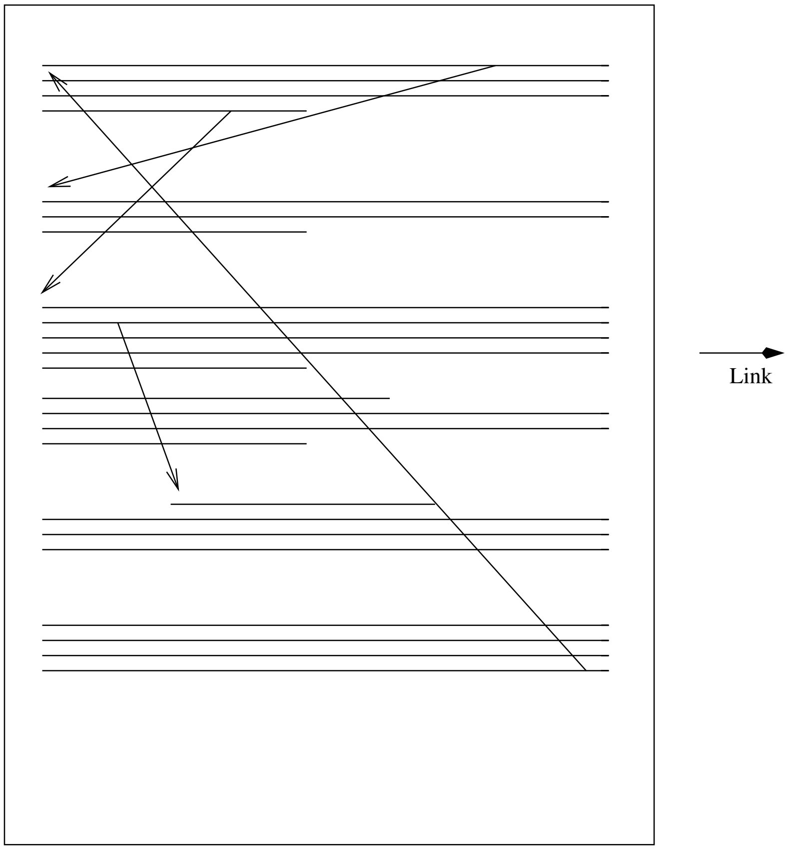

Abbildung 3.1: Netzwerk aus Knoten(Textteilen) und Verweisen

Der Hypertext-Ansatz zerlegt ein Dokument in Teile oder Fragmente; ein Netzwerk aus Knoten speichert und verwaltet alle diese Fragmente (siehe Abb. 3.1). Man kann Knoten, die in irgendeiner Weise inhaltlich ähnlich sind, durch Verbindungen miteinander verknüpfen. Diese Verbindungen nennt man *Informations-Links*, kurz *Links* oder Verweise. Das Netzwerk von Verbindungen ermöglicht nun, einem Pfad aus Verweisen durch die Knoten zu folgen und Informations-Fragmente anzusehen, die in Beziehung zueinander stehen. Dies ist wesentlich einfacher und schneller durchführbar, als eine Suche in einem von der Struktur her streng linearen Dokument, wie beispielsweise einem Buch.

Hypertext ist die Kombination von Fragmenten eines originalen flachen Dokuments und einer Menge von Verbindungen zwischen diesen Fragmenten. Man kann also definieren:

Hypertext = Knoten + Netzwerk aus Verweisen

Verweise sind gerichtet, d.h., es existiert immer ein Ausgangs- und ein Zieldokument. Von jedem Knoten kann eine Anzahl von Verweisen ausgehen, auf jeden Knoten kann eine Anzahl von Verweisen zeigen. Speziell soll erwähnt werden, dass dieses Knoten-Verweis-Paradigma die Speicherung aller Informationen über Verweise, die von einem Knoten ausgehen, in eben diesem Knoten voraussetzt. Damit ergeben sich folgende wichtige Konsequenzen:

1. Verweise sind unidirektional, d.h., das Netz kann nur in Vorwärtsrichtung durchlaufen werden.

2. Alle Verweise, die von einem Knoten ausgehen, gehen bei Löschung dieses Knotens verloren.

Die ursprüngliche Idee, mittels Hypertext einzelne Dokumente zu verwalten, wurde durch die Möglichkeit, Kollektionen von Dokumenten ins System aufzunehmen, erweitert. Ein Hypertext-System erlaubt dem Autor oder Gruppen von Autoren Dokumente miteinander zu verbinden, Pfade entlang miteinander in Beziehung stehender Dokumente zu erstellen und Referenzen zu anderen externen Dokumenten zu erzeugen. Man kann Knoten und Verweise erzeugen oder löschen. Der Inhalt eines Knotens oder eines Verweises kann verändert werden, somit können sich Struktur und Inhalt eines Hypertext-Systems dynamisch entwickeln.

3.2 Hypermedia

Das Hypertext-Konzept erlaubt jedoch nicht nur die Verarbeitung von Klartext. Es besteht die Möglichkeit, mittels der gleichen Technik Informationen in anderen Formaten, wie beispielsweise Graphiken, Bilder, Klang oder Video (*Multimedia*-Informationen) in Hypertext einzubinden.

Diese Weiterentwicklung, auch *Hypermedia* (**Hyper**text + Multi**media**) genannt, erlaubt dem Nutzer virtuell durch alle digital auf Computern speicherbaren Arten von Informationen zu navigieren. Da die meisten Hypertext-Systeme heutzutage in der Lage sind, verschiedene Formate zu verwalten, werden im Folgenden die Terme Hypertext und Hypermedia synonym verwendet.

Multimedia-Dokumente entstehen aus Komponenten wie Musik, Photographien, Hologrammen oder Filmen. Jede dieser Komponenten ist die sichtbare Manifestation eines bestimmten Typs von Daten, man kann generell sagen, dass Multimedia-Systeme mit Datenobjekten arbeiten. Die Basistypen Text, Bild, Ton und Video erfordern unterschiedliche Arten der Datenverarbeitung, der Speicherung, der Daten-Visualisierung und des Retrievals.

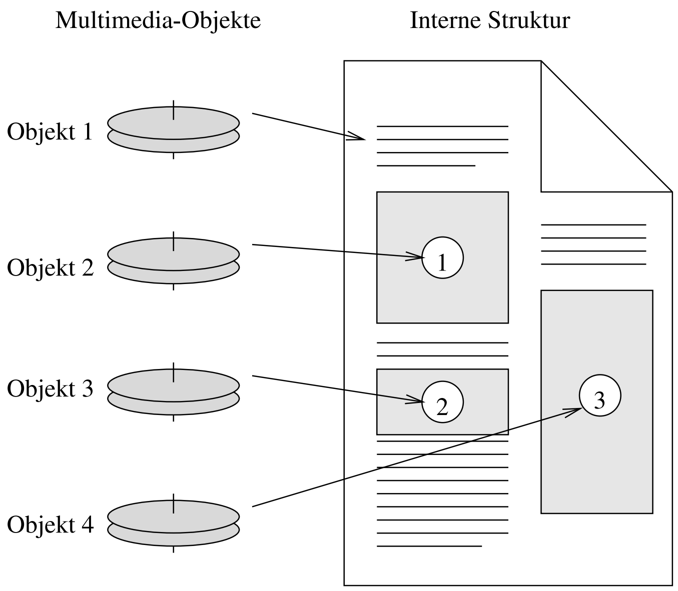

Abbildung 3.2: Interne Repräsentation eines Hypermedia-Dokuments

Zur Organisation von Multimedia-Daten in einem Hypermedia-Dokument existieren je nach Art dieser Daten verschiedene Konzepte. Die einfachste Möglichkeit besteht darin, einen Hintergrundtext mit Markierungen (*Tags*) zu versehen, die die Plazierungen der Multimedia-Daten im Dokument beschreiben (siehe Abb. 3.2).

Dieses Konzept ist jedoch nicht für alle Anwendungen ausreichend, weil das Einbinden dynamischer Multimedia-Daten wie Filmen, Klängen oder Animation auf Schwierigkeiten stößt, da keine zufriedenstellende Schnittstelle für eine Interaktion des Nutzers vorgesehen ist. Man verwendet in diesem Fall ein anderes Konzept, bei dem auf der Basis eines Skripts entsprechende Operationen ausgeführt werden. Das Anklicken eines Bilderrahmens könnte beispielsweise einen Video-Clip starten (Abb. 3.3). Ein Skript besteht aus einer Abfolge von Operationen, die von einem Hypermedia-System in ähnlicher Weise interpretiert werden, wie ein normales Programm von einem Computer.

Neuere Entwicklungen berücksichtigen die in beiden erstgenannten Konzepten nicht enthaltene Zeit-Komponente. Hier wird die Darstellung der Multimedia-Daten an einer Zeitachse orientiert, indem zeitlicher Startpunkt und Dauer der Visualisierung der Daten festgelegt werden (Abb. 3.4).

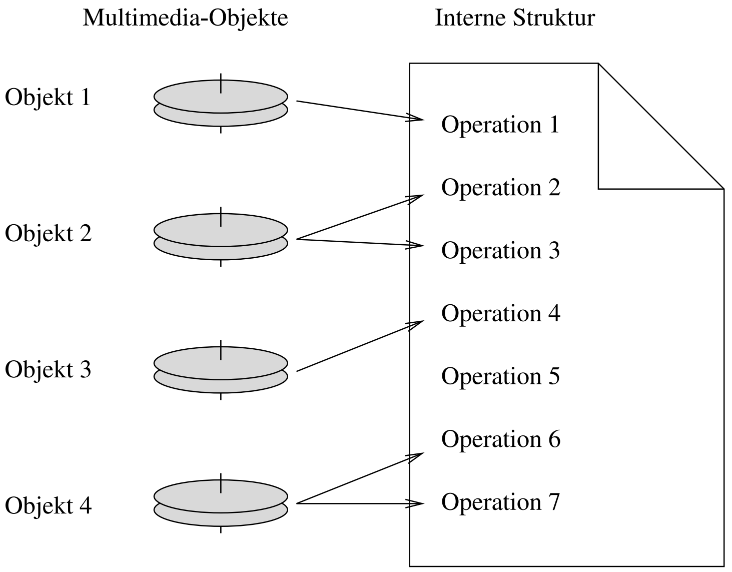

Abbildung 3.3: Hypermedia-Dokument als Skript.

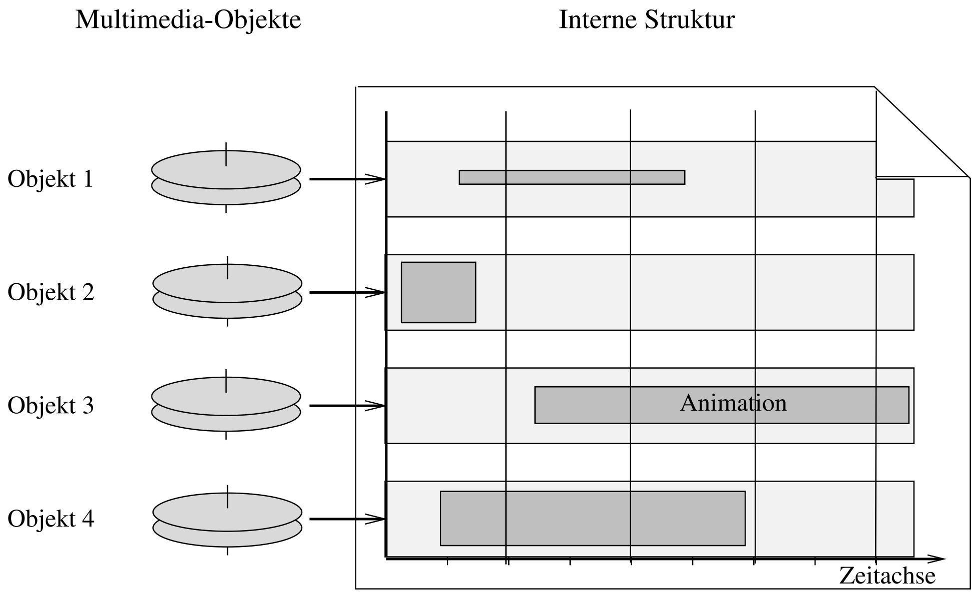

Abbildung 3.4: Hypermedia-Dokument mit zeitlicher Steuerung

3.3 Hypermedia Information Retrieval (HIR)

Die Anzahl der in einem üblichen IR-System verwalteten Dokumente ist normalerweise sehr groß und kann sich auf einige Tausend bis zu vielen Millionen Seiten belaufen. Die Kollektion läßt sich als eine Menge von Dokumenten auf einer flachen Ebene vorstellen. Jedes Dokument der Sammlung ist aus linearen Text-Abschnitten zusammengesetzt. Bei einer Nutzeranfrage werden als Folge dieser Organisation alle Dokumente als gleich relevant für diese Anfrage eingestuft.

Die Dokumente einer Hypertext-Sammlung sind nicht als auf einer Ebene liegend vorstellbar. Der Nutzer kann persönlich die Verweise zwischen den Dokumenten visualisieren und die Anzahl von Dokumenten sehen, die einen Pfad oder den Abstand zwischen zwei Dokumenten bildet. Zusätzlich ist es möglich, Verbindungen zwischen einem Dokument und anderen Dokumenten zu erstellen, auf die sich dieses Dokument bezieht. Das in einem Hypertext-System Dokumente oder Dokumentteile verbindende Netz der Verweise kann also analog wie die Dokumente in einem normalen IR-System zur Extraktion von Informationen verwendet werden. Dies hat zur Folge, dass man in der Hypertext-Architektur zur Beantwortung einer Anfrage zwei in der Semantik unterschiedliche Bestandteile der Dokument-Sammlung zur Informations-Extraktion heranziehen kann:

- die in Knoten organisierten Dokumente selbst und

- das Netzwerk der Verweise, das sie miteinander verbindet.

Der Nutzer eines solchen Systems kann seine Informationssuche auf der Basis zweier völlig unterschiedlicher Retrieval-Techniken durchführen und somit auf ein großes Reservoir an Wissen zum Lesen, Durchblättern oder Navigieren zugreifen.

3.4 HIR in einer Netzwerk-Umgebung

Die heutigen HIR-Systeme lassen sich in zwei Gruppen einteilen:

1. einzelne Hypermedia-Systeme, die Zugriff auf ein auf einem einzelnen Computer installiertes Hyperweb erlauben und

2. große Mehrbenutzer-Systeme, die vielfachen Zugriff auf ein Hyperweb gestatten, das auf viele in einem Netzwerk verbundene Computer verteilt ist.

Man verwendet ein verteiltes System typischerweise, wenn entweder die abzuspeichernde Menge von Informationen zu groß für die Installation auf einem einzelnen Computer ist, oder die Anzahl der Nutzer so hoch ist, dass diese nicht auf einem einzigen Computer bedient werden können.

Der Hauptunterschied zwischen HIR-Systemen im Netzwerk und Einzel-Systemen liegt im Bereich der Informationserschließung. In einem stationären System existiert meist ein Administrator, der das System überwacht und wartet. Er führt die Eingabe neuer Daten und damit verbundene Updates und Löschung überflüssiger Daten großteils manuell durch. Es besteht ein genauer Überblick über die im System enthaltenen Dokumente.

In einer Netzwerk-Umgebung kann jedoch nicht von jedem einzelnen Informations-Anbieter im Netz erwartet werden, dass er eine kontrollierende Instanz über jede Neuerung, Veränderung oder Löschung der Inhalte seiner Dokumente informiert. Der zu verwaltende Bestand verändert sich daher ständig und ist somit manuell nur sehr schwer auf einem aktuellen Stand zu halten.

Weil das hier entwickelte System im World Wide Web eingesetzt werden soll, erfolgt im nächsten Kapitel eine kurze Einführung in dieses Themengebiet.

3.5 Internet und World Wide Web (WWW)

Das Internet ist das größte weltweit existierende Computernetz. Es besteht aus einem Netz von Netzen, das nach Schätzungen inzwischen einige Millionen Computer mit über 250 Millionen Nutzern verbindet; Das Wachstum ist rapide, die Größe verdoppelt sich jährlich. Eine Grund-Charakteristik des Internets besteht im Zusammenschluss vieler verschiedenartiger Computer-Architekturen und Operationssysteme von Dutzenden von Soft- und Hardware-Herstellern. Eine Kommunikation zwischen diesen vielen verschiedenen Plattformen ist nur möglich, wenn eine Übereinstimmung in der Weise des Austausches von Daten erzielt wird.

Das WWW(*World Wide Web*)-Projekt[BL94] entstand durch eine großangelegte Hypertext-Information-Retrieval-Initiative, die sich zum Ziel gesetzt hatte, Nutzern den Zugriff auf eine sehr große Menge von Dokumenten im Internet zu ermöglichen. Die Entwicklung des WWW begann im Rahmen des WWW-Projekts bei der CERN, dem europäischen Labor für Teilchenphyisk. Es war ursprünglich beabsichtigt, ein verteiltes Hypermedia-System aufzubauen, doch die enormen über Ländergrenzen und Zeitzonen hinwegreichenden Kommunikations-Möglichkeiten sorgten für eine schnelle Verbreitung über diesen Rahmen hinaus.

Der Datenaustausch erfolgt zwischen WWW-Servern, die Kollektionen von Hypermedia-Dokumenten verwalten und sie öffentlich zugänglich machen auf der einen Seite, und WWW-Clients, die durch den Einsatz einer Hypermedia-Software, genannt *Browser*, Zugriff auf alle WWW-Server und deren Dokumenten-Sammlungen bekommen auf der anderen Seite. Durch die Verwendung solcher Browser, beispielsweise NETSCAPE, MICROSOFT EXPLORER oder MOSAIC, wird die Navigation in einem Hypermedia-Dokument sehr einfach, weil Browser eine "Point-and-click"-Schnittstelle bereitstellen, die alle für das Suchen im WWW verfügbaren Funktionen unterstützt.

Im Bemühen, generelle Ziele zu formulieren und gemeinsame Standards zu etablieren, baut das WWW-Projekt auf folgenden bereits im Internet verwendeten Standards [BL94], [BL95] auf:

- *TCP/IP, Transmission Control Protocol/Internet Protocol*: Dieses Protokoll organisiert des Zusammenschluss individueller, von verschiedenen Herstellern aufgebauter Netze (*LANs, Local Area Networks*) zu einem großen Netz aus kleineren Netzen. Wichtige Aufgaben sind hierbei der Aufbau sicherer Verbindungen, die Vergabe eindeutiger Adressen (*URL, Uniform Resource Locator*) für Internet-Ressourcen und das Erstellen von Leitwegen (*Routing*) für die auszutauschenden Daten.

- *HTTP, HyperText Transfer Protocol*: Das HTTP-Protokoll unterstützt die Kommunikation mittels Hypertext-Dokumenten im Internet. Typischerweise definiert dieses Protokoll die interne Struktur von Anfragen und Antworten zwischen Servern und Clients. Der Datenaustausch erfolgt auf der unter der HTTP-Protokollebene liegenden TCP/IP-Ebene.

- *HTML, HyperText Markup Language*: HTML als eine Anwendung des SGML-Standards ISO 8879 [SG86] erlaubt die Transformation von flachem Text in Hypertext. HTML definiert die Strukturierung und Darstellung von Informationen in einem Dokument, ermöglicht die Einbeziehung von nicht-textuellen Daten und gestattet die Erstellung von Links oder Verweisen zu anderen Dokumenten oder Dokument-Teilen. HTML ist der de-fakto-Standard der WWW-Sprache und unterliegt ständigen Erweiterungen und Verbesserungen.

Weil die von dem hier beschriebenen System verwalteten Dokumente in HTML verfasst sind, erfolgt nun eine kurze Einführung in diesen Sprach-Standard, um einen Überblick über die Leistungsfähigkeit und Grenzen dieser Sprache zu vermitteln.

3.6 Hypertext Markup Language (HTML)

HTML wird mit Hilfe von SGML (Standard Generalized Markup Language) definiert. SGML ist als ISO-Norm 8879 festgeschrieben.

3.6.1 Auszeichnung von Elementen

HTML ist eine sogenannte Auszeichnungssprache (Markup Language). Sie hat die Aufgabe, die logischen Bestandteile eines Dokuments zu beschreiben. Als Auszeichnungssprache enthält HTML daher Befehle zum Markieren typischer Elemente eines Dokuments, wie Überschriften, Textabsätze, Listen, Tabellen oder Grafik-Referenzen.

Das Beschreibungsschema von HTML geht von einer hierarchischen Gliederung aus und beschreibt Dokumente. Diese haben globale Eigenschaften wie zum Beispiel einen Titel oder eine Hintergrundfarbe. Der eigentliche Inhalt besteht aus Elementen, zum Beispiel einer Überschrift 1. Ordnung. Einige dieser Elemente haben wiederum Unterelemente. So enthält ein Textabsatz etwa eine als fett markierte Textstelle, eine Aufzählungsliste, bestehend aus einzelnen Listenpunkten, oder eine Tabelle, die sich in einzelne Tabellenzellen gliedert.

Die meisten dieser Elemente haben einen fest definierbaren Erstreckungsraum. So erstreckt sich eine Überschrift vom ersten bis zum letzten Zeichen, eine Aufzählungsliste vom ersten bis zum letzten Listenpunkt, oder eine Tabelle von der ersten bis zur letzten Zelle. Auszeichnungen markieren Anfang und Ende von Elementen. Um etwa eine Überschrift auszuzeichnen, lautet das Schema:

[Überschrift] *Text der Überschrift* [Ende Überschrift]

WWW-Browser, die HTML-Dateien am Bildschirm anzeigen, lösen die Auszeichnungsbefehle auf und stellen die Elemente dann in optisch erkennbarer Form am Bildschirm dar. Die Bildschirmdarstellung ist jedoch nicht die einzige denkbare Ausgabeform, HTML kann beispielsweise ebenso mit Hilfe synthetisch erzeugter Stimmen auf Audio-Systemen ausgegeben werden.

3.6.2 Vernetzung der Informationsfragmente

Eine der wichtigsten Eigenschaften von HTML ermöglicht, Verweise (auch *Hyperlinks, Links*) zu definieren. Sie können zu anderen Stellen im eigenen Projekt führen, aber auch zu beliebigen anderen Adressen im World Wide Web und sogar zu Internet-Adressen, die nicht Teil des WWW sind (E-Mail, FTP).

Durch diese einfache Grundeigenschaft eröffnet HTML völlig neue Welten. Das Bewegen zwischen räumlich weit entfernten Rechnern wird bei modernen grafischen WWW-Browsern auf einen Mausklick reduziert. In HTML-Dateien kann man Verweise notieren und dadurch inhaltliche Verknüpfungen zwischen eigenen Inhalten und denen anderer Anbieter herstellen. Ein wesentlicher Teil des hier beschriebenen Systems extrahiert die diesem so entstehenden Netz inhärente Semantik und stellt sie dem Nutzer zur Verfügung.

3.6.3 CGI: Common Gateway Interface

Das Common Gateway Interface (Allgemeine Vermittlungsrechner-Schnittstelle) stellt Programme im WWW bereit, die von HTML-Dateien aus aufgerufen werden, und selbst HTML-Code erzeugen und an einen WWW-Browser senden können.

CGI-Programme liegen auf einem Server-Rechner im Internet und verarbeiten bei Aufruf durch einen entfernten Nutzer bestimmte Daten. Die Datenverarbeitung erfolgt auf dem Server-Rechner. CGI-Programme können auf dem Server-Rechner Daten speichern, zum Beispiel, wie oft auf eine WWW-Seite zugegriffen wurde, oder, was ein Anwender in ein Gästebuch geschrieben hat. Bei entsprechendem Aufruf liest ein CGI-Programm gespeicherte Daten aus und generiert daraus HTML-Code. Dieser "dynamisch" erzeugte HTML-Code wird an den aufrufenden WWW-Browser eines Anwenders übertragen und zeigt dort individuelle Daten in HTML-Form an, wie den aktuellen Zugriffszählerstand einer WWW-Seite oder die bisherigen Einträge in einem Gästebuch.

Abbildung 3.5 zeigt eine typische CGI-Situation für Suchdienste im WWW. Der

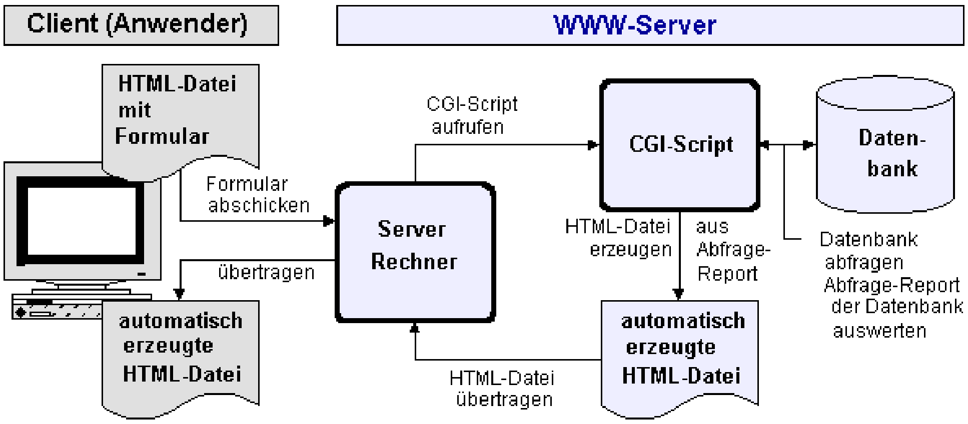

Abbildung 3.5: CGI-Situation für Suchdienste im WWW (aus [MN98])

Anwender kann in eine auf seinem Bildschirm dargestellte HTML-Datei Daten in ein Formular eingeben, etwa die Suche in einer Datenbank formulieren. Nach dem Abschicken des Formulars an den Server-Rechner wird dort ein CGI-Programm aufgerufen, das die vom Anwender eingegebenen Daten in eine Datenbankabfrage umsetzt. Die Datenbank-Anwendung liefert die Suchergebnisse an das aufrufende CGI-Programm zurück oder schreibt sie in eine Datei, die das CGI-Programm dann auslesen kann. Das CGI-Programm erzeugt nun HTML-Code, wobei es die Suchergebnisse als Daten in den HTML-Code einbaut und sendet diesen an den

WWW-Browser, der die Suchabfrage gestartet hat. Am Bildschirm des Anwenders verschwindet die WWW-Seite mit dem Suchformular. Stattdessen erscheint eine neue, vom CGI-Programm dynamisch generierte Seite mit den Suchergebnissen.

Kapitel 4

Theorie der Systemkomponenten

Das nun folgende Kapitel beschreibt den theoretischen Hintergrund der verwendeten Methoden und einzelnen Komponenten.

4.1 Überblick

Das in dieser Arbeit entwickelte Programm besteht aus drei Teilen. Der erste lernt bei der Installation des Systems durch Hinzugabe von Dokumenten aus einem Hypertext-Bereich im WWW das dort enthaltene Wissen. Der Grad der Assoziation aller enthaltenen Begriffe zueinander wird in einer Matrix gespeichert. Sie hat die Funktion eines Ähnlichkeits-Thesaurus, der zusätzliches Wissen aus der gegebenen Umgebung extrahiert und dem Nutzer zur Verfügung stellt. Der Thesaurus besteht aus zwei Teilen, deren einer den Inhalt der Dokumente darstellt, während der andere die in den Link-Verbindungen zwischen den Dokumenten enthaltenen Informationen repräsentiert.

Der zweite System-Teil stellt ein Abfrage-Modul zur Verfügung, das die im ersten Teil gewonnenen Daten abruft und verarbeitet. Stellt der Nutzer eine Anfrage an das System, analysiert es die Terme der Anfrage und erweitert diese durch dort bisher noch nicht enthaltene Terme. Diese stammen aus den Thesauri, die den Informations-Inhalt der abzufragenden Datensammlung und der dort enthaltenen Verbindungen widerspiegeln. Der Vorteil einer solchen Vorgehensweise ist, dass man auch bei ungenauen oder schlecht gestellten Anfragen durch eine semantische Unterstützung ein gutes Retrieval-Ergebnis erzielen kann.

Durch den Einsatz einer Term-Dokument-Matrix ordnet das Modul den Termen der Anfrage und deren Erweiterung relevante Dokumente zu. Der Nutzer bekommt das Ergebnis seiner Anfrage in Form einer nach Wichtigkeit geordneten Reihe von Dokumenten präsentiert.

Der dritte Teil schließlich verwaltet die Dokumenten-Sammlung und überwacht Eingaben oder Entnahmen von Dokumenten. Hier werden die zur Erhaltung eines aktuellen Systemstands erforderlichen Update-Operationen durchgeführt. Die Konstruktion und Aktualisierung aller Komponenten erfolgt automatisch, um die Notwendigkeit einer manuellen, also teuren und zeitaufwendigen Wartung des Systems zu vermeiden.

4.2 Generierung des Term-Vektorraums (TVR)

Um Dokumente und Anfragen in eine systeminterne Darstellung übertragen zu können, wird zunächst ein statistisches Verfahren zur Indexierung verwendet.

Als einer der Pioniere der automatischen Indexierung gilt H.P.Luhn, dessen Modell auf der Ermittlung der Worthäufigkeiten basiert [Luh57]. Dieses Modell besteht im Wesentlichen aus drei Schritten:

1. Für jedes Dokument d_k wird die absolute Häufigkeit h_i jedes einzelnen Begriffs t_i ermittelt.

2. Man errechnet, basierend auf den Ergebnissen aus Schritt 1, die Häufigkeit H_i eines Begriffs t_i in der gesammten Dokumenten-Sammlung.

3. Schließlich werden alle Begriffe, die öfter als ein oberer Schwellenwert H_{max} bzw., die seltener als ein unterer Schwellenwert H_{min} vorkommen als mögliche Deskriptorkanditaten eliminiert. Die verbleibenden Begriffe gelten dann als Deskriptoren. Die Schwellenwerte H_{max} und H_{min} sind in geeigneter Weise festzulegen.

Formal ergibt sich mit N als der Anzahl der Dokumente:

$$H_i = \sum_{k|_{t_i \in d_k}} h_i(d_k) \text{ und } k \in \{0, \cdots, N\} \tag{4.1}$$

Es lässt sich nun eine Bewertungsfunktion $G(i)$ erstellen, die angibt, ob ein Term t_i als Deskriptor verwendet wird oder nicht:

$$G(i) = \begin{cases} 1 & \text{wenn } H_{min} < H_i < H_{max} \\ 0 & \text{sonst} \end{cases} \tag{4.2}$$

Diese Vorgehensweise basiert auf zwei Annahmen: Begriffe, die in einem Datenbestand sehr oft vorkommen, eignen sich nicht als Deskriptoren, weil durch sie die Dokumente nicht hinreichend deutlich voneinander abgegrenzt werden können. Nur selten vorkommende Begriffe hingegen sind zu unbedeutend, um den Inhalt eines Dokuments repräsentieren.

Mit der sich ergebenden Index-Term-Menge kann nun ein hochdimensionaler Term-Vektorraum (TVR) aufgespannt werden. Der Inhalt eines Dokuments wird als Vektor des TVR dargestellt, indem man alle Einheitsvektoren der im Dokument auftretenden Index-Terme zu einem Dokument-Vektor aufaddiert.

Die Methode von Luhn ist einfach zu realisieren, denn die primäre Aufgabe eines Algorithmus, der das beschriebene Verfahren abbildet, ist das Zählen von Termen und das Sortieren nach Häufigkeiten. Das Verfahren von Luhn hat aber auch einige Nachteile [Sal87, S.67f.]:

- Die Eliminierung der Begriffe mit einer Häufigkeit größer als H_{max} kann Verluste beim Recall (Definition s. Kap. 6.3) verursachen, weil solche Hochfrequenzbegriffe den Nachweis zu vieler relevanter Dokumente zur Folge haben. Der Recall stellt das Maß für die Vollständigkeit des Retrieval-Ergebnisses dar. Er ist definiert als das Verhältnis zwischen den gefundenen relevanten Dokumenten und der Gesamtanzahl der im Dokumenten-Bestand vorhandenen relevanten Dokumente.

- Die Eliminierung der Begriffe mit einer Häufigkeit kleiner als H_{min} wiederum kann Verluste bei der Precision (Definition s. Kap.6.3) nach sich ziehen. Die Precision dient zum Messen der Genauigkeit der Suche und als Indikator für die Fähigkeit des Information Retrieval Systems, nicht relevante Dokumente auszuscheiden. Die Precision ist definiert als das Verhältnis der gefundenen relevanten Dokumente zur Zahl aller gefundenen Dokumente.

- Ein Term k, der in sehr vielen Dokumenten vorkommt, wird mit Sicherheit kein guter Deskriptor sein, auch wenn die Häufigkeit seines Auftretens zwischen den Grenzen H_{min} und H_{max} liegt, da er nicht hilft, die einzelnen Dokumente voneinander zu unterscheiden (vgl.Aufgaben einer Indexierungssprache in 2.3). Ein Term k jedoch, der in sehr wenigen Dokumenten relativ oft vorkommt, dessen Gesamthäufigkeit jedoch kleiner als H_{min} ist und damit nicht als Deskriptor ausgewählt wird, wäre ein geeigneter Term, um die Precision zu erhöhen.

Diese eben aufgezeigten Nachteile der Methode von Luhn verursachten weitere Überlegungen und Modell-Konstruktionen, die neben der Worthäufigkeit auch die Anzahl der Dokumente als weiteren Indikator einbeziehen.

4.3 Das tf*idf-Verfahren

Die Bezeichnung tf*idf ist eine Abkürzung für den Terminus "term frequency inverse document frequency". Der Begründer dieses Verfahrens, Sparck Jones, nimmt an, dass die Bedeutsamkeit eines Begriffes k proportional zur Häufigkeit seines Auftretens im Dokument i ist, aber umgekehrt proportional zur Gesamtanzahl der Dokumente, in denen er vorkommt [SJ72], [Sal87 , S. 68f.].
Es müssen somit zwei hauptsächliche Komponenten der Term-Gewichtung unterschieden werden:

1. die Häufigkeit tf_{ik} des Auftretens eines Terms T_k in einem Dokument D_i, auch als Term-Frequenz tf_{ik} von T_k in D_i bezeichnet;

2. die Anzahl idf_k der Dokumente D_i in welchen Term T_k auftritt. Üblicherweise wird idf_k als $log(N/n_k)$ berechnet, wobei N die Gesamtzahl der Dokumente und n_k die Anzahl der T_k enthaltenden Dokumente angibt.

Man kann als Gewichtungsfunktion angeben:

$$w_{ik} = tf_{ik} log(N/n_k) \tag{4.3}$$

Diese Gewichtung begünstigt Terme, die in der Dokumenten-Sammlung relativ häufig vorkommen, jedoch im einzelnen Dokument relativ selten auftreten. Die Unterscheidung und Abgrenzung der einzelnen Dokumente voneinander wird durch dieses Vorgehen genauer.

Um das Retrieval-Ergebnis weiter zu verbessern, kann die Dokumentenlänge, dargestellt durch die Anzahl der dem Dokument zugeordneten Index-Terme, in die Berechnung aufgenommen werden. Weil längere Dokumente eine größere Anzahl von Termen enthalten, erfolgt ihre Auswahl mit einer größeren Wahrscheinlichkeit, als die von als kürzeren Dokumenten. Zum Ausgleich dieser Wahrscheinlichkeiten kann man die Term-Gewichte w_{ik} normalisieren:

$$w_{ik} = \frac{tf_{ik} log(N/n_k)}{\sum_{k=1}^{t} (tf_{ik})^2 (log(N/n_k))^2} \tag{4.4}$$

Zur Messung der Effizienz des tf*idf-Verfahrens wurde die in der IR-Forschung meistverwendete TREC-Testkollektion [Ha94] herangezogen. Hier erwies sich das Verfahren als eine der leistungsfähigsten Methoden für die Gewichtung von Termen.

4.4 Reduktion der TVR-Dimension

Weil die Anzahl der zu verwaltenden Terme bei steigender Dokumentenzahl sehr groß werden kann, stellt sich die Frage, ob es tatsächlich notwendig ist, alle Terme in die Sammlung aufzunehmen. Es wäre sicher sinnvoll, die für die Darstellung des Inhalts eines Dokuments weniger relevanten Terme aus der Sammlung zu entfernen.

Es existieren diverse statistische Ansätze für die Auswahl nützlicher Terme aus einer Dokumenten-Sammlung. Die Poisson-Verteilung beispielsweise, eine diskrete Zufallsverteilung, wird basierend auf Arbeiten von Bookstein und Swanson [Bo74], Srinivasan [Sr90] und Margulis [Ma92] zur Auswahl von Index-Termen verwendet. Man geht dabei von der Annahme aus, dass sich die Verteilungen von relevanten und nicht relevanten Termen in einem Dokument voneinander unterscheiden. Die Verteilung der relevanten Terme entspricht hiernach nicht einer M-Poisson-Verteilung, wohl aber die Verteilung unrelevanter Terme. Die Terme, die entsprechend einer Poisson-Verteilung gestreut sind, betrachtet man somit als nicht genügend informativ zur Beschreibung des Dokuments, das sie enthält und entfernt sie aus der Index-Sammlung.

Ein weiterer Ansatz stammt von Salton, Yang und Yu [SY75]. Hier erfolgt die Einführung eines *Diskiminanzwerts* zur Beschreibung der Term-Relevanz. Dieser wird durch die Veränderung der Ähnlichkeit zwischen zwei Dokumenten bestimmt, die die Entfernung des besagten Terms verursachen würde. Das Diskriminanzwert-Modell verwendet die Annahme, dass die Steigerung der mittleren Ähnlichkeit zweier Dokumente das Retrieval-Ergebnis verbessert, wenn der Term nicht entfernt, sondern für die Indexierung verwendet wird. Somit ist ein Term mit hohem Diskriminanzwert erwünscht und erhält im Folgenden die Bezeichnung "guter Term".

Die Berechnung sowohl der Diskriminanzwerte als auch der Poisson-Verteilung erfordert einen hohen rechnerischen Aufwand. Glücklicherweise besteht eine starke Korrelation zwischen Poisson-Verteilungen, Term-Häufigkeiten in Dokumenten und Diskriminanzwerten [Sr90]. Speziell weisen empirische Untersuchungen eine starke Korrelation zwischen dem Diskriminanzwert eines Terms und der Häufigkeit seines Auftretens in Dokumenten nach. Häufig auftretende Terme sind demzufolge schlecht zur Unterscheidung einzelner Dokumente geeignet und werden aus der Term-Sammlung entfernt. Margulis [Ma92] erkannte, dass über 70% der häufig auftretenden Terme entsprechend einer M-Poisson-Verteilung verteilt sind. Wendet man diese Regel an, können je nach Art der Dokumenten-Sammlung 10 bis 20% der Terme entfernt werden, ohne das dies eine Verschlechterung des Retrieval-Ergebnisses verursacht.

Eine weitere Reduktion der TVR-Dimension läßt sich erzielten, indem man durch Vergleich aller Dokumente mit den Termen einer Stoppwortliste alle diejenigen Ter-

Rang	Begriff	Häufigkeit
1	the	69.971
2	of	36.411
3	and	28.852
4	to	26.149
5	a	23.237
6	in	21.347
7	that	10.595
8	is	10.099
9	was	9.816
10	he	9.543

Tabelle 4.1: Wort-Häufigkeiten (Textstichprobe 1.000.000 Wörter)

me als Deskriptoren entfernt, die sich in der Stoppwortliste befinden. Die Stoppwortliste enthält Bindewörter, Artikel und sonstige Füllwörter, die in einem Text sehr häufig vorkommen, jedoch nichts zur inhaltsbeschreibenden Erklärung des Dokuments beitragen.

Untersuchungen haben gezeigt, dass 20% der am häufigsten auftretenden Wörter ungefähr 70% der Texte abdecken. Tabelle 4.1 zeigt an einem Beispiel aus dem Englischen mit einer Textstichprobe von 1,000.000 Wörtern die 10 häufigsten Begriffe [Sal87, S.66]. Hier kann man erkennen, daß häufig Terme in einem Text auftreten, die für dessen Inhaltsbescheibung völlig ungeeignet sind und daher nicht als Deskriptoren verwendet werden können.

In vielen Systemen führt man zum Zweck einer weiteren Verkleinerung der Term-Menge eine Reduktion der Begriffe auf ihren Wortstamm durch [Sch86]. Dieses Verfahren kann die vorgegebene Textmenge um ca. 20% verringern.

Verschiedene Versuche [Ha91], [GCY95] haben jedoch ergeben, dass bei der Anwendung der in Kapitel 6.3 beschriebenen Messverfahren keine Verbesserung des Retrieval-Ergebnisses durch derartige Algorithmen erzielt werden kann. Die Untersuchung unter anderem, dass die Verwendung von Singular- oder Pluralform eines Nomen in keiner besonderen Beziehung zueinander stehen. Der englische Term "note" wird beispielsweise zu weniger als 25% als Nomen verwendet, "notes" hingegen zu mehr als 95%. Man könnte also anstatt die Menge der Index-Terme durch Wortstamm-Algorithmen zu reduzieren, auch eine entsprechende Anzahl zufällig gewählter Terme aus der Index-Liste entfernen. Dies hätte die gleiche Verschlechterung der Ausdruckskraft der Menge der Index-Terme zur Folge, wie der Einsatz eines

Wortstamm-Algorithmus.

Weiterhin können Probleme bei Anfragen entstehen, die die exakte Wortform erfordern. So würde beispielsweise eine Anfrage nach dem Namen "Billy" also nicht nur alle Träger des Namens "Bill", sondern auch alle Referenzen auf den Begriff "bill" (Rechnung) zurückliefern [Cr97]. Aus den genannten Gründen erfolgt im hier entwickelten System kein Einsatz von Wortstamm-Algorithmen.

4.5 Erstellung der Wissensdatenbanken

Aus der internen Darstellung der im System abgelegten Dokumente sollen nun Informationen gewonnen werden. Dies geschieht durch statistische Ermittlung von Term-Assoziationen unter Einsatz einer Abwandlung des beschriebenen tf*idf-Verfahrens. Die erhaltenen Informationen legt das System intern in Form von Assoziations-Thesauri ab.

Ein Thesaurus (griechisch: Schatz) ist eine Menge von Phrasen oder Wörtern und einer Relation über dieser Menge. Die bekannteste Art von Thesauri sind Wörterbücher, die Synonyme, Antonyme, Abstraktionen oder verwandte Terme von üblicherweise verwendeten Wörtern oder Ausdrücken enthalten. Sie werden manuell erzeugt (z.B. Roget, WordNet) und finden nur selten in IR-Systemen Verwendung. Neben diesen linguistischen Formen setzte man viele spezielle Thesauri für Dokumentation, Katalogisierung und Suche in Bibliotheken, Archiven und Datenbanken ein.

Thesauri spielen eine wichtige Rolle bei der Lösung von Problemen mit dem Vokabular, auf die ein Nutzer bei Informations-Suche trifft. Intelligente IR-Systeme wie z.B. CODER, UMLS oder I3S (Ein Überblick wird in [Ch93] gegeben) versuchen Expertenwissen in Thesaurus-Wissensbasen zu speichern und den Nutzer damit bei der Anfrage und Suche zu unterstützen.

Es gibt diverse algorithmische Methoden zur automatischen Konstruktion von Thesauri, die praktisch alle auf statistischen Bewertungen des gemeinsamen Auftretens von Termen beruhen. Im Gegensatz zu den verschiedenartigen Relationen zwischen Termen (z.B. Abstraktionen, Hierarchien, Definitionen u.s.w.), die manuell erstellte Thesauri bereitstellen, können diese Techniken nur unspezifische Ähnlichkeiten oder Assoziationen zwischen Termen beschreiben, die sich in einem bestimmten Umfeld befinden.

Dennoch haben Untersuchungen in wissenschaftlichen Umgebungen gezeigt, dass der Einsatz automatisch generierter Thesauri eine sehr brauchbare Unterstützung bietet [Ch95]. Einige von Grefenstette [Gr95] vorgestellte Ergebnisse zeigen sogar, dass automatisch und manuell erstellte Thesauri signifikante Ähnlichkeiten aufweisen.

Weil die manuelle Erstellung eines Thesaurus gleiche Probleme im Hinblick auf entstehende Kosten und unwägbare zeitliche Verzögerungen wie die manuelle Indexierung verursacht, kommen im hier implementierten System nur automatisch generierte Thesauri zum Einsatz, die eine für ein WWW-Umfeld notwendige Flexibilität bei geringem personellen Aufwand garantieren.

4.6 Erschließung des Dokument-Inhalts

Der konventionelle Ansatz zur automatischen Konstruktion eines Dokument-Term-Thesaurus berechnet eine Dokument-Term-Matrix A , deren Einträge w_{ik} sich aus den Werten der gewählten Gewichtungsfunktion ergeben und die Gewichte von Termen t_k in Dokumenten D_i angeben:

$$A = \begin{array}{c} \\ d_1 \\ d_2 \\ \vdots \\ d_m \end{array} \begin{array}{c} \begin{array}{cccc} t_1 & t_2 & \cdots & t_n \end{array} \\ \begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2m} \\ \vdots & \vdots & & \vdots \\ w_{m1} & w_{m2} & \cdots & w_{mn} \end{pmatrix} \end{array}$$

Um nun die Ähnlichkeit von Termen in einer Matrix C anzugeben, kann man

$$C = A^T A = \begin{array}{c} \\ t_1 \\ t_2 \\ \vdots \\ t_n \end{array} \begin{array}{c} \begin{array}{cccc} t_1 & t_2 & \cdots & t_n \end{array} \\ \begin{pmatrix} w'_{11} & w'_{12} & \cdots & w'_{1n} \\ w'_{21} & w'_{22} & \cdots & w'_{2n} \\ \vdots & \vdots & & \vdots \\ w'_{n1} & w'_{n2} & \cdots & w'_{nn} \end{pmatrix} \end{array}$$

berechnen [Sal83].

Dies bedeutet im Ergebnis, dass eine konventionelle Term-Term-Matrix auf der Basis eines gemeinsamen Auftretens von Termen in einem Dokument und einer Gewichtsfunktion erstellt wird. Die Gewichtsfunktion gibt an, wie gut ein Term ein Dokument repräsentiert, in dem er auftritt.

Der Ansatz kann modifiziert werden, indem man die Berechnung der Term-Term-Ähnlichkeit auf der Wahrscheinlichkeit aufbaut, wie gut ein Dokument die Bedeutung eines Terms repräsentiert [Qiu94]. Diese Methode vertauscht die Stellung von Term und Dokument des konventionellen Ansatzes, d.h. die Dokumente indexieren die Terme.

Der Vorteil dieses Verfahrens liegt in der Verwendung von nur tatsächlich in der Sammlung vorhandenem Wissen zur Indexierung. Bei jeder Indexierung besteht die Gefahr, dass Indexterme im normalen Sprachgebrauch auch andere, als die in der Dokumenten-Sammlung verwendeten Bedeutungen repräsentieren. In der Folge kann hierdurch eine ungenaue oder falsche Darstellung des Inhalts eines Dokuments auftreten. Im hier gewählten Ansatz ist der verwendete Index, nämlich das Dokument, mit seiner gesamten Bedeutung im System enthalten und repräsentiert nur Inhalte, die explizit im System enthalten sind.

Mit diesem Ansatz lässt sich eine Term-Dokument Matrix B berechnen mit

$$B = \begin{array}{c} \\ t_1 \\ t_2 \\ \vdots \\ t_m \end{array} \begin{array}{c} \begin{array}{cccc} d_1 & d_2 & \cdots & d_n \end{array} \\ \begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2m} \\ \vdots & \vdots & & \vdots \\ w_{m1} & w_{m2} & \cdots & w_{mn} \end{pmatrix} \end{array}$$

wobei B offensichtlich nicht gleich der transponierten Matrix A^T ist. Die Gewichte w_{ik} beschreiben in diesem Falle, wie ein Dokument d_k den Term t_i repräsentiert.

Weil die Stellung von Index-Term und Dokument vertauscht sind, bilden analog zum oben beschrieben Fall des TVR nun die Dokumente einen Vektorraum DVR. Die Darstellung der Terme erfolgt hier durch Addition der Dokument-Einheitsvektoren des DVR. Die Gewichtsfunktion ist deshalb entsprechend zu modifizieren. Der Bestimmung der Gewichte liegen folgende Annahmen zugrunde:

- Kürzere Dokumente sind wichtiger als solche größerer Länge, weil die Wahrscheinlichkeit, dass sich zwei Terme ähneln in einem kurzen Dokument größer ist als in einem langen.

- Je häufiger ein Dokument einen bestimmten Term indexiert, desto höher ist die Wahrscheinlichkeit, dass das Dokument die Bedeutung dieses Terms repräsentiert.

- Indexieren viele Dokumente einen bestimmten Term, kann das entweder bedeuten, dass dieser viele verschiedenen Bedeutungen hat, oder ein breit gestreutes Konzept beschreibt.

Mit diesen Annahmen kann man folgende Definition einer Gewichtsfunktion angeben:

$$w(t_i, d_k) := \frac{df(d_k, t_i) \cdot itf(d_k)}{\sqrt{\sum_{d_j \in t_i} (df(d_j, t_i) itf(d_j))^2}} \tag{4.5}$$

mit $df(d_k, t_i)$ als Anzahl der Dokumente d_k mit Term t_i. Der Wert $itf(d_k)$ wird als

$log(\frac{n}{|d_k|})$ berechnet, wobei n die Anzahl der Terme in der Kollektion und $|d_k|$ die Anzahl unterschiedlicher Terme im Dokument d_k angibt.

Nach der Bestimmung der Gewichtsfunktion kann man nun das Gewicht berechnen, das angibt, wie gut ein Dokument die Bedeutung eines Terms repräsentiert. Ein Term läßt sich als Vektor im durch die Dokumente aufgespannten Dokument-Vektorraum (DVR) darstellen

$$\vec{t_i} = (w(t_i, d_1), w(t_i, d_2), \cdots, w(t_i, d_m))$$

Die Ähnlichkeit zweier Terme kann man als einfaches skalares Vektorprodukt berechnen:

$$\begin{aligned} SIM_T(t_i, t_j) := \vec{t_i} \star \vec{t_j} &= \sum_{k=1}^{m} w(t_i, d_k) \cdot w(t_j, d_k) \\ &= \sum_{d_k \in t_i \cap t_j} w(t_i, d_k) \cdot w(t_j, d_k) \end{aligned} \tag{4.6}$$

Mit der beschriebenen Gewichtsfunktion ergibt sich wieder eine symmetrische Term-Term Matrix S_T:

$$S_T = BB^T = \begin{array}{c} \\ t_1 \\ t_2 \\ \vdots \\ t_n \end{array} \begin{array}{c} \begin{array}{cccc} t_1 & t_2 & \cdots & t_n \end{array} \\ \begin{pmatrix} d_{11} & d_{12} & \cdots & d_{1n} \\ d_{21} & d_{22} & \cdots & d_{2n} \\ \vdots & \vdots & & \vdots \\ d_{n1} & d_{n2} & \cdots & d_{nn} \end{pmatrix} \end{array}$$

mit $d_{ij} = SIM_T(t_i, t_j)$

Die Berechnung der Matrix erfolgt zur Vereinfachung späterer Updates in zwei Schritten: Zunächst in der Berechnung des dokumentenspezifischen Teils, dann in der Normierung. Für den unnormierten Teil der Gewichte gilt beim Term-Term-Thesaurus:

$$w'(t_i, d_k) := df(d_k, t_i) \cdot itf(d_k) \tag{4.7}$$

Bei der Erst-Installation des Systems wird in jedem Dokument k für alle dort auftretenden Term-Paare (t_i, t_j) das dokumentspezifische Term-Term-Ähnlichkeitsgewicht $t'_{ik} \cdot t'_{jk}$ berechnet und abgespeichert. Das globale Term-Term-Ähnlichkeitgewicht $sim_T(t_i, t_j)$ ergibt sich aus der Summe aller dokumentenspezifischen Term-Term-Ähnlichkeitgewichte:

$$sim_T(t_i, t_j) = \sum_{d_k | t_i, t_j \in d_k} w'_{ik} \cdot w'_{jk} \tag{4.8}$$

Gleichzeitig erfolgt für jeden Term die Bestimmung und Abspeicherung des dokumentspezifischen Normierungsfaktors t_{ik}^2. Der globale Normierungsfaktor $c(t_i)$ ergibt sich wieder aus der Summe aller dokumentenspezifischen Normierungsfaktoren:

$$c(t_i) = \sum_{d_k | t_i \in d_k} {w'_{ik}}^2 \tag{4.9}$$

Im zweiten Schritt werden die globalen Normierungsfaktoren eingesetzt und die endgültigen Einträge $SIM_T(t_i, t_j)$ der Term-Term Matrix S_T ergeben sich dann als

$$SIM_T(t_i, t_j) := \frac{sim(t_i, t_j)}{\sqrt{c(t_i) \cdot c(t_j)}} \tag{4.10}$$

Weil $sim(t_i, t_j) = sim(t_j, t_i)\ \forall i, j$ gilt, ist die Matrix symmetrisch zu ihrer Diagonalen und somit lediglich die Berechnung eines Dreiecks (oben oder unten) der Matrix erforderlich. Man kann zur Erst-Bestimmung der Matrix S_T nun folgenden Algorithmus verwenden:

Dokumentspezifische Term-Term-Gewichte und Normalisierungs-Koeffizienten:
for every document d_k **do**
  for every term t_i contained in document d_k **do**
    $c(t_i) := c(t_i) + {t'_{ik}}^2$
    for every term t_j contained in document d_k mit $i \leq j$ **do**
      $sim(t_i, t_j) = sim(t_i, t_j) + t'_{ik} \cdot t'_{jk}$
    end
  end
end
Normalisierung der Term-Term-Gewichte:
for every good term t_i of the collection **do**
  for every good term t_i of the collection **and** $i \leq j$ **and** $sim(t_i, t_j) > 0$ **do**
    $SIM_T(t_i, t_j) := \frac{sim(t_i, t_j)}{\sqrt{c(t_i) \cdot c(t_j)}}$
  end
end

Vorteile bietet dieses Verfahren durch

1. einen beschleunigten Zugriff zum Thesaurus, weil dieser relativ klein ist und
2. die Möglichkeit Update-Operationen durchzuführen, ohne eine Neuberechnung der gesamten Kollektion ausführen zu müssen.

4.7 Informations-Extraktion aus der Verweis-Struktur

Um weiteres Wissen aus der Dokumenten-Sammlung zu extrahieren, kann die Hypertext-Struktur der Dokumente ausgenutzt werden, speziell im hier beschriebenen System die vorhandenen Hypertext-Verweise.

Grundlegend machen wir nun die Annahme, dass auch solche *Dokumente Terme repräsentieren, die durch einen Hypertext-Verweis mit dem den Term enthaltenden Dokument verbunden sind.* Mit Hilfe dieser Annahme wird im Folgenden ein neues Verfahren vorgestellt, das in der Verweis-Struktur enthaltene Informationen extrahiert.

.

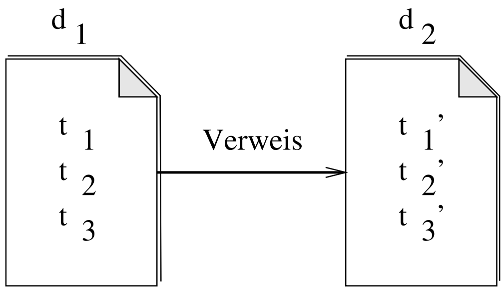

Abbildung 4.1: Terme in Dokumenten mit Verweisen

In Abbildung 4.1 beschreibt der Verweis $d_1 \to d_2$ einen inhaltlichen Zusammenhang der Dokumente d_1 und d_2. Analog zur Annahme der Term-Ähnlichkeit bei deren gemeinsamem Auftreten in einem Dokument, wird auch hier angenommen, dass wegen dieses Verweises das Dokument d_2 die Bedeutung der Terme t_1, t_2 und t_3 repräsentiert, ebenso wie Dokument d_1 die Bedeutung der Terme t'_1, t'_2 und t'_3.

Das hier verwendete Auswertungsschema für die Verweise ist binär, d.h., nur das Vorhandensein eines Verweises vom einen zum anderen Dokument geht in die Berechnung ein, ein mehrfaches Auftreten oder die Rückrichtung nicht. Einen Grund dafür liefert eine Analyse von Dokumenten-Sammlungen auf Hypertextbasis. Viele Verweise dienen nur der Verwaltung von Inhaltsangaben oder treten mehrfach zwischen zwei Dokumenten auf. Die Implementation des hier beschriebenen Systems basiert auf der Annahme, dass der Grad der Ähnlichkeit zwischen zwei Dokumenten nicht analog zur Anzahl der Verweise zwischen diesen ist. Einen Hinweis auf die Korrektheit dieser Annahme geben die Auswertungsergebnisse aus Kapitel 6.7.

Weiter gilt es, Berechnungen in Schleifen innerhalb der Verweis-Struktur zu vermeiden, die zu unkontrollierbaren Verfälschungen des Ergebnisses führen können, oder nicht terminieren.

Der letzte Grund besteht in der Gefahr des Auftretens kombinatorischer Explosionen, die entstehen können, wenn Verweise auf Dokumente wie z.B Inhaltsverzeichnisse gerichtet sind, die den Ausgangspunkt einer großen Anzahl weiterer Verweise bilden.

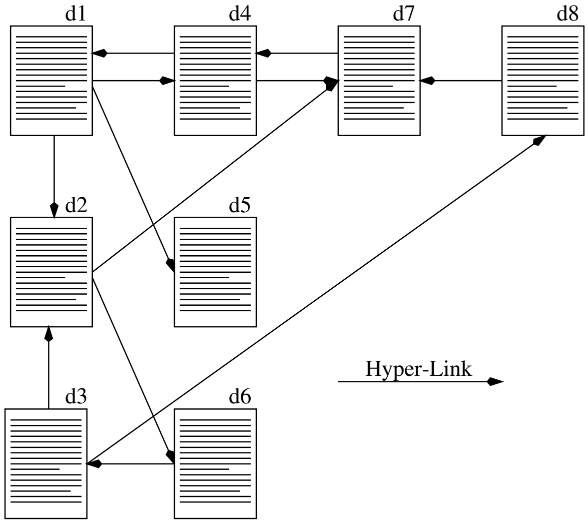

Abbildung 4.2: Hypertext-Dokumente mit Verweisen

Im Beispiel von Abbildung 4.2 läßt sich die Ähnlichkeit der Terme, die durch einen Verweis von Dokument d1 nach Dokument d4 entsteht, über eine Gewichtsfunktion bestimmen. Die Rückrichtung von d4 nach d1 leistet aus den o.g. Gründen keinen Beitrag zu einer stärkeren Gewichtung. Wie aus dem Bild zu erkennen ist, kann man auf diese Weise auch die iterative Berechnung der Schleifen d1-d2-d7-d4-d1 oder d1-d2-d6-d3-d8-d7-d4-d1 vermeiden. Ein einfacher Versuch mit einem konventionellen Thesaurus zeigt, dass der Sinnzusammenhang zweier Terme schon in der 2. Stufe einer transitiven Verknüpfung stark nachlässt. Da bisher keine Möglichkeit vorgesehen ist, eine Gewichtung für diesen Fall zu erstellen, kann das Retrieval-Ergebnis durch zufällige Einwirkungen verfälscht werden.

Es gilt nun auch hier, eine geeignete Gewichtsfunktion zu definieren, welche die Ähnlichkeit zweier Terme numerisch ausdrücken kann. Zur Bestimmung der Termgewichte bietet sich die bereits verwendete Gewichtsfunktion an:

$$w(t_i, d_k) := \frac{df(d_k, t_i) \cdot itf(d_k)}{\sqrt{\sum_{d_j \in t_i} (df(d_j, t_i) itf(d_j))^2}}$$

In die Vektor-Darstellung des Dokument-Vektorraums übertragen ergibt sich für die Terme:

$$\vec{t_i} = (w(t_i, d_1), w(t_i, d_2), \cdots, w(t_i, d_m))$$

und

$$\vec{t_j} = (w(t_j, d_1), w(t_j, d_2), \cdots, w(t_j, d_m))$$

Seien nun die Dokumente d_k mit $|d_k| = m$ und d_l mit $|d_l| = n$ durch einen Verweis verbunden (in Zeichen: $d_k \rightsquigarrow d_l$). Die Berechnung der Term-Ähnlichkeit $SIM_L(t_i, t_j)$ lässt sich mit der oben gegebenen Vektor-Darstellung der Terme folgendermaßen durchführen:

$$\begin{aligned} SIM_L(t_i, t_j) &:= \vec{t_i} \star \vec{t_j} \\ &= \begin{cases} \sum_{i=1}^{m} \sum_{j=1}^{n} w(t_i, d_k) \cdot w(t_j, d_l) & \text{für } k < l \text{ und } d_k \rightsquigarrow d_l \\ 0 & \text{sonst} \end{cases} \end{aligned} \tag{4.11}$$

Es wird nun eine weitere Term-Term-Matrix S_L (L für **L**inks) erstellt, die die durch die Verweise zwischen den Dokumenten ausgedrückten Term-Assoziationen repräsentiert:

$$S_L = \begin{array}{c} \\ t_1 \\ t_2 \\ \vdots \\ t_n \end{array} \begin{array}{c} \begin{array}{cccc} t_1 & t_2 & \cdots & t_n \end{array} \\ \begin{pmatrix} l_{11} & l_{12} & \cdots & l_{1n} \\ l_{21} & l_{22} & \cdots & l_{2n} \\ \vdots & \vdots & & \vdots \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{pmatrix} \end{array}$$

mit $l_{ij} = SIM_L(t_i, t_j)$

Die Berechnung der Matrix erfolgt auch hier in zwei Schritten für den dokumentenspezifischen Teil und die Normierung. Für den unnormierten Teil der Gewichte gilt wieder:

$$w(t_i, d_k) := df(d_k, t_i) \cdot itf(d_k)$$

Nun kann man für alle Verweise zwischen Dokumenten d_k, d_l für alle dort auftretenden Term-Paare ($t_i \in d_k, t_j \in d_l$) den dokumentspezifischen Term-Link-Ähnlichkeitsgewichts $t'_{ik} \cdot t'_{jl}$ berechnen und abspeichern. Das globale Term-Link-Ähnlichkeitgewicht $sim_L(t_i, t_j)$ ergibt sich aus der Summe aller dokumentenspezifischen Term-Link-Ähnlichkeitgewichte:

$$sim_L(t_i, t_j) = \sum_{d_k, d_l | d_k \rightsquigarrow d_l,\ k<l} t'_{ik} \cdot t'_{jl} \tag{4.12}$$

Nach Einsetzung der bereits berechneten globalen Normierungsfaktoren ergeben sich die endgültigen Einträge $SIM_L(t_i, t_j)$ der Term-Link Matrix S_L als

$$SIM_L(t_i, t_j) := \frac{sim_L(t_i, t_j)}{c(t_i) \cdot c(t_j)}. \tag{4.13}$$

$sim_L(t_i, t_j)$ ist ebenfalls symmetrisch zu ihrer Diagonalen und macht somit lediglich die Berechnung eines Dreiecks der Matrix S_L mit folgendem Algorithmus erforderlich:

Dokumentenspezifische Term-Link-Gewichte:
for every link from document d_k to document d_l **do**
  for every term t_i contained in document d_k **do**
    for every term t_j contained in document d_l **do**
      $sim_L(t_i, t_j) = sim_L(t_i, t_j) + t'_{ik} \cdot t'_{jl}$
    end
  end
end

Normalisierung der Term-Link-Gewichte:
for every good term t_i **do**
  for every good term t_i of the collection **and** $i \leq j$ **and** $sim_L(t_i, t_j) > 0$ **do**
    $SIM_L(t_i, t_j) := \frac{sim_L(t_i, t_j)}{\sqrt{c(t_i) \cdot c(t_j)}}$
  end
end

4.8 Erweiterung der Anfrage

Wie oben ausgeführt, enthalten die Thesauri aus der Dokument-Sammlung extrahiertes Wissen. Es gibt zwei Ansätze, dem Nutzer dieses Wissen zugänglich zu machen: Entweder kann man ihn im Thesaurus nach anderen Ausdrücken suchen lassen, die für seine Informationssuche relevant und nicht in der Anfrage enthalten sind, oder man benutzt den Thesaurus für eine automatische Erweiterung der Anfrage mit Termen, die den Anfrage-Termen im hier beschriebenen Sinn möglichst ähnlich sind. Das hier entwickelte System kombiniert diese Möglichkeiten. Der Nutzer kann auswählen, welchem Verfahren er den Vorzug gibt, er kann auch beide gleichzeitig oder gar nicht verwenden.

Die Idee der automatischen Anfrage-Erweiterung wird bereits seit 3 Jahrzehnten untersucht. Ziel des Verfahrens ist es, durch zusätzliche Terme eine größere Anzahl von relevanten Dokumenten zu finden. Zunächst wurde die Ähnlichkeit der Terme auf der Basis der Assoziations-Hypothese berechnet, dann die Termsammlung durch das Setzen einer Ähnlichkeitsschranke in Klassen ähnlicher Terme eingeteilt [SJ71] [Mi72]. Man erweiterte eine Anfrage, indem man alle Terme derjenigen Klassen hinzufügte, die einen Anfrage-Term enthielten. Es zeigte sich jedoch, dass dieser Ansatz der Klasseneinteilung und die Annahme der Äquivalenz der Mitglieder einer Klasse zu naiv waren, um gute Ergebnisse zu erzielen. Andere Versuche beruhten auf der Basis der Nutzung linguistischer Kenntnisse, der Analyse von Grammatik und des Einsatzes von Wörterbüchern. Die Ergebnisse waren jedoch nur geringfügig besser [Gr92],[Ru92].

Nachteil der Erweiterung der Anfrage mittels der Klasseneinteilung ist, dass keinerlei Gewichtung der zur Erweiterung herangezogenen Terme vorgenommen werden kann. Außerdem ist die Erweiterung der Anfrage stark auf die *einzelnen* Terme und nicht auf die *Gesamtheit* der Anfrage bezogen. Dies lässt sich mit dem Nachschlagen eines Begriffes in einem Wörterbuch vergleichen, was in vielen Fällen keine endgültige Lösung liefert. Erst durch das Wissen eines Übersetzers, der den Sinn des Textes versteht, können die passenden Terme gefunden werden und zu einem guten Ergebnis führen. Ein zur Erweiterung herangezogener Term sollte daher dem *Konzept* und nicht einem *einzelnen* Term der Anfrage ähnlich sein.

Wenn Information über die Relevanz verfügbar ist, lassen sich globale Informations-Strukturen wie Pseudo-Thesauri [Sal80] oder maximale aufspannende Bäume [SR83] konstruieren. Eine Anfrage-Erweiterung mit diesen Mitteln hängt stark von den Relevanz-Informationen des Nutzers ab. Sie ergab bei entsprechenden Untersuchungen keine durchgängige Verbesserung [SR83].

Ein gravierender Nachteil all dieser Verfahren besteht im dort zugrundeliegenden Ansatz, Dokumente durch Terme zu repräsentieren, deren Bedeutung durch die Verwendung im allgemeinen Sprachschatz festgelegt ist. Hier besteht, wie oben erwähnt, die Gefahr, dass die eigentlich gemeinte Bedeutung der Terme in einer Dokumenten-Sammlung nicht oder nur teilweise dieser allgemeinen Verwendung entspricht. In der Konsequenz repräsentieren u.U Terme anderen Inhalt als den der zu verwaltenden Dokumente. Durch die hier vorgenommene Indexierung durch die Dokumente findet nur das tatsächlich in der Sammlung vorhandene Wissen Verwendung.

Im Folgenden wird nun beschrieben, wie sich aus den Wissensdatenbanken Terme zur Erweiterung einer Anfrage gewinnen lassen.

Sei TVR der Vektorraum, der durch die aus der Dokument-Sammlung extrahierten Index-Terme definiert wird. Sei DVR der durch die einzelnen Dokumente definierte Vektorraum. Eine Anfrage q läßt sich im Vektorraum TVR durch den

Vektor

$$\vec{q} = (w_1(t_1), w_2(t_2), \cdots, w_n(t_n)) \tag{4.14}$$

beschreiben. Weil der Term-Term-Thesaurus die Ähnlichkeit der Terme im Vektorraum DVR enthält, wird der Vektor $\vec{q}$ von TVR nach DVR übertragen. Man kann somit die Ähnlichkeit eines Terms mit der Anfrage bestimmen. Jeder Term t_i der Anfrage läßt sich durch einen Einheitsvektor $\vec{t_i}$ im DVR darstellen, der durch eine Anzahl von Dokumenten definiert ist. Das in der Anfrage durch den Term t_i ausgedrückte Konzept hat also die Gewichtung $w_i(t_i) \star \vec{t_i}$ für die Anfrage.

Angenommen, das Konzept der Anfrage werde nur durch deren Terme ausgedrückt, dann ergibt sich für den dieses Konzept repräsentierenden Vektor $\vec{q_c}$ im Vektorraum DVR der virtuelle Term-Vektor:

$$\vec{q_c} := \sum_{t_i \in q} w_i(t_i) \cdot \vec{t_i} \tag{4.15}$$

Die Ähnlichkeit zwischen einer Anfrage q und einem Term t bezeichnen wir mit $simqt(q,t)$. Als Ähnlichkeitsmaß ergibt sich das skalare Vektorprodukt zwischen $\vec{q_c}$ und $\vec{t}$:

$$simqt(q,t) := \vec{q_c} \star \vec{t} = (\sum_{t_i \in q} w_i(t_i) \cdot \vec{t_i}) \star \vec{t} = \sum_{t_i \in q} w_i(t_i) \cdot \vec{t_i} \star \vec{t} \tag{4.16}$$

Das Produkt $\vec{t_i} \star \vec{t}$ ist als Eintrag $SIM_T(t_i, t)$ und $SIM_L(t_i, t)$ in den beschriebenen Term-Term-Thesauri bekannt und somit schon berechnet. Die Berechnung der Term-Ähnlichkeiten erfolgt somit durch

$$simqt_T(q,t) := \sum_{t_i \in q} w_i(t_i) \cdot SIM_T(t_i, t) \tag{4.17}$$

und analog

$$simqt_L(q,t) := \sum_{t_i \in q} w_i(t_i) \cdot SIM_L(t_i, t) \tag{4.18}$$

für die sich aus dem Link-Thesaurus ergebenden Werte.
Älle Terme der Dokument-Sammlung lassen sich nun in der Reihenfolge ihrer Wichtigkeit bezüglich der Anfrage q durch ihren $simqt_T$- und $simqt_L$-Wert anordnen. Das Gewicht $weight(q,a)$ eines zusätzlichen Such-Terms a berechnen wir als Funktion von $simqt$. Im Term-Term-Thesaurus ergibt sich:

$$weight_T(q,a) := \frac{simqt_T(q,a)}{\sum_{t_i \in q} w_i(t_i)} \tag{4.19}$$

mit $0 \leq weight_T(q,a) \leq 1$.

Die Anfrage q wird also durch die Anfrage $\vec{q_{eT}} = (a_1, a_2, \cdots, a_n)$ erweitert. Analog gilt für den Term-Link-Thesaurus:

$$weight_L(q, b) := \frac{simqt_L(q, b)}{\sum_{t_i \in q} w_i(t_i)} \tag{4.20}$$

mit $0 \leq weight_L(q, b) \leq 1$.

Hier erfolgt die Erweiterung der Anfrage q durch $\vec{q_{eL}} = (b_1, b_2, \cdots, b_n)$. Weil die Leistung eines IR-System erheblich von der Anzahl der in der Anfrage-Erweiterung übergebenen Terme abhängt, ist hier eine Begrenzung erforderlich. Die Einführung eines Schwellenwertes für das Gewicht lässt keine Voraussage über die Anzahl der zur Anfrage zugefügten Terme zu. Daher erfolgt die Vorgabe eines Wertes $r \in I\!N$, der die r Terme mit den höchsten $simqt(q, a)$- und analog $simqt(q, b)$-Werten zur Erweiterung der Anfrage ausfiltert.

Als Ergebnis des beschriebenen Verfahrens erhält man neben dem Anfrage-Vektor zwei weitere Vektoren aus den Thesauri, die jeweils r den Anfrage-Termen ähnlichsten Terme enthalten.

Der folgende Abschnitt beschreibt nun, wie die Vektoren geeignet kombiniert werden können.

4.9 Kombination der Anfrage-Erweiterungen

Die Leistung eines Retrievals läßt sich durch die Kombination verschiedener Retrieval-Ergebnisse aus unterschiedlichen Retrieval-Algorithmen signifikant verbessern. Der Grund dafür ist, dass unterschiedliche Algorithmen unterschiedliche Eigenschaften der Dokumenten-Sammlung hervorheben und somit eine breitere Wissens-Extraktion erfolgt. Es gibt -wie auch in den anderen Bereichen des IR- eine Reihe von Ansätzen: Etwa Lewis' und Crofts [LC90] Vergleich von Term-, Phrasen- und Konzept-Klustern oder das Bayes'sche CEO-Modell von Thomson [Th93].

Das hier entwickelte System bietet durch Kombination der Assoziations-Thesauri die neuartige Möglichkeit, den semantischen Suchausschnitt in der Datenbank verschieben zu können.

Es kommt, wie auch in den meisten oben erwähnten Methoden, ein Linearkombinations-Modell zum Einsatz. Man kann bei starker Gewichtung der Anfrage-Terme syntaktisch sehr eng am Konzept der Anfrage suchen lassen. Bei zunehmender Gewichtung der Term-Assoziationsmatrix und dann folgend der Link-Assoziationsmatrix, ist es möglich, die Suche in ein zunehmend ausgedehnteres semantisches Umfeld der Anfrage zu verlagern.

Analog zum im Kapitel 4.8 beschriebenen Verfahren sei die Anfrage $\vec{q}$ gegeben als

$$\vec{q} = (q_1(t_1), q_2(t_2), \cdots, q_n(t_n))$$

mit $n = dim(TVR)$ und $q_1, \cdots, q_n$ als den Gewichten für die Anfrage-Terme. Sei nun weiter $1 \leq i \leq n$. Das System erzeugt zu jedem $t_i \in \vec{q}$ mit $q_i > 0$ einen Vektor $\vec{d_i}$ mit zu t_i assoziierten Termen und den zugehörigen Gewichten

$$\vec{d_i} = (d_{i1}(t_1), d_{i2}(t_2), \cdots, d_{in}(t_n))$$

aus der Term-Matrix und einen weiteren Vektor $\vec{l_i}$ mit

$$\vec{l_i} = (l_{i1}(t_1), l_{i2}(t_2), \cdots, l_{in}(t_n))$$

aus der Link-Matrix. Sei nun k die Anzahl der Anfrage-Terme. Dann ergeben sich aus den Anfrage-Erweiterungen zwei Matrizen $\mathbf{T}$ und $\mathbf{L}$ mit den Einträgen

$$\mathbf{T}_{ij} = t_{ij}(t_i) \text{ und } \mathbf{L}_{ij} = l_{ij}t_i$$

mit $1 \leq j \leq k$. Zur Gewichtung der einzelnen Teile der Anfrage-Erweiterung seien nun w für die Anfrage, wt für die Term-Matrix und $wl \in IR$ für die Link-Matrix gewählt. Dann erfolgt die Gewichtung:

$$\vec{q'} = w \cdot \vec{q} \quad , \mathbf{T}' = wt \cdot \mathbf{T} \text{ und } \mathbf{L}' = wl \cdot \mathbf{L} \tag{4.21}$$

Durch geeignete Wahl der Gewichte w, wt und wl kann nun die oben erwähnte Verschiebung des semantischen Suchausschnitts in der Datenbank erreicht werden.

Aus den sich ergebenden Matrizen und dem Anfrage-Vektor kann man die Einträge w_{im} mit $1 \leq m \leq 2k+1$ der Gesamtmatrix $\mathbf{Q}_{EX}$ für die Anfrage-Erweiterung bilden:

$$w_{im} = \begin{cases} d_{ij} \in \mathbf{T}' & m = j \qquad j = 1, \cdots, k \\ l_{ij} \in \mathbf{L}' & m = k + j \quad j = 1, \cdots, k \\ w_i \in \vec{q'} & m = 2k + 1 \end{cases}$$

Für die Auswertung der Matrix $\mathbf{Q}_{EX}$ sind zwei Verfahren vorgesehen:

1. Verfahren: Aufsummierung der Term-Gewichte für die Spaltenvektoren:

   $$w(t_i) = \sum_{j=1}^{2k+1} w_{ij} \tag{4.22}$$

   Dies bedeutet, dass bei mehrfachem Auftreten eines Erweiterungs-Terms, nämlich in den beiden System-Antworten und dem Anfrage-Vektor, seine Gewichte zu einem Gesamtgewicht aufsummiert werden und in die Anfrage-Erweiterung eingehen.

2. Verfahren: Bildung des Maximums der Spaltenvektoren:

$$w(t_i) = max(w_{ij}) \text{ für } j = 1 \cdots 2k + 1 \tag{4.23}$$

   Hier erfolgt bei mehrfachem Auftreten eines Terms in der Anfrage-Erweiterung die Auswahl seines maximalen Gewichtes .

Nach Anwendung eines dieser Verfahren kann man nun die Menge der Anfrage-Terme auf einen vorgesehene Größe einschränken. Zur Auswahl kommen nur noch die entsprechende Anzahl von Termen mit dem höchsten Gewicht.

Wie das System mit der so erhaltenen Anfrage-Erweiterung die Bestimmung der relevanten Dokumente vornimmt, beschreibt das nun folgende Kapitel.

4.10 Auswahl der Dokumente mit der erweiterten Anfrage

Zur Auswertung der erweiterten Anfrage verwenden wir ein hochdimensionales Vektorraum-Modell, welches Dokumente und Anfragen durch Vektoren darstellt, die miteinander vergleichbar sind [Sal75].

Seien t verschiedene Terme zur Inhaltsbeschreibung zur Verfügung, so lässt sich ein Dokument D_i intern als n-dimensionaler Vektor von Paaren darstellen:

$$\mathbf{D_i} = (t_{1i}w_{t_{1i}}, t_{2i}w_{t_{2i}}, \cdots, t_{ni}w_{t_{ni}})$$

d_{ij} bezeichnet den j-ten Term des Dokuments D_i und $w_{d_{ij}}$ das zugehörige Gewicht. Grundsätzlich können in jedem Vektor alle t Terme auftreten, das Gewicht Null bezeichnet die Abwesenheit eines Terms, Werte zwischen 0 und 1 bezeichnen das aktuell zugeordnete Gewicht.

Ebenso werden die Anfragen als Vektoren gewichteter Terme dargestellt. Man kann also die Anfrage $\mathbf{Q}$ formulieren als

$$\mathbf{Q} = (q_1w_{q1}, q_2w_{q2}, \cdots, q_tw_{qt}),$$

wobei das Gewicht 0 wiederum das Fehlen eines Terms bezeichnet.

Sind nun sowohl der gespeicherte Text als auch die Informations-Anfragen durch Vektoren aus gewichteten Termen repräsentiert, kann durch Vergleich dieser Vektoren den Grad der Ähnlichkeit eines Dokument-Anfrage-Paares aufgrund der Gewichtung der übereinstimmenden Terme bestimmt werden. Man verwendet gewöhnlich das Cosinus-Maß, das als inneres Produkt der Vektor-Elemente normalisiert, durch die Vektorlänge berechnet wird:

$$sim(\mathbf{Q}, \mathbf{D}_i) = \frac{\sum_{j=1}^{t} w_{qj}w_{d_{ij}}}{\sum_{j=1}^{t}(w_{qj})^2 \sum_{j=1}^{t}(w_{d_{ij}})^2} \tag{4.24}$$

Gleichung (1) ergibt den Cosinus des Winkels zwischen den Vektoren $\mathbf{Q}$ und $\mathbf{D}_i$. Existieren keinerlei gemeinsame Terme der Vektoren, ergibt sich der Wert 0, und sind alle Terme identisch, ist das Ergebnis 1.

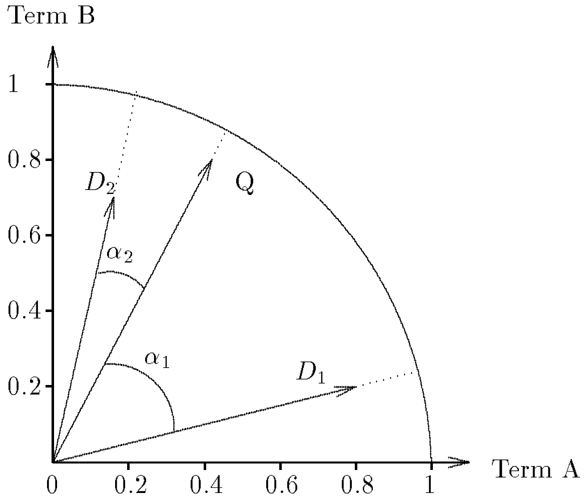

Abbildung 4.6 Vektor-Vergleich

Seien im Beispiel Abbildung 4.6 der Anfrage-Vektor $\mathbf{Q} = (0.4, 0.8)$ die Dokumenten-Vektoren $\mathbf{D_1} = (0.8, 0.3)$, $\mathbf{D_2} = (0.2, 0.7)$ und die Gewichte 0.4 und 0.8 für Term A und Term B gegeben, dann ergibt sich für die Ähnlichkeit der Anfrage Q mit Dokument D_2:

$$sim(Q, D) = \cos\alpha_2 = \frac{(0.4 \cdot 0.8) + (0.8 \cdot 0.7)}{\sqrt{[(0.4)^2 + (0.8)^2] \cdot [(0.2)^2 \cdot (0.7^2)]}} = \frac{0.64}{\sqrt{0.42}} = 0.98$$

Um die Dokumenten-Auswahl zu beschleunigen, kann man eine Matrix A berechnen, deren Zeilenvektoren sich direkt für das obige Verfahren verwenden lassen:.

$$A = \begin{array}{c c} & \begin{array}{cccc} t_1 & t_2 & \cdots & t_n \end{array} \\ \begin{array}{c} d_1 \\ d_2 \\ \vdots \\ d_m \end{array} & \left(\begin{array}{cccc} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2m} \\ \vdots & \vdots & & \vdots \\ w_{m1} & w_{m2} & \cdots & w_{mn} \end{array} \right) \end{array}$$

Zur Berechnung der Gewichte in Matrix A setzen wir die in Kapitel 4.3 beschriebene tf*idf-Methode ein, die sich im Bereich der IR-Forschung als eines der effizientesten und stabilsten aller Verfahren zur Bestimmung von Term-Dokumenten-Assoziationen erwiesen hat.

Die Auswertung nach diesem Schema ergibt im hier beschriebenen System eine in der Reihenfolge ihrer Wichtigkeit geordnete Mengen von Dokumenten: die durch den Term-Term-Thesaurus und die durch den Term-Link-Thesaurus erhaltenen Ergebnisse.

Weil die Menge der ausgegebenen Dokumente sehr groß werden kann, ist es u.U. erforderlich eine Begrenzung vorzunehmen. Dies ist durch Einführung einer Relevanzschwelle möglich, die die Ausgabe von Dokumenten geringeren Gewichts verhindert. Hier erweist sich die Fähigkeit des Systems, die gefundenen Dokumente in der Reihenfolge der Wichtigkeit ausgeben zu können als vorteilhaft. Die Wahrscheinlichkeit, dass sich unter den durch die Relevanzschwelle zurückgehaltenen Dokumenten solche mit großer Wichtigkeit befinden, ist entsprechend geringer.

4.11 Update-Operationen

In einer dynamischen Dokumenten-Sammlung kommt der Updateproblematik eine große Bedeutung zu. Weil die Beziehungen aller Terme der gesammten Kollektion in den Retrieval-Prozess einbezogen sind, ist bei Löschung oder Hinzugabe von Termen die Beachtung aller Beziehungen dieser Terme zu anderen Termen erforderlich. Dies macht je nach Umfang des Updates eine teilweise oder komplette Neuberechnung der Thesauri notwendig.

Die Dokumente der Sammlung sind durch Index-Dateien repräsentiert, in denen alle für die Repräsentation des jeweiligen Dokuments verwendeten Indexterme enthalten sind. Eine regelmäßige Untersuchung der Web-Adressen ergibt, welche Dokumente gelöscht, verändert oder hinzugefügt wurden. Man kann die für eine Entfernung bei Löschung eines Dokuments benötigten Informationen aus den zugehörigen Index-Dateien entnehmen. Bei Veränderung oder Hinzugabe von Dokumenten, können wie bei der Erst-Installation neue Index-Dateien erstellt werden, nur die betroffenen Dateien erfordern eine Neuberechnung.

Die Leistungsfähigkeit eines HIR-Systems hängt besonders vom Grad der Aktualität der verwalteten Dokumente ab. Je größer die Anzahl von Updates ist, desto aktueller bleibt der Datenbestand der Wissenbasis. Die mögliche Anzahl von Updates ist eine Funktion der dazu notwendigen System-Belastung. Je geringer diese ist, desto öfter kann man ohne Beeinträchtigung der anderen Nutzer Aktualisierungen vornehmen.

Um den Aufwand für Update-Operationen zu senken, kann man versuchen, möglichst viele Vorberechnungen durchzuführen, deren Ergebnisse direkt für das Update zu verwenden sind und keine neuerliche Systembelastung verursachen. Im hier gegebenen Fall bietet sich an, die Bestimmung der Term-Gewichte in zwei Schritte zu unterteilen: Zunächst lediglich die Berechnung des dokumentenspezifischen Teils t'_{ik} der in 4.5 gegebenen Gewichtsfunktion, im zweiten Schritt die Normierung durch das globale Term-Term-Ähnlichkeitgewicht. Für alle Terme, die eine Update-Operation nicht hinzufügt oder löscht, bleibt der dokumentenspezifische Teil des Gewichtes erhalten und lässt sich ohne Neuberechnung für die folgende Normierung einsetzen.

Bei einem Update sind im hier beschriebenen System beide Thesauri zu berechnen. Zur Neuberechnung der nicht normierten Gewichte $sim(t_i, t_j)$ muss man lediglich die aus der Sammlung entfernten oder hinzugefügten Terme beachten:

$$\begin{aligned} sim(t_i, t_j) = sim(t_i, t_j) \quad & + \sum_{\text{neue Dokumente } d_k} t'_{ik} \cdot t'_{jk} \\ & - \sum_{\text{entfernte Dokumente } d_k} t'_{ik} \cdot t'_{jk} \end{aligned} \tag{4.25}$$

Analog dazu erfolgt die Aktualisierung der Normierungskoeffizienten:

$$c(t_i) = c(t_i) + \sum_{\text{neue Dokumente } d_k} {t'_{ik}}^2 - \sum_{\text{entfernte Dokumente } d_k} {t'_{ik}}^2 \tag{4.26}$$

Kapitel 5

Aufbau des Systems

Der Aufbau des Systems muss den in einem solchen dynamischen Umfeld wie dem WWW gegebenen Besonderheiten Rechnung tragen. Eine kurze Zusammenfassung soll zeigen, welchen Problembereichen besondere Beachtung zukommen muss.

5.1 Besonderheiten im WWW-Umfeld

Die in einem vernetzten Hypermedia-Umfeld zu beachtenden Faktoren und Problemfelder lassen sich in folgende Kategorien einteilen:

1. Nutzer:
   Die Verteilung des Datenbestandes über ein so heterogenes Netz wie das Internet hat zur Folge, dass auf die Ansprüche verschiedenartigster Nutzergruppen eingegangen werden muss. Sowohl auf der Seite der Informations-Anbieter, als auch auf der der Informations-Sucher ist ein breites Spektrum an unterschiedlicher Behandlung und Bewertung von Informationen zu berücksichtigen.

   Auf der Seite der Informationssucher wird der versierte Fachmann andere Ergebnisse bei einer Anfrage an das System erwarten als der mit dem Umgang von Informationssystemen nicht vertraute Laie. Das System sollte also in der Lage sein, dem Experten präzise Antworten zu liefern, während der Laie mehr Hilfe zur Informationssuche benötigt.

   Andererseits ist es durch die Verbreitung von vielen Tools zur Erstellung von Hypermedia-Dokumenten nicht nur für einen geübten Programmierer möglich, sein Informations-Angebot im Netz zu präsentieren. Nahezu jeder, der in der Lage ist, eine Applikation auf einem Computer zu starten, kann seine Homepage einrichten und dem Informationsbestand des Netzes hinzufügen. Das System muss also eine Fülle von verschiedenartig strukturierten Informationsquellen zu allen denkbaren Themenbereichen verwalten und sollte keine konstruktionsbedingten Unterschiede zwischen diesen machen.

Nicht zuletzt ist es durchaus möglich, dass in einem Netz unterschiedliche Landessprachen Verwendung finden. Falls es nicht direkt in einem englischsprachigen Umfeld installiert ist, befinden sich meist Dokumente, die in der entsprechenden Landessprache und solche, die in englischer Sprache verfasst sind, im Informationsbestand.

2. Systemkapazität:
   Je genauer und tiefergehend eine Informationserschließung erfolgen soll, desto mehr System-Ressourcen, wie Festplatten- oder Hauptspeicherkapazität werden benötigt. Weil sehr viele Themenbereiche ins System Aufnahme finden, muss auch sehr viel Material zur Verfügung stehen, um befriedigende Auskünfte zu ermöglichen. Auf der anderen Seite sollen aber auch die Antwortzeiten des Systems auf Anfragen in einem verträglichen Rahmen bleiben, weil Nutzer größere Wartezeiten nicht akzeptieren.

   Es ist also notwendig, im System einen Kompromiss aus Tiefe der Informationserschließung und Systemkapazitäten zu finden, um einerseits gute Retrieval-Ergebnisse und andererseits kurze System-Antwortzeiten zu liefern.

3. Veränderlicher Dokumenten-Bestand:
   Das typische Einsatzgebiet des Systems ist eine WWW-Umgebung, daher verändert sich der zu verwaltende Dokumentenbestand ständig. Neue oder modifizierte Dokumente müssen in das System ebenso integriert werden, wie gelöschte Dokumente samt dem darin enthaltenen Wissen zu entfernen sind. Das System muss daher einfache und schnelle Update-Operationen ermöglichen, denn Updates sind immer mit einer Belastung des Computer-Netzes verbunden und müssen relativ häufig durchgeführt werden, wenn der Nutzer einen aktuellen Informationsstand vorfinden soll.

   Weiterhin muss bei Updates garantiert sein, dass die Systemkapazität nicht durch unkontrollierte Aufnahme neuer Dokumente überlastet wird. Weil ein weitgehender Automatisierungsgrad erforderlich ist, müssen geeignete Begrenzungskriterien vorhanden sein.

4. Der ständige Fortschritt in Forschung und Entwicklung verlangt es, Veränderungen und Anpassungen des Systems auch auf der Ebene der verwendeten Methoden und Strategien vornehmen zu können. Im Bereich von Bild-Erkennung und Sprachanalyse sind ständige Neuerungen zu erwarten, die Möglichkeit der Aufnahme solcher Weiterentwicklungen ist somit erforderlich.

   Ebenso ist die Lage im syntaktischen Bereich. Weil man ständig Erweiterungen und Verbesserungen des HTML-Standards vornimmt, müssen Anpassungen leicht und schnell durchführbar sein. Ein Problem entsteht auch durch den Wettbewerb unterschiedlicher Hersteller, die häufig eigene, vom Standard abweichende Software-Lösungen in ihren Produkten verwenden. Auch diese müssen berücksichtigt werden, wenn das HIR-System universell einsetzbar sein soll.

Die folgenden Abschnitte beschreiben nun die einzelnen Teile des Systems.

5.2 Installationskomponenten

Die Installation des Systems erfolgt in zwei Schritten: Im ersten liest das System die Hypermedia-Daten ein, transformiert sie in eine systeminterne Darstellung und speichert sie ab. Aus den internen Daten erstellt es die Wissensdatenbanken und speichert diese mit allen notwendigen Hilfsdateien ab. Alle zu diesem Zeitpunkt möglichen statistischen Berechnungen werden durchgeführt, um zur Laufzeit des Systems kurze Antwortzeiten zu erhalten. Die Durchführung dieser Operationen ist nur bei der Erst-Installation erforderlich und stellt die größten Anforderungen an System-Ressourcen und Netzperformance.

Im zweiten Schritt wird eine Komponente zur Verarbeitung von Nutzer-Abfragen bereitgestellt. Das System analysiert die Anfrage und bereitet sie auf. Das Ergebnis wird unter Einsatz der in Schritt 1 erstellten Wissensbasis für die Auswahl der Dokumente herangezogen. Schließlich präsentiert das System dem Nutzer die Suchergebnisse. Dieser kann nun entscheiden, welche Dokumente für ihn relevant sind. Einen Überblick über den Aufbau des Systems gibt Abbildung 5.1.

Zunächst erfolgt nun die Beschreibung des Aufbaus und der Wirkungsweise aller für Schritt 1 erforderlichen Komponenten.

5.2.1 Der Roboter

Das Anliefern der Web-Seiten für die Dokumenten-Kollektion erledigt ein Software-Roboter. Es handelt sich dabei um ein Programm, das über eine Internet-Verbindung eine vorgegebene Internet-Adresse besucht und deren Inhalt, das Hypermedia-Dokument, abruft. Sind seitens der besuchten Adresse keine Einschränkungen vorgeschrieben, wird das Hypermedia-Dokument nach Umwandlung in eine interne Darstellung im System abgelegt.

Der Roboter erhält bei der Initialisierung des Systems Adressen der Startseiten für den zu verwaltenden WWW-Bereich zugewiesen, die er läd und zur Verarbeitung an den Indexer weitergibt. Solche Startseiten bestehen meist aus Inhaltsverzeichnissen oder Begrüßungs-Seiten, die es ermöglichen, einen größeren Internet-Bereich zu

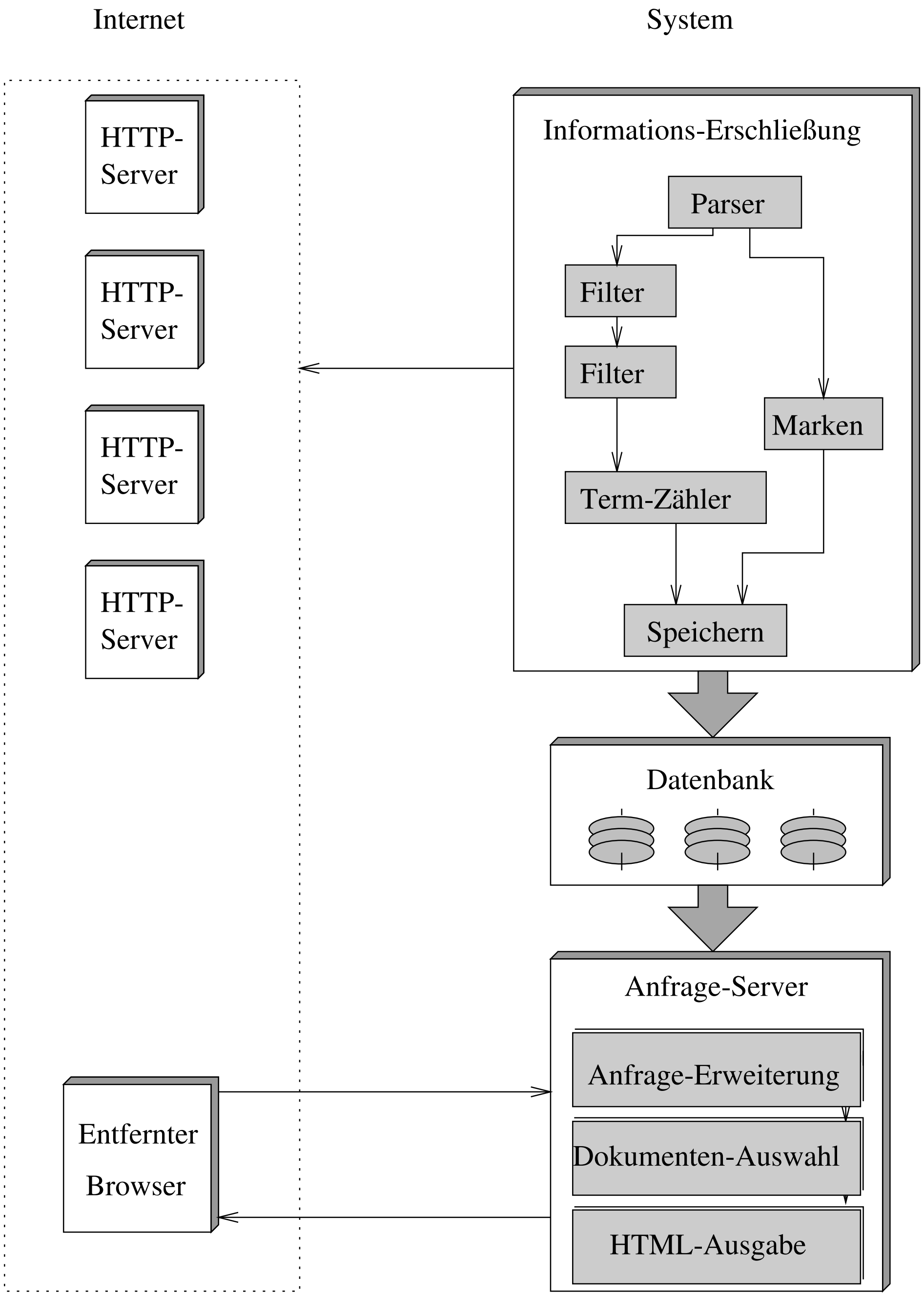

Abbildung 5.1: Schematische Darstellung der Systemkomponenten.

erreichen. Der Indexer liefert in den Web-Seiten gefundene Adressen an den Roboter zurück, der diese wiederum besucht und den Inhalt, wenn möglich, wieder an den Indexer weiterreicht. Das System muss die entstehende Schleife durch Beschränkungsvorgaben abbrechen, um eine Überschreitung der internen Kapazitäten zu verhindern. Im hier beschriebenen System wird dies durch eine Einschränkung des Wirkungsbereichs des Roboters erreicht, indem man ihm nur Zugriff auf vorgegebene Internetbereiche, i.a. bestimmte Server, gestattet.

5.2.2 Der Indexer

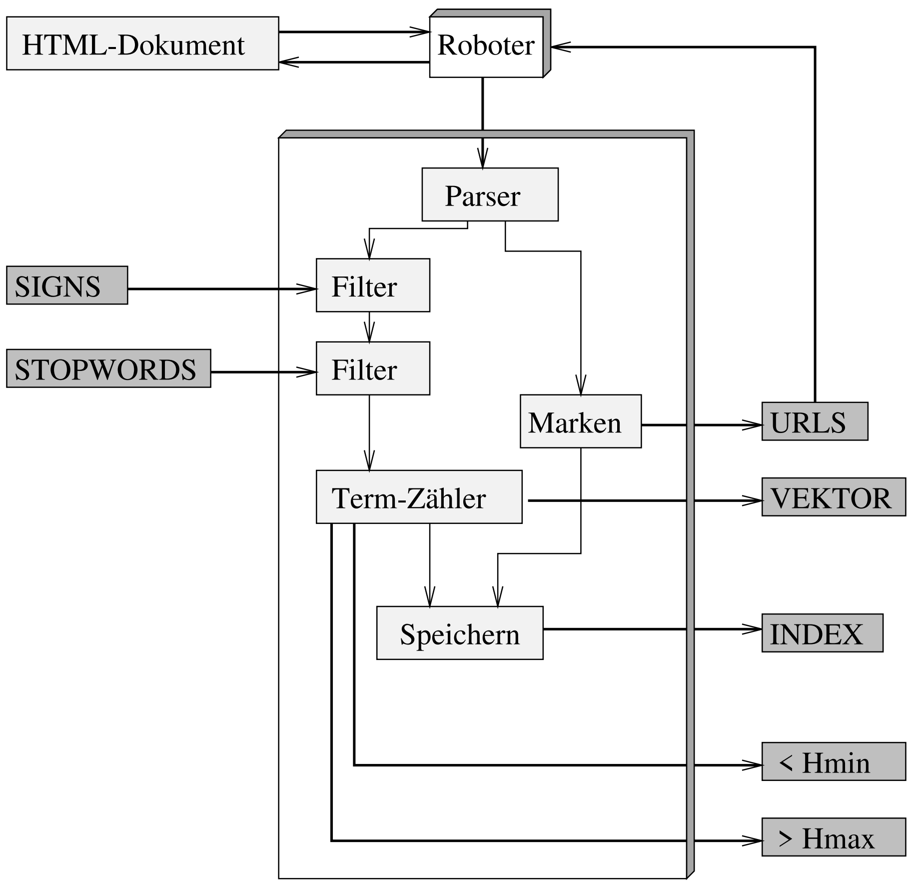

Abbildung 5.2: Die Komponenten des Indexers

Der Indexer besteht aus mehreren Teilen (s. Abb. 5.2) und arbeitet mit dem Roboter zusammen, indem er diesem nach Zerlegen und Analyse der originalen Hypermedia-Dokumente die dort enthaltenen Adressen zurückliefert. Weiterhin erstellt er Hilfsdateien, die für die späteren statistischen Berechnungen benötigt werden. Einige dieser Berechnungen sind erst möglich, wenn alle zu verwaltenden Dokumente durch das System erfasst sind, so zum Beispiel die zur Normierung benötigte Häufigkeit des Auftretens eines Terms in der gesamten Dokumenten-Sammlung, oder die Gesamtzahl der Dokumente im System.

Der Indexer liest die vom Roboter angelieferte Hypertext-Seite nach Klartext und Textmarken getrennt ein. Die Angaben innerhalb der Textmarken bleiben unverändert, der Klartext wird einer weiteren Verarbeitung unterzogen: Die HTML-Sonderzeichen der ISO 8859-1-Norm werden ausgefiltert oder im Fall deutscher Umlaute ersetzt. Weil die Hypertext-Standards einer fortlaufenden Erweiterung unterliegen, ist eine editierbare Liste der zu extrahierenden Zeichen in einer Festplatten-Datei gespeichert, um die eventuell notwendigen Anpassungen an Norm-Änderungen leicht durchführen zu können. In einem weiteren Schritt werden Zeichen wie "+", "!", ";" u.s.w. ausgefiltert, weil man für Indexterme nur Klartext verwenden kann.

5.2.3 Stoppwortlisten

Es erfolgt das Ausfiltern von Funktionswörtern (und, oder, so, der ...) und Stoppwörtern aus dem Klartext, da diese ungeeignet zur Repräsentation des Dokumenteninhalts und somit als Indexterme nicht geeignet sind. Die manuell für jede im System benutzte Landessprache erstellten Funktions- und Stoppwortlisten liegen in einer Datei auf der Festplatte.

5.2.4 Bearbeitung der Textmarken

Nun filtert das System die relevanten Informationen der Textmarken aus. Am wichtigsten sind die enthaltenen Web-Adressen, da diese an den Roboter zurückgeliefert werden müssen. Häufig sind weitere interessante Informationen in den Kopfzeilen der HTML-Dokumente enthalten, wo mittels *Meta-Angaben* Autorenschaft, eine Kurzbeschreibung des Inhalts oder charakteristische Stichwörter abgelegt sein können. Zusätzlich wird die Struktur der Überschriftenabfolge gesichert, um eventuell spätere Untersuchungen für Term-Assoziationen auf Abschnittsebene durchführen zu können.

5.2.5 Index-Dateien

Das System legt den Text zusammen mit den gefundenen Adressen und Überschriften in eine Index-Datei ab, deren Struktur die Struktur der Original-Seite widerspiegelt. Dies ermöglicht auch eventuelle spätere System-Erweiterungen, für die Informationen über den Aufbau der Seiten erforderlich sind, ohne eine komplette Neuinstallation der Index-Dateien durchführen zu müssen. Außer der Extraktion von

Wissen erlaubt die Index-Datei auch das Entfernen von Informationen aus dem System. Löscht der Autor eines Web-Dokuments seine Seite, so kann diese nicht mehr im WWW aufgerufen werden, um die enthaltenen Informationen aus der Wissensbasis zu entfernen. Mit Hilfe der noch vorhandenen Index-Datei ist es möglich, ein einfaches Update durchzuführen, eine kompletter Neuaufbau der Index-Dateien und somit auch der Wissensdatenbank wäre sonst erforderlich.

5.2.6 Erstellung des hochdimensionalen Wort-Vektors

Während der Aufnahme der vorgesehenen Dokumente, können verschiedene Werte, die für spätere statistische Berechnungen der Term-Gewichte erforderlich sind, vorberechnet werden. Sind alle in der Dokumenten-Sammlung vorkommenden Terme eingelesen, erfolgt die Erstellung des hochdimensionalen Wortvektors, der den Term-Vektorraum für die Indexterme aufspannt. Ebenso ist nun bekannt, wie häufig ein Term sowohl in einem einzelnen Dokument, als auch der gesammten Sammlung auftritt. Mit diesen Informationen lässt sich die Menge der "guten Terme" bestimmen, die man für die Indexierung verwenden kann.

5.3 Das Anfrage-Modul

Um die in den Wissensbasen gespeicherten Informationen abrufen zu können, ist die Installation eines Anfrage-Moduls erforderlich. Abbildung 4.5 zeigt die einzelnen Komponenten des hier beschriebenen Systems und deren Zusammenwirken.

Ruft ein Nutzer die zur Suchmaschine gehörende Internet-Adresse auf, so erscheint ein Formular (Kap A.2.1) auf seinem Browser, welches ihm erlaubt, ohne syntaktische Vorgaben Anfragen zu formulieren. Ist dies erfolgt, kann er das Formular durch Betätigung eines Bestätigungs-Buttons zum System senden. Der hier installierte Internet-Server leitet es an ein gleichzeitig als Internet-Client (Kap A.2.2) ausgelegtes CGI-Skript (Kap 3.6.3) weiter, welches seinerseits die im URL-Format (Kapitel A.2.3) empfangene Anfrage dekodiert und an den Anfrage-Server übergibt. Dieser führt die in 4.8 beschriebene Anfrage-Erweiterung durch und ermittelt relevante Dokumente wie in 4.10 gezeigt. Nach Umwandlung ins Html-Format geht die Antwort auf dem gleichen Wege zurück, den die Anfrage nahm und erscheint im Browser des Nutzers. Zusätzlich zur Auswahl der Dokumente präsentiert der Browser eine Anzahl der Anfrage ähnlicher Terme aus der Wissensbasis, die der Nutzer zur weiteren Spezifizierung seiner Anfrage verwenden kann.

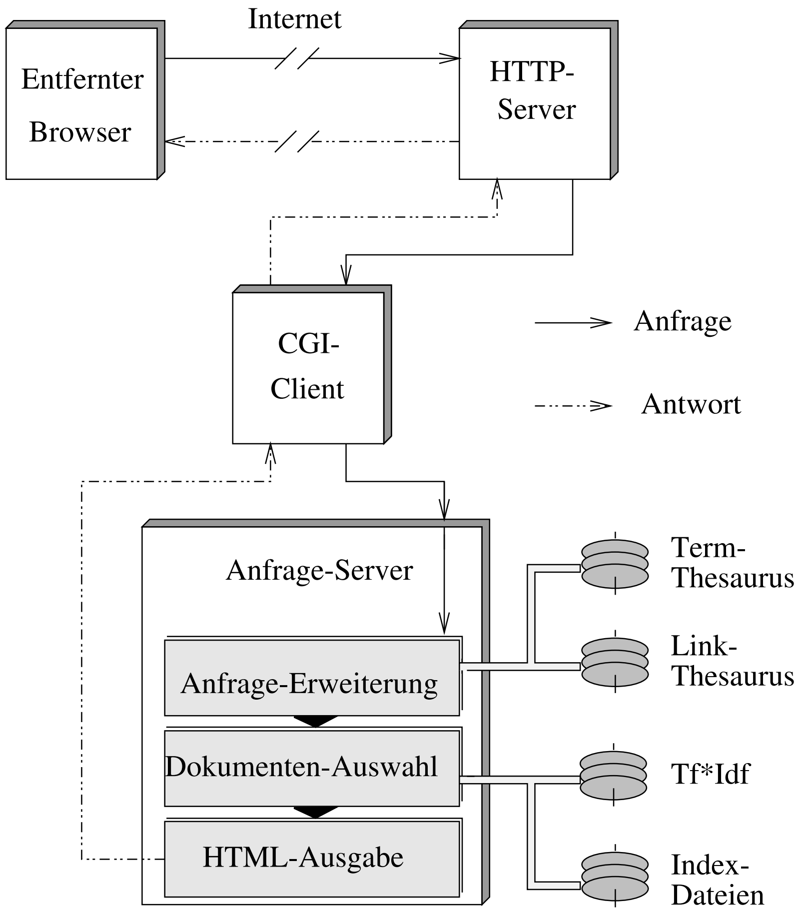

Abbildung 4.5 Die Komponenten des Anfrage-Moduls.

Kapitel 6

Auswertungen und Experimente

6.1 Allgemeines

Die Auswertung von IR-Systemen ist ein schwieriges Problem. Zu seiner Lösung betreibt man viel Forschungsaufwand. Jährlich erscheint eine große Anzahl von Artikeln zu diesem Thema, ständig publiziert man neue Methoden, die bessere Meßergebnisse versprechen, als die bisher verwendeten. Grundsätzlich stellen sich in diesem Zusammenhang drei Fragen:

1. Wozu Auswertungen durchführen,
2. was auswerten und
3. wie auswerten ?

Die erste Frage ist vorwiegend sozialer und ökonomischer Natur. Der soziale Teil bezieht sich vorwiegend auf die Messung des Nutzens, den man aus der Verwendung vollautomatischer und interaktiver IR-Systeme im Gegensatz zum Einsatz traditioneller Informationsquellen ziehen kann. Weitere Studien gehen in diese Richtung, doch die Ergebnisse sind nur schwer interpretierbar. Der ökonomische Teil zielt auf die Kostenfrage, aber auch hier ist selbst eine einfache Aussage nur schwer zu machen. Kosten für den Einsatz von Computern lassen sich sicherlich leicht bestimmen, diejenigen für personellen Aufwand jedoch nur mit erheblicher Ungenauigkeit.

Die zweite Frage reduziert sich auf die Messbarkeit der System-Eigenschaften, die die Ansprüche eines Nutzers erfüllen. Schon 1966 formulierte hier Cleverdon sechs Hauptkriterien [Cl66]:

1. die Erfassung von Informationen, d.h., den Grad des Enthaltenseins relevanter Informationen im System,
2. die zeitliche Verzögerung zwischen Suchanfrage und System-Antwort,

3. die Form der Ausgabe-Präsentation,

4. den Aufwand des Nutzers zur Erlangung von System-Antworten,

5. den *Recall*, d.h., den Anteil des insgesamt im System verfügbaren relevanten Materials für eine Anfrage, und

6. die *Precision*, d.h., den Anteil des relevanten Materials in einer System-Antwort.

In Forscherkreisen herrscht Einigkeit darüber, dass für die Kriterien 1 - 4 bereits endgültige Abschätzungen vorliegen. Meist bestimmen heute die Werte Recall und Precision die *Effizienz* eines Retrieval-Systems. M. a. W., man misst also die Fähigkeit eines Systems, relevante Dokumente zu einer Anfrage zu liefern, gleichzeitig jedoch unrelevantes Material zurückzuhalten. Es gab viele Diskussionen darüber, ob Recall und Precision in der Tat zur Messung der Effizienz eines IR-Systems geeignet seien. Eine populäre Alternative war die Messung von Recall und Fall-out (der Anteil unrelevanter Dokumente in einer Antwort), dennoch erfordern alle vorgeschlagenen Möglichkeiten in irgendeiner Form die Bestimmung der *Relevanz* des ermittelten Materials. In den folgenden Abschnitten soll daher noch genauer auf die Bedeutung der Begriffe Relevanz, Recall und Precision eingegangen werden.

Die Beantwortung der dritten Frage ist stark technischer Natur. Tatsächlich hängt die Technik der Messung von Retrieval-Effizienz hauptsächlich von der betrachteten Retrieval-Strategie und der Form der Ausgabe ab. Die späteren Ausführungen in dieser Arbeit sollen zeigen, welche Anpassungen erforderlich sind, um eine Bestimmung der Effizienz des hier entwickelten Systems zu ermöglichen.

6.2 Relevanz

Relevanz ist ein subjektiver Begriff. Nutzer differieren oft in der Auffassung über Relevanz oder Nicht-Relevanz ausgewählter Dokumente für eine gegebene Anfrage. Dennoch sind die Abweichungen in diesen Auffassungen nicht so erheblich, dass Experimente mit Dokumenten-Kollektionen bezüglich der Einschätzung von Relevanz von Systemantworten für Anfragen mit aussagekräfigen Ergebnissen undurchführbar sind.

Für solche Experimente erstellt man einen Katalog, der aus Anfragen beliebiger System-Nutzer zusammengestellt ist. Eine Gruppe von Experten auf dem entsprechenden Gebiet nimmt dann eine Einschätzung der Relevanz vorhandener Dokumente für die Fragen im Katalog vor. Es existiert damit eine Anzahl von Anfragen, zu denen "korrekte" Antworten bekannt sind, und die für Messungen an verschiedenen IR-Systemen zur Verfügung stehen. Im Gebiet der IR herrscht üblicherweise die

Auffassung, eine unter experimentellen Bedingungen gut funktionierende Retrieval-Strategie sei auch mit guten Ergebnissen in einem operationalen Umfeld, dort einsetzbar, wo die Relevanz zu Beginn unbekannt ist.

6.3 Recall und Precision

Die am häufigsten verwendeten Maße zur Beurteilung der Güte eines Information Retrieval Systems sind Recall und Precision [Kai97]. Die Gründe hierfür liegen einerseits in der weitverbreiteten Verwendung und andererseits im tiefergehenden Verständnis dieser Größen. Abbildung 6.1 zeigt eine typische Retrieval-Situation, in der der Satz der ermittelten Dokumente sowohl aus für die Anfrage relevanten, als auch aus solchen ohne Relevanz besteht. Der Recall R stellt das Maß für die

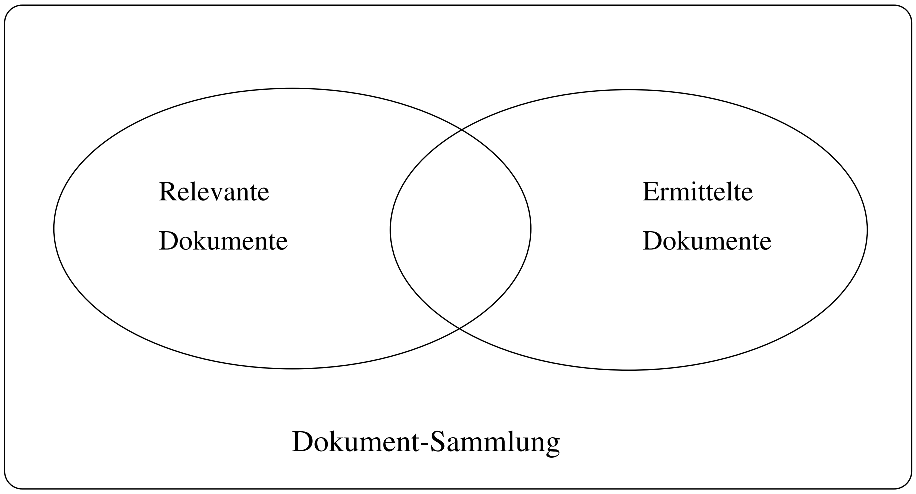

Abbildung 6.1: Typische Retrieval-Situation.

Vollständigkeit des Retrieval-Ergebnisses dar und ist definiert als das Verhältnis zwischen den gefundenen relevanten Dokumenten a und der Gesamtanzahl der im Dokumentenbestand vorhandenen relevanten Dokumente g:

$$R = \frac{a}{g} \tag{6.1}$$

Der Wertebereich des Recalls geht von 0 bis 1 ($0 \leq R \leq 1$). Ein Recall von 0 bezeichnet das schlechteste Ergebnis, 1 das bestmögliche.

Das Problem beim Recall liegt in der Bestimmung der Größe g im Nenner, weil die Anzahl aller relevanten Dokumente eines Bestandes im Allgemeinen nicht bekannt ist.

Die Gesamtanzahl g aller relevanten Dokumente in einem Dokumentenbestand läßt sich darstellen als die Anzahl a der gefundenen relevanten Dokumente vermehrt um die Anzahl c der nicht gefundenen relevanten Dokumente.

$$g = a + c \tag{6.2}$$

Damit ergibt sich für den Recall:

$$R = \frac{a}{a + c} \tag{6.3}$$

a ist leicht bestimmbar; für die Größe c jedoch muß zur Bestimmung des Nenners des Recalls ein Schätzwert angenommen werden.

Dafür entwickelte man verschiedene Methoden, um eine möglichst genaue Annäherung an die Gesamtzahl aller relevanten Dokumente bzw. die im System verbleibenden relevanten Dokumente c zu erhalten. Alle Verfahren sind in erster Linie von pragmatischen Gesichtspunkten geleitet. In den meisten Fällen wird ein relativ kleines Subset von Anfragen und des Dokumenten-Bestandes vollständig auf relevantes Material hin durchsucht und diese Zahl dann auf den tatsächlichen Dokumentenbestand hochgerechnet [WH89 , S. 35] .

Die Precision P dient zum Messen der Genauigkeit der Suche und als Indikator für die Fähigkeit des Information Retrieval Systems, nicht relevante Dokumente auszuscheiden. Die Precision ist definiert als das Verhältnis der Anzahl a der gefundenen relevanten Dokumente zur Zahl e aller gefundenen Dokumente.

$$P = \frac{a}{e} \tag{6.4}$$

Auch der Wertebereich der Precision geht von 0 bis 1 ($0 \leq P \leq 1$). Auch bei der Precision wird versucht, den Wert von P zu maximieren.

Sinnvoll ist jedoch nur eine Betrachtung beider Maße, weil der Recall die Ballastquote unberücksichigt läßt, und leicht auf das Maximum 1 gesetzt werden kann, indem man den Dokumentenbestand vollständig nachweist. In diesem Fall ist dann allerdings der Wert der Precision sehr niedrig.

Eine alleinige Betrachtung der Precision wiederum sagt nichts über die Vollständigkeit des Retrieval-Ergebnisses aus. Die Precision alleine kann man durch die Ausgabe sehr weniger Dokumente nachweisen.

Daher kommt bei vielen Evaluierungen von Retrieval-Experimenten beispielsweise auch der Recall-Precision-Graph zum Einsatz. In diesem Graph trägt man auf der x-Achse die Precision und auf der y-Achse den Recall auf und versucht so, ein Bewertungsmaß zu schaffen, das beide Größen einbezieht.

Zur Konstruktion dieses Graphen bestimmt man zu jeder Anfrage einer gegebenen Test-Kollektion die Precision-Recall-Werte. Abhängig von systembedingten Ausgabe-Parametern λ_i, wie z.B. Position in der Gewichtungswertung, Koordinations-Level (wieviele Begriffe der Anfrage treten in Anfrage und Dokument gemeinsam auf) oder Anzahl der für eine eventuell vorgesehene Anfrage-Erweiterung verwendeter Terme, berechnet man unterschiedliche Werte. Sie bilden Recall-Precision-Graphen, die die Leistung des Systems bezüglich jeder Anfrage darstellen (Siehe Abbildung 6.2).

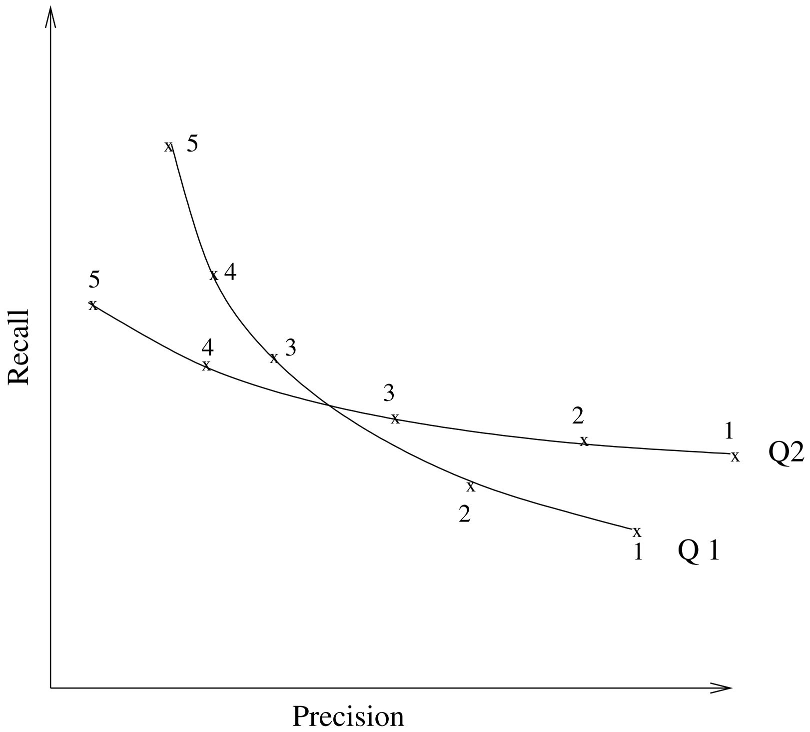

Abbildung 6.2: Precision-Recall-Graph zweier Anfragen. Die Kurven stellen die Werte für einen Ausgabe-Parameter λ dar.

Um die Gesamtleistung des Systems zu ermitteln, kombiniert man die für die Anfragen unterschiedlichen Meßwerte zu einer gemittelten Gesamtkurve. Es gibt eine Reihe von Verfahren zur Mittelwertbildung [Rij79], deren Einsatz weitgehend von der verwendeten Retrieval-Strategie abhängt. Bei einem Retrieval-System, dessen Verfahren auf der Untersuchung von Term-Gemeinsamkeiten beruht, setzt man die sogenannte *Mikro-Auswertung* ein. Das Verfahren läßt sich folgendermaßen beschreiben: Sei S eine Menge von Anfragen, dann gilt:

$$|\tilde{A}| := \sum_{s \in S} |A_s|$$

mit A_s als der Menge der für die Anfrage s relevanten Dokumente. Sei λ ein Ausgabe-Parameter, dann soll weiter gelten:

$$|\tilde{B}_\lambda| := \sum_{s \in S} |B_{\lambda_s}|$$

wobei B_{λ_s} die Menge der unter dem Ausgabe-Parameter λ erhaltenen Dokumente angibt. Die Berechnung des Recall erfolgt dann durch

$$R_\lambda = \sum_{s \in S} \frac{|A_s \cap B_{\lambda_s}|}{|\tilde{A}|}$$

Die Precision wird durch

$$P_\lambda = \sum_{s \in S} \frac{|A_s \cap B_{\lambda_s}|}{|\tilde{B}_\lambda|}$$

ermittelt.

6.3.1 Test-Kollektionen

Zur Evaluierung von IR-Systemen stehen eine Reihe von Test-Kollektionen zur Verfügung. Grundsätzlich bestehen sie aus drei unterschiedlichen Teilen:

1. Einer Menge von Dokumenten,
2. einer Menge von (natürlichsprachlichen) Anfragen und
3. einer von Experten erstellten Relevanz-Beziehung zwischen Dokumenten und Anfragen.

Die Kollektionen unterscheiden sich sowohl in der Anzahl ihrer Dokumente und Anfragen, als auch im Themengebiet, aus dem sie stammen. Die folgende Tabelle zeigt einen Überblick über die in der IR gebräuchlichsten Sammlungen:

Kollektion	Dokumente	Anfragen	Größe in MB
LISA	5,872	35	3.4
NPL	11,429	93	3.1
CACM	3,204	64	2.2
CISI	1,460	112	2.2
Cranfield	1,400	225	1.6
Time	423	83	1.5
Medline	1,033	30	1.1
ADI	82	35	0.04

Des Weiteren stehen sehr große Kollektionen, wie OSHUMED aus dem medizinischen Bereich, Reuters 21578 oder die TREC-Sammlung zur Verfügung, deren Dokumenten-Umfang im Bereich von einigen GB liegt. Weil verschiedene Kollektionen unterschiedliche Test-Parameter untersuchen, müssen die für Experimente verwendeten im Hinblick auf die Architektur und den Anwendungsbereich des zu bewertenden IR-Systems ausgewählt werden.

Für das hier entwickelte System kommen die Cranfield-, Medline- und NPL-Kollektion zum Einsatz. Das Erstellen der Assoziations-Matrizen erfordert eine hohe Rechenleistung, sehr umfangreiche Sammlungen wie Reuters oder TREC würden somit die mögliche Anzahl der untersuchten Ausgabe-Parameter verkleinern, ohne das Testergebnis zu verbessern. Ein weiterer Grund für diese Auswahl besteht darin, dass man mit den vorhandenen Kollektionen neue Eigenschaften des hier entwickelten Systems, wie z.B die Auswertung der Link-Struktur der Dokumenten-Sammlung, nicht bewerten kann, und sie daher modifizieren zu muss. Dies ist bei den großen Kollektionen nur mit erheblichem Aufwand möglich.

Leider existieren zur Zeit keine speziellen Testsammlungen für Hypermedia-Systeme. Die erhaltenen Ergebnisse können somit keine zahlenmäßig absolute Aussage über die System-Performance machen. Dennoch sind deutliche Tendenzen nachweisbar, und die Messungen zeigen sicher nur untere Grenzen der Leitungsfähigkeit des Systems auf.

6.4 Durchführen der Auswertungen

Um die Test-Kollektionen für das hier beschriebene System verwenden zu können, erfolgten einige Modifikationen: Zunächst die Umwandlung der Text-Dokumente in HTML-Format, dann die Einbindung verschiedener Link-Strukturen in die Test-Dokumente, um eine Untersuchung der Link-Datenbasis des Systems ermöglichen.

Ein wichtiger Punkt für die aussagekräftige Auswertung besteht in der Auswahl der zu untersuchenden Ausgabe-Parameter. Hierbei müssen Einsatz- und Aufgabenbereich des Systems beachtet werden. Im hier beschriebenen System basiert die Dokumenten-Auswahl auf dem bewährten und in der Praxis am häufigsten verwendeten tf*idf-Gewichtungsverfahren (Siehe Kapitel 4.3). Um Ausgangswerte zu bekommen erfolgt also zunächst eine Effizienzmessung, die lediglich die Leistung des Systems unter Einsatz dieses Verfahrens ermittelt. Danach werden Anfrage-Erweiterung und Link-Auswertungsverfahren zugeschaltet und unter den gleichen Bedingungen getestet. Zuletzt zeigt ein Vergleich der Ausgangswerte mit den unter den verschiedenen Zusatzverfahren erhaltenen Werten, wie sich die Systemleistung verändert hat.

6.5 Messergebnisse des tf*idf-Systems

Die erste Messung bestimmt die Systemleistung unter der Veränderung des Wertes h_{max}. Dieser Wert gibt die maximale Häufigkeit an, mit der ein Term in der gesamten Dokumenten-Kollektion auftreten darf, um noch als Index-Term Verwendung zu finden und hat grundsätzlichen Einfluss auf die Gesamt-Performance. Eine zu hoch gewählte Schwelle verschlechtert die Antwortzeit des Systems, weil sehr viel mehr Terme -gerade die häufig auftretenden - verwaltet werden müssen. Ist die Schwelle zu niedrig gewählt, fehlen aussagekräftige Index-Terme, der Informationsgehalt der Assoziations-Matrizen verschlechtert sich und damit auch das mögliche Retrieval-Ergebnis.

Der Wert für h_{min} ist auf 2 festgesetzt. Terme, die nur einmal in der Kollektion auftreten, können auch nur in einem einzigen Dokument vorkommen. Das System gibt dieses Dokument aus, und versieht es mit dem zugehörigen tf*idf-Gewicht. Versuche, bei denen die Einzelterme in die Indexmenge aufgenommen wurden [Qiu94] ergaben eine Verschlechterung des Retrieval-Ergebnisses.

Zur Messung der Systemleistung ermitteln wir nun Recall und Precision bei verschiedenen Werten für h_{max} in Abhängigkeit von der Anzahl N der in der Kollektion enthaltenen Dokumente. Die verwendeten Werte sind $N/50$, $N/20$, $N/10$ und N. Zu jedem Wert erfolgt eine Bestimmung von Recall und Precision unter Veränderung der Menge der ausgegebenen Dokumente. Das System kann die Anzahl der ausgegebenen Dokumente in zwei Weisen kontrollieren: Durch Ordnen der Dokumente nach ihrer Relevanz und Ausgabe der gewünschten Anzahl, oder durch Angabe eines Gewichts-Schwellwerts unterhalb dessen kein weiterer Nachweis von Dokumenten mehr erfolgt.

Abbildung 6.3 zeigt, dass ein Absenken des Wertes von h_{max} unter einen Wert von $N/10$ ein starkes Absinken der Effizienz des Retrievals bewirkt. Oberhalb dieses Wertes wird jedoch kein nennenswerter Gewinn mehr erzielt. Weiter ist zu erkennen, dass ein niedriger Schwellenwert für das Dokument-Gewicht einen hohen Recall zur Folge hat, jedoch die Precision schlechtere Werte annimmt. Der Grund dafür ist eine zunehmende Anzahl ausgegebener Dokumente bei niedrigerem Schwellenwert. Die nächste Messung untersucht, wie in Abbildung 6.4 gezeigt, das Verhalten des Recall-Precision-Graphs bei Veränderung der Anzahl ausgegebener Dokumente, der Schwellenwert spielt hierbei keine Rolle. Das Verhalten von Recall und Precision zeigt ein ähnliches Verhalten wie zuvor bei der Änderung des Schwellenwertes. Der

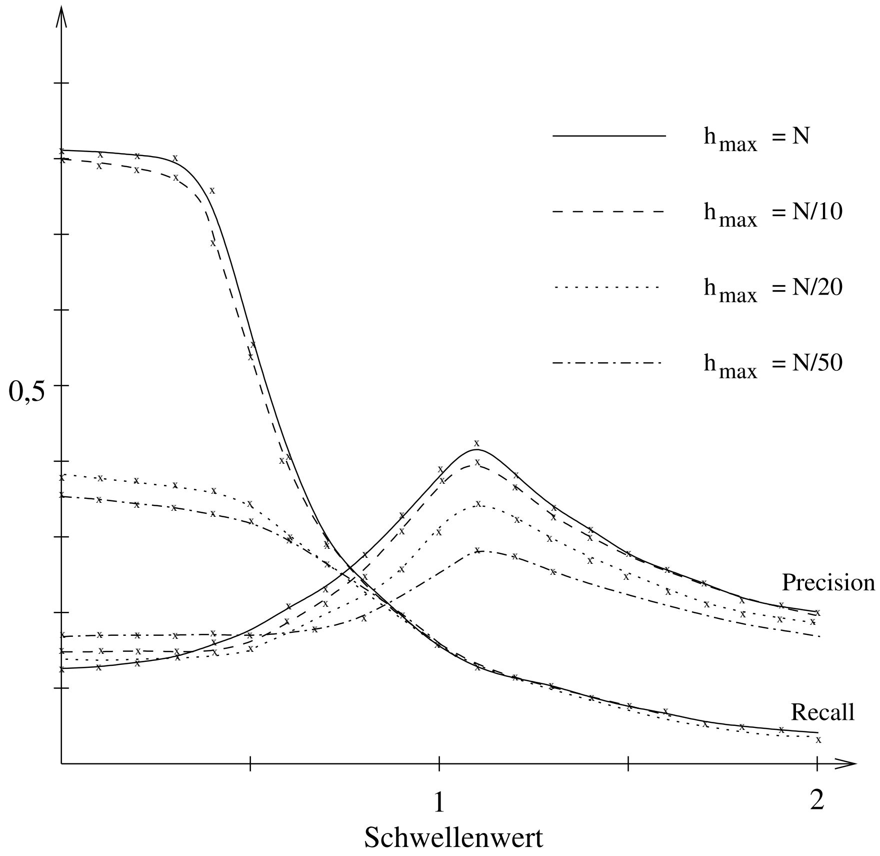

Abbildung 6.3: Recall-Precision-Graph bei unterschiedlichen Werten von h_{max} am Beispiel der MED-Kollektion.

Einfluss durch die Wahl des Wertes von h_{max} zeigt sich hier ebenfalls sehr deutlich.

6.6 Messergebnisse der Term-Assoziations-Matrix

Mit den bis hierher gewonnenen Ergebnissen erfolgt nun eine Auswertung der zusätzlichen Systemkomponenten. Als Voreinstellungen sind gewählt:

- $h_{max} = N/10$,
- Gewicht auszugebender Dokumente > 0.3,
- Anzahl der auszugebenden Dokumente $= 50$

Die Auswahl der Parameter ist durch den Einsatz des Systems in einem WWW-Umfeld begründet. Der Wert für h_{max} garantiert die Begrenzung der Datenbasis-Größe auf ein verwaltbares Maß, ohne den Informationsgehalt zu stark einzuschrän-

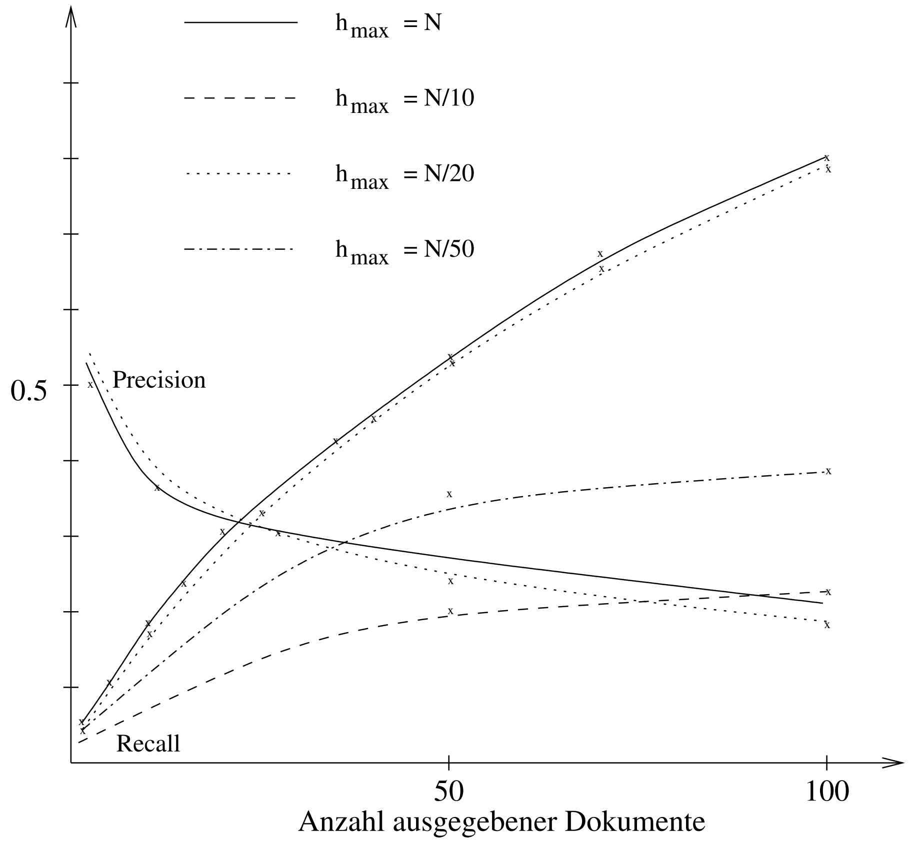

Abbildung 6.4: Recall-Precision-Graph bei unterschiedlicher Anzahl ausgegebener Dokumente .

ken. Der Schwellenwert 0.3 verhindert die Ausgabe von wenig relevanten Dokumenten. Da es einem Nutzer zumutbar ist, 50 Dokumente zu durchsuchen, wird die Anzahl der ausgegebenen Dokumente auf diesen Wert beschränkt.

Die folgenden Messungen untersuchen die System-Performance bei unterschiedlichen Test-Kollektionen. Der Ausgangswert liegt im Ursprung des Koordinatensystems für die Messkurven und beschreibt die Leistung des Systems unter Einsatz des reinen tf*idf-Systems, also ohne Anfrage-Erweiterung. Die Messkurven beschreiben die Verbesserung der Performance bei zunehmender Zugabe von Termen in die Anfrage-Erweiterung.

In den Abbildungen 6.5, 6.6 und 6.7 sind die Ergebnisse bei Einsatz der Term-Term-Assoziationsmatrix in den verschiedenen Kollektionen dargestellt.

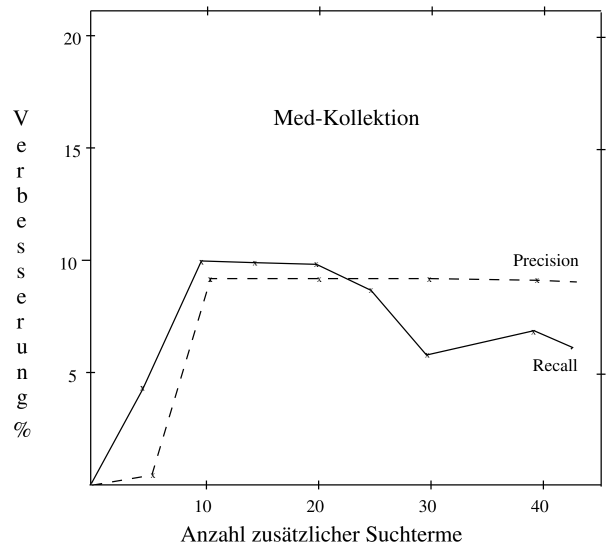

Abbildung 6.5. Systemleistung in der MED-Kollektion

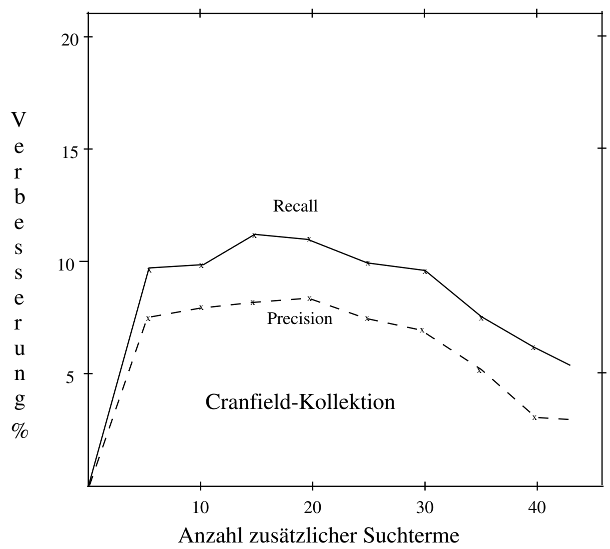

Abbildung 6.6. Systemleistung in der Cranfield-Kollektion

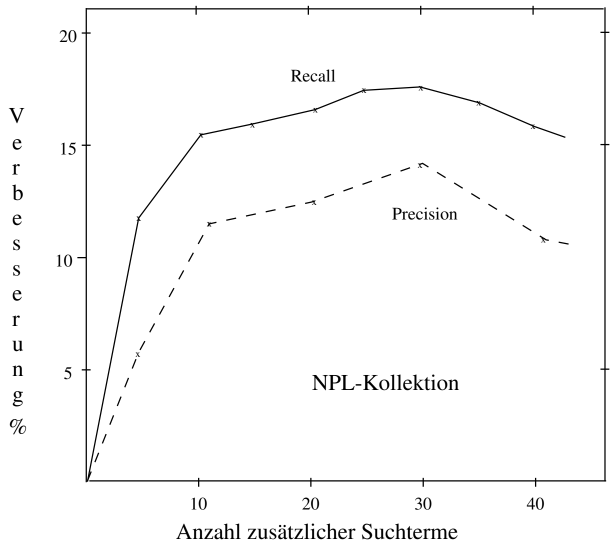

Abbildung 6.7. Systemleistung in der NPL-Kollektion

Die Messergebnisse zeigen Performance-Steigerungen in allen Test-Kollektionen. Optimal ist eine Anfrage-Erweiterung mit etwa 10 - 20 Termen, weitere Zugaben verschlechtern das Ergebnis wieder etwas.

In den Test-Kollektionen Cranfield und NPL sind die Ergebnisse besser als in der Med-Kollektion. Dies läßt die Vermutung zu, dass das System auf größeren Daten-Kollektionen leistungsfähiger ist, als auf kleinen.

6.7 Messergebnisse der Link-Assoziations-Matrix

Um die Ergebnisse der durchgeführten Effizienz-Messungen auch mit traditionellen Systemen vergleichbar zu machen, wird ein neues Messverfahren entwickelt. Für die Messungen an der Link-Matrix ist eine Erweiterung der Test-Kollektionen erforderlich. Weil ursprünglich in den dort enthaltenen Dokumenten keine Verweise zur Auswertung zur Verfügung stehen, müssen sie nachträglich eingesetzt werden. Der Konstruktion zusätzlicher Test-Verweis-Strukturen liegt hier die Annahme zugrunde, dass Verweise sinnverwandte Textpassagen miteinander verbinden. Verwendet wird nun die in den Test-Kollektionen enthaltene Relation, die jeder Anfrage eine Antwort in Form einer Menge von relevanten Dokumenten zuordnet. Es liegt nahe anzunehmen, dass die Dokumente einer solchen Antwort ebenfalls eine Sinnverwandtschaft aufweisen, und sie mit Verweisen zu verbinden. Um Leistungsunterschiede feststellen zu können, werden drei unterschiedliche Verweis-Strukturen in die Antwort-Dokumentmengen eingebracht:

In der ersten Verweis-Struktur sind die korrekten Antwort-Dokumentmengen von nur 10 Prozent der Anfragen mit Verweisen versehen. Die Anzahl der Verweise ist jeweils gleich der Anzahl der Dokumente in einer korrekten Antwort, d.h., die Dokumente sind in einer kettenartigen Topologie verknüpft.

In der zweiten Struktur sind die Dokumente, die zu den korrekten Antworten aller Anfragen gehören, in der gleichen Weise miteinander verbunden.

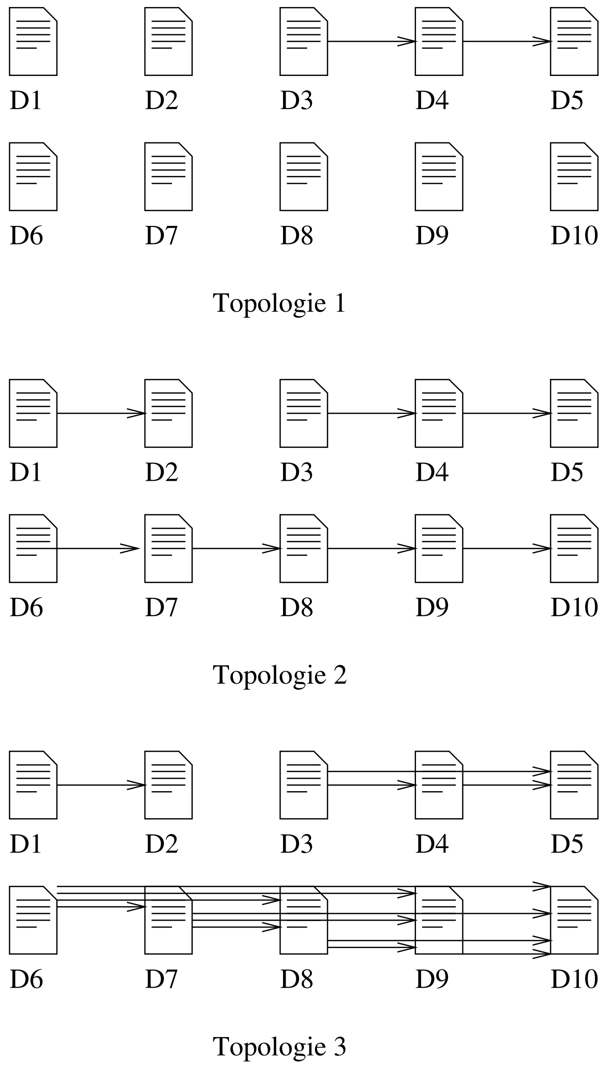

Abbildung 6.8 Verweis-Topologien

Die dritte Struktur erstellt alle Verbindungen zwischen den Dokumenten der korrekten Antworten, die möglich sind.

Abbildung 6.8 zeigt an einem Beispiel mit drei Anfragen und den zugehörigen korrekten Antworten, den Dokumentmengen {D1,D2}, {D3,D4,D5} und {D6,D7,D8, D9,D10}, wie die Verweis-Struktur konstruiert ist. Das Beispiel ist idealisiert, die Dokumentmengen müssen nicht paarweise verschieden sein, auch kann die Dokument-Sammlung Dokumente enthalten, die nicht Antwort einer Test-Anfrage sind.

Die Messergebnisse unter den verschiedenen Test-Kollektionen sind in den Abbildungen 6.9, 6.10 und 6.11 dargestellt.

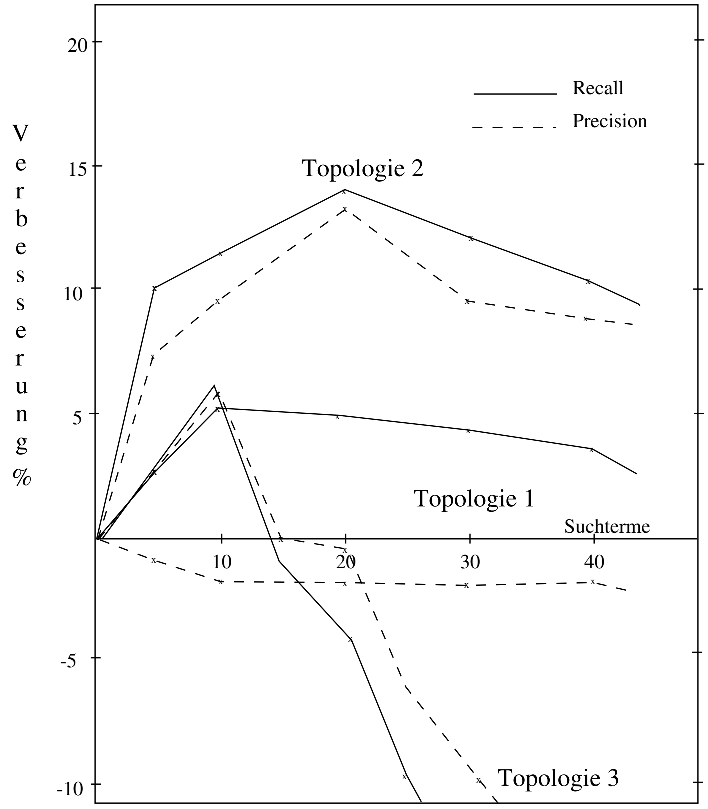

Abbildung 6.10 Messergebnisse an der MED-Kollektion

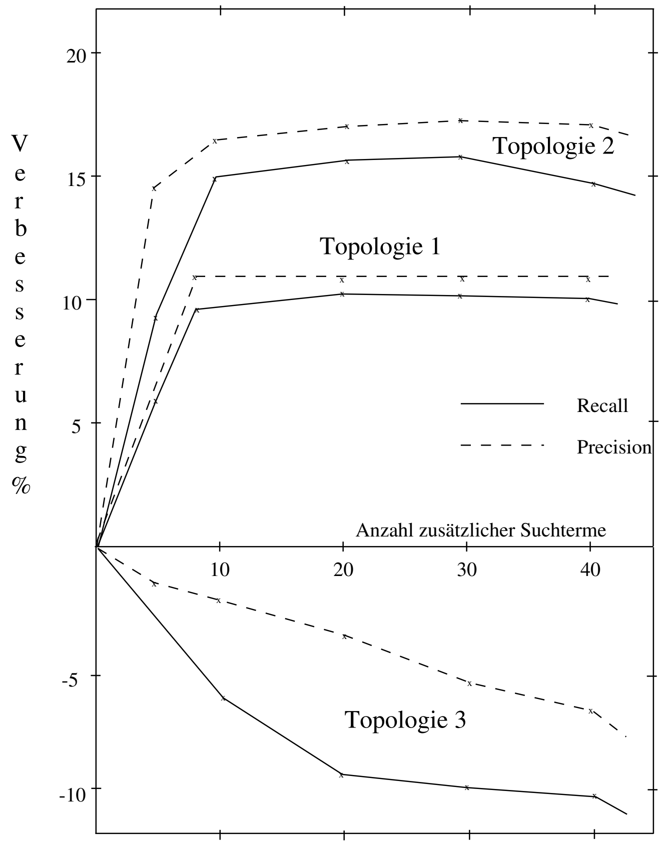

Abbildung 6.9 Messergebnisse an der Cranfield-Kollektion

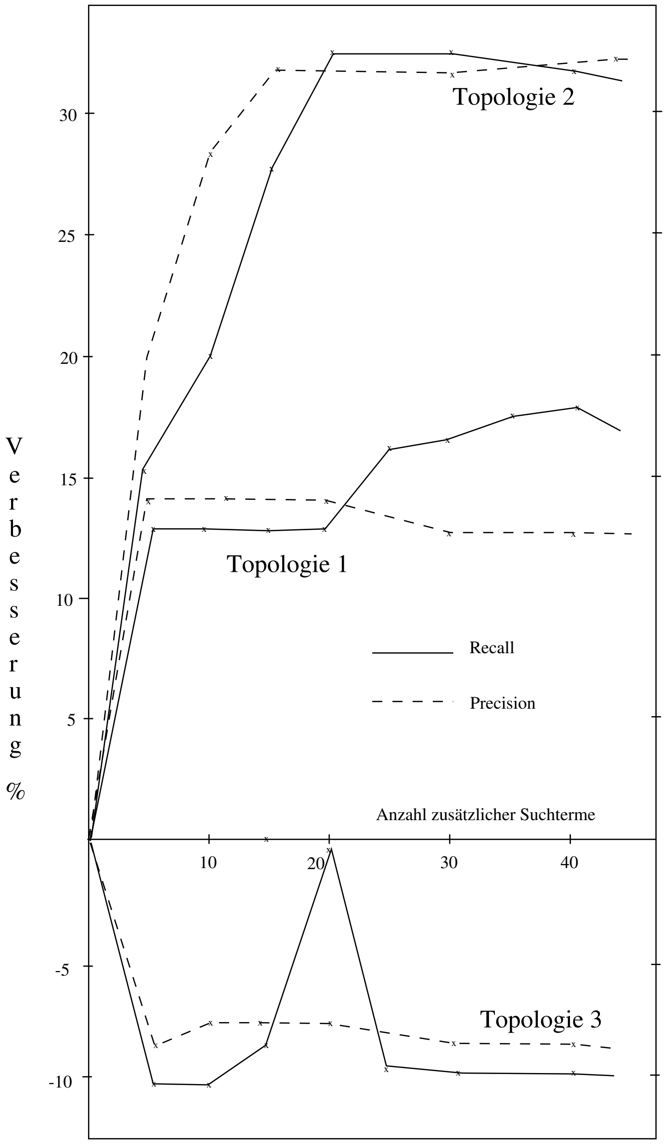

Abbildung 6.11 Messergebnisse an der NPL-Kollektion

Die Messergebnisse zeigen die beste Performance des Systems unter der Verweis-Topologie 2. Die höchsten Werte erzielt auch hier eine Erweiterung der Anfrage um etwa 10 -20 Terme.

Unter Topologie 2 sind die Ergebnisse schlechter, weil diese Struktur wesentlich weniger Information enthält als die erste, und die Trennung der relevanten Dokumente von den nicht relevanten daher nicht in dem Maße gelingt.

Die Verschlechterung der Gesamtleistung des Systems unter der Topologie 3 ist durch das massive Vermischen der Verweis-Informationen bei Auftreten eines Dokuments in mehreren Mengen von Dokumenten zu erklären, die für eine Anfrage relevant. Weiterhin bewirkt diese Struktur, dass sich die Lösungsmenge des Systems durch den starken Einfluß der Link-Struktur vom ursprünglichen Konzept der Anfrage sehr weit entfernt.

In der Praxis wird Topologie 3 kaum auftreten, weil Verknüpfungen sachfremder Themengebiete durch Verweise eher selten auftreten. Häufiger ist eine Mischung aus den Topologien 1 und 2 zu erwarten, die das System den vorliegenden Messungen zufolge zufriedenstellend verarbeiten kann.

Auch hier ist, wie im Falle der Term-Assoziationen zu beobachten, dass das System bei den größeren Test-Kollektionen bessere Werte liefert. Weil der Informationsgehalt der Assoziations-Matrizen höher ist, kann das System bei der Auswahl der Dokumente feiner unterscheiden.

6.8 Weitere Messungen

Die folgenden Messungen zeigen die Ergebnisse bei gemeinsamem Einsatz von Term- und Link-Thesaurus. Hier zeigt sich, dass eine Koppelung der Datenbasen nicht zu einer Addition der Retrieval-Ergebnisse führt. Der Term-Thesaurus verspricht eine gute Leistung bei relativ "flachen", d.h., mit einer geringen Anzahl von Verweisen versehen Dokument-Kollektion, während die Ergebnisse des Link-Thesaurus von den in einer vorhandenen Link-Struktur enthaltenen Informationen abhängen.

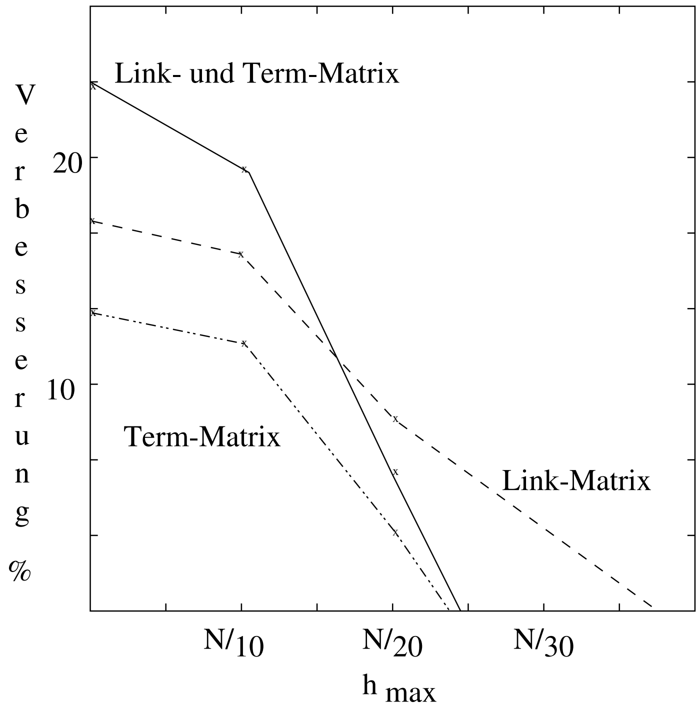

Abbildung 6.12 Unterschiedliche Dimensionen des Term-Raums

Die in den vorangegangenen Messungen beobachtete Verbesserung der Performance in umfangreicheren Dokument-Sammlungen soll nun durch weitere Versuche bestätigt werden. Die erste Messung überprüft die relative Leistungsänderung bei Einsatz der

Assoziations-Matrizen bei Änderung der Dimension des Term-Raums durch Modifikation der Schwelle h_{max}. Die zweite Messung zeigt relative Unterschiede bei Änderung der Anzahl ausgegebener Dokumente.

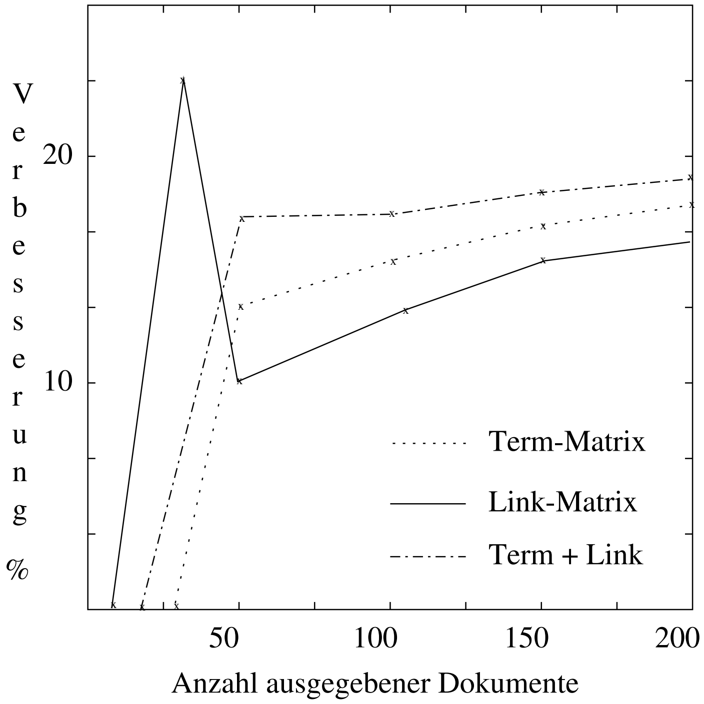

Abbildung 6.12 Ausgabe unterschiedlicher Dokument-Mengen

Hier ist deutlich zu sehen, dass mit zunehmender Anzahl von nachgewiesenen Dokumenten eine Leistungssteigerung einhergeht. Somit bestätigt sich auch hier die Beobachtung der Fähigkeit des Systems, große Datenmengen effektiver ordnen zu können als kleinere.

Kapitel 7

Zusammenfassung und Ausblick

Die vorliegende Arbeit beschreibt die Entwicklung und Implementation eines Systems zur Informationsgewinnung in einer WWW-Umgebung. Um eine hohe Anpassungsfähigkeit des Systems zu erhalten, waren leichte Benutzbarkeit und hoher Automatisierungsgrad Hauptziele bei System-Design und Auswahl der verwendeten Strategien zur Informationsextraktion- und Darstellung.

Durch das Lernen assoziativer Beziehungen zwischen Begriffen soll das System in die Lage versetzt werden, auch einen ungeübten Nutzer durch das Anbieten zusätzlicher, der Anfrage semantisch ähnlicher Begriffe, auf der Informationssuche zu unterstützen.

Im Bereich der Informationserschließung wurden aus der Information-Retrieval bekannte Methoden zur automatischen Konstruktion von Thesauri eingesetzt. Diese Methoden bedurften zusätzlicher neuer Entwicklungen, um sie in einer Hypertext-Umgebung anwenden zu können.

Im Anfrage-Bereich ist das System mit einer wahlweise automatischen oder manuellen Anfrage-Erweiterung ausgestattet und bietet auch hier dem Nutzer verbesserte Hilfestellungen durch eine neuentwickelte Einstellmöglichkeit für den Suchbereich an.

Die Messergebnisse zeigen eine starke Abhängigkeit der System-Performance von der Größe des einbezogenen Datenbestandes. Verwendet man sehr kleine Dokumentmengen, ist der in der Wissenbasis vorgehaltene Informationsgehalt zu gering, um Verbesserungen des Retrieval-Ergebnisses im Vergleich zu herkömmlichen Methoden erzielen zu können. Dies ändert sich jedoch bei dem hier auch vorgesehenen Einsatz des Systems in einem inhaltlich unterschiedlichen und datenmengenmäßig umfangreichen Umfeld. Hier sind Verbesserungen bis zu 25% gegenüber bisher verwendeten Verfahren messbar.

Das System zeigte sich bei der Feinabstimmung der unterschiedlichen Steuerungsparameter für Term-Reduktion, Ausgabe oder Gewichtungs-Strategien als relativ empfindlich und ist bei Einsatz passender Test-Kollektion für den Hypertext-Bereich zur Abstimmung in der Performance weiter steigerbar .

Die Verbesserungs- und Erweiterungsmöglichkeiten sind mannigfaltig. Man kann im Bereich der Gewichtungsverfahren für die Matrix-Gewichte weitere Leistungssteigerungen durch verstärkte Ausnutzung der Struktur von HTML-Dokumenten erzielen. Es wäre beispielsweise möglich, eigene Term-Matrizen für Terme in Überschriften oder Inhalts-Beschreibungen der Dokumente zu erstellen und diese in geeigneter Weise miteinander zu kombinieren.

Eine eingehendere Untersuchung der Link-Struktur des verwalteten Bereichs kann den dort enthaltenen Informationsgehalt tiefergehend erschließen. Beispielsweise wäre es möglich, die Abstimmung bei der Wertung der unterschiedlichen Thesauri für Term-Assoziation und Link-Assoziation für die Anfrage-Erweiterung genauer an das gegebene Umfeld anzupassen, als dies bisher in diesem System erfolgt. Weiter könnte man Mehrfach-Verweise zur Inhaltsextraktion verwenden, indem man die Dokumente in kleinere Teildokumente zerlegt. Dies hätte jedoch einen entsprechend gesteigerten Rechenaufwand zur Folge.

Erweiterungen sind ebenfalls in der Multimedia-Komponente des Systems möglich. Wenn geeignete Verfahren zur Inhalts-Extraktion von Graphiken, Bildern, Film- und Tonmaterial entwickelt sind, kann man diese Dateiformate problemlos für die Informationsgewinnung hinzuziehen.

Weil der Bedarf an HIR-Systemen mit dem Wachstum der im Internet-Bereich angebotenen Informationsmenge sicher steigen wird, ist eine Verbesserung der Messmethoden für die Systemleistung und die Entwicklung entsprechender Test-Kollektionen zu erwarten. Damit ist es möglich, die wichtigen Feinabstimmungen mit größerer Genauigkeit vorzunehmen.

Durch die im System vorhandene Möglichkeit, Anfragen in natürlicher Sprache zu verarbeiten, könnte man die Benutzungsfreundlichkeit weiter steigern, indem man eine entsprechende Schnittstelle einsetzt, die Tastatureingaben überflüssig macht.

Anhang A

Implementierung des HIR-Systems

Der Quellcode des Systems ist in C++ verfasst und setzt auf den effizienten Klassenbibliotheken des *LEDA research project* des Max-Planck-Institut für Informatik in Saarbrücken auf [Le99].

Zunächst gibt man dem System vor, welche Server es in den Datenbank-Bestand aufnehmen soll. Dies geschieht durch Editieren der Textdatei **SERVER_NAMES**, in der die Internet-Adressen dieser Server abgelegt sind. Die Installation des Systems erfolgt in zwei Phasen:

1. Erstellen einer Datenbank durch Abscannen der Server, Bearbeiten der erhaltenen Daten und Sicherung der Ergebnisse auf einer Festplatte.

2. Starten eines Nutzerprogramms zur Eingabe und Auswerten der Nutzeranfrage und zur Darstellung der Ergebnisse.

   Überwachen der Notwendigkeit von Update-Operationen, bei Bedarf Durchführung.

A.1 Die Installationsphase

Als Vorgabe benötigt das System eine oder mehrere Startseiten, die Ausgangspositionen zu den auf den Servern zu untersuchenden HTML-Dokumente liefern.

Der Roboter besucht zunächst die angegebene Startseite und läd deren Inhalt in das System. Die Verarbeitung der Startseite und der aus der Startseite extrahierten Seiten erfolgt in mehreren Schritten:

1. Funktion parse(string filename, list<string> doc,
   dictionary<string, int> h_files):
   Die HTML-Datei *<filename>* wird eingelesen, in Klartext und Markierungen zerlegt und zeilenweise in *list<string> doc* abgelegt. Der Name *filename* wird in *dictionary<string, int> h_files* abgelegt, um die erfolgte Bearbeitung der

Datei festzuhalten. Da im Netz der Verweise Kreise auftreten können, wird durch eine Abfrage von h_files eine nochmalige Bearbeitung vermieden.

2. Funktion remove_text_tags(list<string> text, list<string> signs, dictionary<string, int> n_signs):
   Sonderzeichen können im HTML-Standard durch *benannte Zeichen*, die in der ISO 8859-1 Norm festgelegt sind, ausgedrückt werden. Da man viele dieser Zeichen für die folgende Klartext-Analyse nicht benötigt, löscht *remove_text_tags* diese. Die deutschen Umlaute werden wieder In ASCII-Zeichen zurückübersetzt. Die benannten Zeichen sind auf der Festplatte in der Datei **N_SIGNS** abgelegt und werden nach *dictionary<string, int> n_signs* eingelesen. Ein weiterer Schritt liest nicht benötigte Zeichen wie z.B. Klammern, Punkte, Kommata oder Ausrufungszeichen aus der Datei **SIGNS** auf der Festplatte in *list<string> signs* ein und entfernt diese aus dem Text, der mit *list<string> text* übergeben wurde.

3. Funktionen clean(list<string> clean, string name) und filter_tags(string filter, string file):
   Die Funktion *clean* legt die einzelnen Wörter zeilenweise in der Reihenfolge ihres Auftretens im Originaldokument in die Liste clean ab. Es erfolgt die Auswertung der URLs und der Textüberschriften in *filter_tags*. Die ausgefilterten URLs stehen am Anfang der Liste clean, da deren Reihenfolge im Dokument nicht von Wichtigkeit ist. Die Überschriften sind ebenfalls an der Stelle in der Liste abgelegt, die dem tatsächlichen Auftreten im Originaldokument entspricht.

4. Funktion remove_stops(list<string> text, dictionary<string,int> stops):
   Die Funktion entfernt aus dem in der Liste *list<string> text* abgelegten Text Stoppwörter, die aus der auf der Festplatte liegenden Datei **STOPWORDS** in *dictionary<string,int> stops* eingelesen wurden.

5. Funktion all_terms(list<string> text, dictionary<string, ones> TF, string doc_id):
   Funktion *all_terms* legt den hochdimensionalen Wortvektor TF an, der zur Festlegung der Menge der Indexterme benötigt wird. In TF ist die Häufigkeit des Auftretens jedes Terms der Kollektion abgelegt.

   Weitere Informationen wie die maximale Anzahl des Auftretens des Terms in einem Dokument sowie die Anzahl der Dokumente, in denen der Term vorkommt, sind ebenfalls hier abgelegt. Diese Werte können zum Test anderer Gewichtungsverfahren verwendet werden.

6. Funktion store(dictionary<string,int> D, string html_file):
   Die aus der Bearbeitung entstandene Index-Datei wird auf der Festplatte abgelegt. Durch Ersetzung aller Vorkommen von ”/”durch ”_” in der URL ergibt sich der Name der Hilfsdatei.

7. Funktion good_terms(dictionary<string, ones> TF,
   dictionary<string,int> TF_good, int MAX_TERM_NUMBER):
   Wenn alle Seiten der Kollektion abgearbeitet sind, ermittelt die Funktion *good_terms* die *guten Terme* mit $1 < f(T) < n/10$ (n: Anzahl der Dokumente in der Sammlung). Terme, die nur einmal vorkommen, werden in der Datei **ONES**, Terme, die häufiger als n/10 auftreten, in der Datei **N10** auf der Festplatte gesichert.

Der resultierende hochdimensionale Wortvektor liegt in der Datei **VEKTOR**, die Namen der bearbeiteten HTML-Dokumente in der Datei **HELP_FILES** auf der Festplatte.

Nun kann man aus den Index-Dateien zwei Matrizen $SIMT$ und $SIML$ konstruieren, deren Einträge aus den Term-Term-Gewichten t_{ij} im einen Fall und den Term-Link-Gewichten l_{ij} im anderen Fall bestehen. Die Ablage der Gewichte erfolgt ohne Normierung, um ein vereinfachtes Update durchführen zu können. Zusätzlich werden die Normierungskoeffizienten gesammelt.

Das Hauptproblem bei der Berechnung der Matrizen liegt in Notwendigkeit, alle Terme miteinander verarbeiten zu müssen, weil auch alle Terme miteinander auftreten können. Bei einer Anzahl von 10^5 Index-Termen ergibt sich ein theoretischer Speicherplatzbedarf von $10^5 \cdot 10^5 \cdot 4$ Byte = 40 Gigabyte, bei 50000 Termen immerhin noch von 10 Gigabyte.

Weil der auf dem Heap vorhandene Speicherplatz nur in den seltensten Fällen ausreicht, um die gesammten Berechnungen vollständig im Speicher durchführen zu können, erfolgt die Ablage der Zwischenergebnisse auf der Festplatte. Die Berechnung der Matrix wird in zwei Schritten durchgeführt:

1. Während eines Durchlaufs durch die Index-Dateien erfolgt die Berechnung von Term-Term-Assoziationen oder im anderen Fall Term-Link-Assoziationen. Das System überprüft fortwährend die Belegung des Speicherplatzes auf dem Heap, um das Swappen der Daten durch das Betriebssystem zu verhindern. Die Verarbeitungszeiten verlängern sich in diesem Fall auf ein nicht akzeptables Maß. Die Konstante **MAXTERMS** regelt die maximale Anzahl von Einträgen auf dem Heap. Ist der gesammte Speicherplatz belegt, legt das Programm die bis dahin erstellte Matrix als temporäre Zwischendatei auf der Festplatte ab und

gibt den Speicher für neue Berechnungen frei. Dieser Vorgang setzt sich fort, bis alle Hilfsdateien abgearbeitet sind. Man erhält also auf der Festplatte eine Reihe von Matrizen, deren nicht berechnete Einträge gleich 0 gesetzt sind:

$$SIMT_{(1)} = \begin{array}{c} \\ t_1 \\ t_2 \\ \vdots \\ t_n \end{array} \begin{array}{c} \begin{array}{cccc} t_1 & t_2 & \cdots & t_n \end{array} \\ \begin{pmatrix} w_{11}^{(1)} & w_{12}^{(1)} & \cdots & w_{1n}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} & \cdots & w_{2n}^{(1)} \\ \vdots & \vdots & & \vdots \\ w_{n1}^{(1)} & w_{n2}^{(1)} & \cdots & w_{nn}^{(1)} \end{pmatrix} \end{array}$$

$$SIMT_{(2)} = \begin{array}{c} \\ t_1 \\ t_2 \\ \vdots \\ t_n \end{array} \begin{array}{c} \begin{array}{cccc} t_1 & t_2 & \cdots & t_n \end{array} \\ \begin{pmatrix} w_{11}^{(2)} & w_{12}^{(2)} & \cdots & w_{1n}^{(2)} \\ w_{21}^{(2)} & w_{22}^{(2)} & \cdots & w_{2n}^{(2)} \\ \vdots & \vdots & & \vdots \\ w_{n1}^{(2)} & w_{n2}^{(2)} & \cdots & w_{nn}^{(2)} \end{pmatrix} \end{array}$$

$$\vdots$$

$$SIMT_{(k)} = \begin{array}{c} \\ t_1 \\ t_2 \\ \vdots \\ t_n \end{array} \begin{array}{c} \begin{array}{cccc} t_1 & t_2 & \cdots & t_n \end{array} \\ \begin{pmatrix} w_{11}^{(k)} & w_{12}^{(k)} & \cdots & w_{1n}^{(k)} \\ w_{21}^{(k)} & w_{22}^{(k)} & \cdots & w_{2n}^{(k)} \\ \vdots & \vdots & & \vdots \\ w_{n1}^{(k)} & w_{n2}^{(k)} & \cdots & w_{nn}^{(k)} \end{pmatrix} \end{array}$$

2. Um alle Ergebnisse in einer einzigen Matrix zusammenzufassen, erfolgt die zeilenweise Verrechnung der Zwischendateien. Seien $w_{ij}^{(1)}, w_{ij}^{(2)}, \cdots, w_{ij}^{(k)}$ die Einträge der Zwischendateien $SIMT_1 \cdots SIMT_k$, dann erhält man die Einträge st_{ij} der entgültigen Matrix $SIMT$ durch

$$st_{ij} = \sum_{l=1}^{k} w_{ij}^{(l)} \tag{A.1}$$

für die Term-Term-Assoziationen. Die Berechnung der Term-Link-Assoziationen in der Matrix $SIML$ erfolgt analog.

Weil auch bei dieser Summenbildung eine Überlastung des vorhandenen Speicherplatzes eintreten kann, berechnet das Programm die mögliche Anzahl der Matrix-Zeilen, die gleichzeitig im Heap gehalten werden können, und richtet die Anzahl der Berechnungen daran aus.

Um schnelle Antwortzeiten des Systems zu erhalten und den Speicherplatzbedarf zu verringern, erfolgt die weitere Bearbeitung der Matrizen $SIMT$ und $SIML$ in zwei Schritten:

1. Beschränkung der Anzahl von Einträgen für jeden Term in der Matrix auf eine vorgegebene Zahl M, weil nur eine begrenzte Anzahl (typischerweise $M \in \{30, \cdots, 100\}$) von ähnlichen Termen zur Erweiterung der Anfrage benötigt wird.

2. Normierung der Termgewichte. Diese erfolgt erst jetzt, weil sich die Normierungskoeffizienten bei jedem Löschen oder Hinzufügen von Termen ändern. Bei einem Update ist dann nur noch die Neuberechnung des Vektors mit den Normierungsgewichten und die der verkürzten Matrix erforderlich.

Die Einträge der Matrix $SIMT$ haben die endgültige Form:

$$st_{ij} = \frac{\sum_{l=1}^{k} w_{ij}^{(l)}}{\sqrt{c(t_i)c(t_j)}} \qquad j \in \{1, \cdots, M\} \tag{A.2}$$

mit $c(t_i)$ und $c(t_j)$ als den in Kapitel 4.6 eingeführten Normierungsfaktoren.

Die Berechnung der entgültigen Einträge für Matrix $SIML$ erfolgt analog.

A.2 Das Anfrage-Modul

Zur Ausgabe der im System gespeicherten Daten muss eine Anfrage-Einheit installiert werden. Die genaue Architektur ist im Kapitel 5.3 beschrieben. Im Folgenden sollen nun die verwendeten Datenstrukturen der einzelnen Komponenten des Anfrage-Moduls beschrieben werden.

A.2.1 Die Startseite

In die durch den Nutzer aufzurufende Startseite ist ein HTML-Formular eingebettet, das ihm die Eingabe seiner Anfrage ermöglicht. Die Datenübertragung erfolgt durch Aufruf des CGI-Clients (Siehe Kapitel 3.6.3) mittels der **GET**-Methode. Abbildung A.1 zeigt die Startseite nach einer Anfrage:

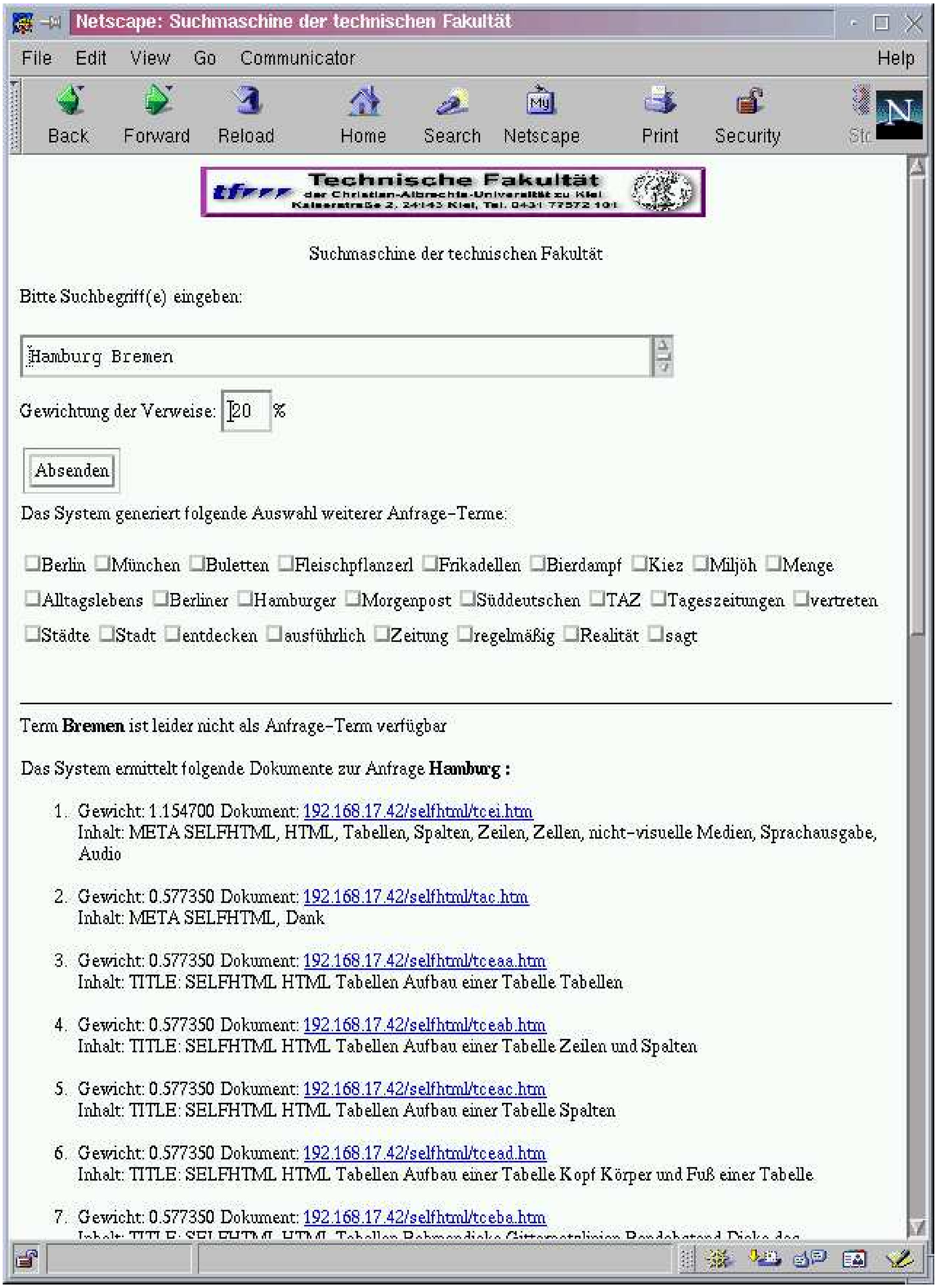

Abbildung A.1 Die Anfrage-Seite

Nachdem der Anfrage-Server die Anfrage bearbeitet hat, fügt er der Startseite 50 durch den Nutzer anwählbare Index-Terme hinzu, die den Assoziations-Matrizen entnommen sind. Daran anschließend folgt die Ausgabe einer nach Wichtigkeit für die Anfrage geordneten Liste von Dokumenten. Jedes Dokument ist mit einer Kurzbeschreibung versehen, der Nutzer kann sich dieses durch Anwählen einer Schaltfläche im Original betrachten.

Weitere Schaltflächen erlauben, das Gewicht der eingesetzten Assoziations-Matrizen zu verändern und damit einen unterschiedlichen Ausschnitt in der Wissensdatenbank bei der Suche zu wählen. Bei gering eingestelltem Gewicht sucht das System Dokumente, die dem Wortlaut der Anfrage möglichst ähnlich sind. Stellt man größere Gewichte ein, so wird ein weiterer Umkreis des Konzepts der Anfrage ermittelt.

A.2.2 Der CGI-Client

Der CGI-Client besteht aus einem CGI-Skript (Siehe Kapitel 3.6.3) zum Empfang des Anfrage-Formulars und einem Internet-Client zum Datenaustausch mit dem Anfage-Server. Er liest beim Start die vom HTTP-Server belegte Umgebungsvariable **QUERY_STRING** aus, in der die Daten des Anfrage-Formulars abgelegt sind. Danach errichtet er ein Byte-Stream-Socket und nimmt Kontakt zum Anfrage-Server über dessen Internet-Adresse auf. Kommt ein Kontakt zustande, schickt er die vorher erhaltenen Anfrage-Daten über die neue Socket-Verbindung zum Anfrage-Server und wartet auf dem gleichen Kanal auf eine Antwort. Bei Erhalt einer Antwort leitet der Client diese an den HTTP-Server weiter, der seinerseits den entfernten Browser damit bedient. Bleibt aus irgendwelchen systeminternen Gründen die Antwort aus, so erhält der Nutzer eine Fehlermeldung.

A.2.3 Der Anfrage-Server

Den Hauptteil des Anfrage-Moduls bildet der Anfrage-Server. Er ist als Internet-Server konzipiert und arbeitet auf der Internet-Adresse des Rechners, auf dem er installiert ist. Um weitere Abstimmungsarbeiten bei der Installation zu vermeiden, ermittelt der Server diese Adresse selbst und verwendet sie für den Kontakt mit dem CGI-Client. Beim Start liest er eine Reihe von benötigten Hilfs-Dateien ein:

1. den Vektor des Term-Vektorraums,

2. den Vektor des Dokument-Vektorraums,

3. die Assoziations-Matrix für die Terme,

4. die Assoziations-Matrix für die Links,

5. die Terme, die nur einmal im System auftreten und

6. die Terme deren Häufigkeit über der Schwelle h_{max} liegt.

Weil diese Dateien für ein Halten im Haupt-Speicher viel zu groß sind, ermittelt der Server nur die Startadressen der den einzelnen Termen zugeordneten Matrix-Zeilen auf der Festplatte und legt sie in einem Array aus Plattenadressen ab. Wird die einem Term zugehörige Matrix-Zeile benötigt, ermittelt das System über den Index des Terms die Startadresse aus dem Array und liest die Zeile ein.

Der Server öffnet nun einen Satz von Sockets und geht in eine Warteschleife, in der er diesen auf ankommende Nachrichten seitens der CGI-Clients überwacht. Sockets sind Kanäle für die Kommunikation zwischen Prozessen, die nicht nur lokal, sondern auch auf verschiedenen, durch ein Netzwerk verbundenen Rechnern ablaufen können. Es gibt verschiedene Arten der Datenübertragung durch Sockets, wie z.B über temporäre Dateien oder Internet-Verbindungen. Hier ist eine Internet-Verbindung gewählt, um den Anfrage-Client auch auf anderen Rechnern starten zu können, als dort, wo sich der Anfrage-Server befindet.

Eine dabei auftretende Schwierigkeit besteht in der Möglichkeit mehrerer gleichzeitig eintreffender Anfragen. Würde man die Sockets mit einer einfachen "read"- oder "recv"-Anweisung auslesen, wäre das Programm durch die zuerst eintreffende Anfrage geblockt, die anderen würden verlorengehen. Um dies zu vermeiden, muss die kompliziertere "select"-Anweisung verwendet werden. Durch die Möglichkeit mit "select" I/O-Multiplexing [St96] einzusetzen , kann man den gesamten Satz von Sockets überwachen und alle zusätzlich eingehenden Anfragen bedienen.

Wenn eine Verbindung zustande kommt, übernimmt der Server die Anfrage-Daten aus dem Stream und verarbeitet sie weiter. Üblicherweise sind von einem Internet-Browser lesbare Daten nach sogenannten *MIME-Typen*(**M**ultipurpose **I**nternet **M**ail **E**xtensions) codiert, und der Server muß diese zunächst in eine normale Textform dekodieren. Den Mime-Typ der Formular-Daten, die zwischen Browser und CGI-Skript ausgetauscht werden, bezeichnet man als "*x-www-form-urlencoded*".

Als Ergebnis der Entschlüsselung liegen nun die Anfrage-Terme und der Gewichtungsparameter für die Assoziations-Matrizen vor. Der Server bildet die in Kapitel 4.8 beschriebene Anfrage-Erweiterung, indem er die den einzelnen Termen zugehörigen Informationen aus beiden Assoziations-Matrizen unter Einsatz der Arrays mit Festplatten-Positionen von der Platte liest.

Der nächste Schritt fügt die erhaltenen Terme aus beiden Matrizen unter Berücksichtigung des durch den Nutzer vorgegebenen Gewichtungsparameters zu einer gesammten Anfrage-Erweiterung zusammen. Diese unterzieht das System dem im Kapitel 4.10 beschriebenen Verfahren zur Auswahl der relevanten Dokumente. Sind in der Anfrage Terme enthalten, die nur einmal in der Dokumenten-Kollektion auftreten, so werden die entsprechenden Dokumente gesondert in der Antwort nachgewiesen. Sollten in der Anfrage Terme auftauchen, deren Häufigkeit die Schwelle h_{max} überschreiten, erfolgt eine Aufforderung an den Nutzer, diesen Term seiner Anfrage weiter zu spezifizieren.

Das nun erhaltene Ergebnis -die Terme der Anfrage-Erweiterung und die ermittelten Dokumente- setzt das Anfrage-Modul in einen HTML-Text um und sendet ihn an den wartenden CGI-Client zurück. Sind alle Verbindungen einwandfrei zustandegekommen, erscheint auf dem Browser des anfragenden Nutzers die oben beschriebene Antwortseite des Retrieval-Systems. Er kann sich nun die Kurzinformationen der angebotenen Seiten ansehen und danach auswählen, oder seine Anfrage mit eigenen oder den angebotenen Termen spezifizieren und andere Dokumente nachweisen lassen.

Abbildungsverzeichnis

Anhang B

Literatur

[ANS92] ANSI/NISO (1992). *Standard Guidelines for Indexes, Draft 2.1, ANSI/NISO Z39.4-1992*. National Information Standard Organization (Z39).

[Ba94] Bartell, B.T.,Cottrell, G.E. and Belew, R.K.(1994). *Automatic Combination of multiple Ranked Retrieval Systems.* In W.B. Croft and C.J. van Rijsbergen(Eds.), Proc. 17 SIGIR Conference, Dublin, (Ireland), 172-181.

[BL94] Berners-Lee, T. et al (1994). *The Wourld Wide Web.* Communucations of the ACM, 37(8), 76-82.

[BC95] Berners-Lee, T. and Connolly, D. (1995). *Hypertext Markup Language - 2.0.* HTML Working Group, Internet-draft, 22. September 1995.

[Bo74] Bookstein,A. and Swanson, D.(1974). *Probabilistic models for automatic Indexing.* Journal of the ASIS, 25(5), 312-318.

[Ch93] Chen, H., Schatz,B. (1993). *Automatic thesaurus generation for FlyBase.* Proceedings of SIGIR 1993.

[Ch94] Chen, H. (1994). *Machine Learning for IR: Neural Networks, Symbolic Learning and Genetic Algorithms.* MIS Department, University of Arizona, (602) 621-4153.

[Ch95] Chen, H., Yim, T., Fye, D. and Schatz, B. (1995.) *Automatic thesaurus generation for an electronic community system.* Journal of the American Society for Information Science, 46(3):175-193.

[Cl66] Cleverdon, C.W., Mills, J. and Keen, M. (1966.) *Factors Determining the Performance of Indexing Systems.* Volume I-*Design.* ASLIB Cranfield Project, Cranfield.

[Co87] Conklin, J. (1987). *Hypertext: an Introduction and Survey.* IEEE Computer, 20(9), 17-41.

[Cr95] Croft, B. et al (1995). *IQUERY System Overview.* Computer Science Department, University of Massachusetts, 1995.

[Cr97] Croft, B. et al. (1997). *Providing Government Informations on the Internet: Experiences with THOMAS.* Libary of Congress, ITS 9332.

[Df82] M. Dillon and P. Federhart (1982). *The use of discriminate analysis to select content-bearing words.* Journal of the American Society for Information Science, pages 245–253.

[DF84] M. Dillon and P. Federhart (1984). *Statistical recognition of content terms in general text.* Journal of the American Society for Information Science, pages 3–10.

[Fr88] Frisse, M.E. (1988). *Searching for Information in a Hypertext Medical Handbook.* Communications of the ACM, 31(7), 880-886.

[GCY95] Gale, W., Church, K. and Yaranowsky, D. *Discrimination Decision for 100,000-Dimensional Spaces.* AT&T Bell Laboratories 1995 pg 16-18.

[Gr92] Grevenstette, G. (1992). *Use of syntactic context to produce term assotiation lists for retrieval.* In Proc. of the 15th International ACM SIGIR Conference on R & D in IR, pg 89-97. ACM Press, New York.

[Gr95] Gefenstette, G. (1995). *Automatic Thesaurus Generation from Raw Text using Knowledge-Poor Techniques.* Proceedings of SIGIR 1995.

[Ha92] Hancock-Bealieu, M (1992). *Query Expansion: Advances in Research in Online Catalogues.* Journal of Information Science, 18, 99-103.

[Ha91] Harman, D. (1991). *How Effective is Suffixing?* Journal of the American Society for Information Science, 42,7-15.

[Ha94] Harman, D., editor (1994). *The Second Text REtrieval Conference (TREC-2), Gaithersburg, MD 20899.* National Institute of Standards and Technology.

[Har86] Harter, S. (1986). *Online Information Retrieval.* Academic Press, INC.,Orlando Florida.

[Ho87] Hopefield J.J. (1987). *Collective Computation in neuronlike circuits.* Scientific American, 257(6):104-144, Dec 1987.

[Ja90] Jacobs, P.S. and Rau, L.F. (1990). *Extracting Information from On-line News.* ACM, 33(11), 88-97.

[Je96] Jensen, F. (1996). *An introduction to Bayesian Networks.* UCL Press Limited, London 1996.

[JG88] Jüttner,G. and Güntzer, U. (1988). *Methoden der Künstlichen Intelligenz für Information Retrieval.* K.G. Saur, 1988.

[Ka93] Kandzia, P., Klein, H.-J. (1993) *Theoretische Grundlagen relationaler Datenbanksysteme.* BI-Wissenschaftsverlag.

[Kai97] Kaiser,A. (1997). *Computer-unterstütztes Indexieren in intelligenten Information Retrieval Systemen.* http://aia.wu-wien.ac.at/Publikationen /Kaiser/diss.html.

[Kn83] Knorz, G. (1983). *Kooperatives (Referenz-) Retrieval - Eine Herausforderung für KI- und IR-Forschung.* Bericht, TH Darmstadt, Fachgebiet Datenverwaltungssysteme II.

[LC90] David D. Lewis and W. Bruce Croft (1990). *Term clustering of syntactic phrases.* In Proceedings of the ACM SIGIR, Brussels, Sept 1990.

[Le99] Leda Research Project, http://www.mpi-sb.mpg.de/LEDA, Algorithmic Solutions GmbH, Postfach 151101, 66041 Saarbrücken.

[Luh57] H.P. Luhn (1957). *A statistical approach to mechanized encoding and searching of libary automation.* IBM Journal of Research and Development, pages 309–317.

[Lu85] Lustig, G. (1985). *Zur Pilotanwendung AIR/PHYS der automatischen Indexierung.* in Endres-Niggemeyer,B. Krause,J., Eds. Srachverarbeitung in Information und Dokumentation, Springer Verlag 1985.

[Lu89] Lustig, G (1989). *Automatische Indexierung und Information Retrieval - Erfahrungen und Perspektiven.* Studien zur Klassifikation, pages 137–148.

[Ma92] Margulis, E.L. (1992). *N-Poisson document modelling.* In Proc. of the 15th International ACM SIGIR Conference on R & D in Information Retrieval, pages 160-169. ACM Press New York.

[Mau98] Maurer, H. et al. (1998). *From Databases to Hypermedia.* Springer-Verlag Berlin.

[Mi72] Minker,J., Wilson,G. and Zimmerman, B.(1972). *An evaluation of query expansion by the addition of clustered terms.* Information Storage & Retrieval, 8(6), 329-348.

[MK60] M.E. Maron and J.L. Kuhns (1960). *On relevance, probabilistic indexing and information retrieval.* Journal of the ACM, pages 216–244.

[Ml91] Mauldin, M.L. (1991). *Conceptual Information Retrieval: A Case Study in Adaptive Partial Parsing.* Kluwer Academic Publishers.

[MN98] Münz, S. and Nefzger, W. (1998). *HTML 4.0 Handbuch.* Franzis-Verlag 1998.

[Ni90] Nielsen, J. (1990). *Hypertext and Hypermedia.* Academic Press, Boston.

[Pan87] Panyr, J. (1987) *Interaktive Retrievalstrategien: Relevanzfeedback-Ansätze.* Nachrichten für Dokumentation, pages 145–152.

[Qiu94] Qiu, Y., Frei, H.P.(1994). *Improving the Retrieval Effevtivness by a Similartiy Thesaurus.* Technical Report 225, ETH Zurich, Department of Computer Science.

[RI98] Ristad, E.S., Yianilos, P. (1998) *Learning String-Edit Distance* IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 20 No. 5.

[Rij77] van Rijsbergen, C.J. (1977). *J.Doc.* **33,** 106.

[Rij79] van Rijsbergen, C.J. (1979). *Information Retrieval.* London: Butterworths, 1979.

[Ru92] Ruge, G. (1992). *Experiments on Liguistically-based term associations.* Information Processing & Management, 28(3), 317-332.

[Sal75] Salton, G., Yang, C.S., Wong,A.(1975). *Commun. Assoc. Compute. Mach.* **18**, 613.

[Sal80] Salton, G. (1980). *Automatic term class construction using relevance - a summary of work in automatic pseudoclassifcation.* Information Processing & Management, 16(1), 1-15.

[Sal83] Salton,G and McGill, M.J. (1983). *Introduktion to Modern Information Retrieval.* McGraw-Hill, New York.

[Sal87] Salton, G. (1987). *Information Retrieval - Grundlegendes für Informationswissenschaftler.* McGrawHill.

[Sal90] Salton, G. and Buckley, C.(1990). *Improving Retrieval Performance by Relevance Feedback.* Journal of the American Society for Information Science, 41, 288-297.

[Sal91] Salton, G. (1991). *Developments in automatic text retrieval.* Science, 253:974-979.

[Sch86] Schwarz, G. and Thurmair, C. (1986). *Informationslinguistische Texterschließung.* Olms Verlag, 1986.

[SG86] SGML (1986). *ISO Standard Generalized Markup Language.* ISO 8879: 1986

[SJ71] K. Sparck Jones. (1971). *Automatic keyword classification for information retrieval.* Buttersworth, London.

[Sm80] Smith, L.C. (1980). *Artificial Intelligence Applications in Information Systems.* Williams, M.E. (Ed.) Annual Review of Information Science and Technology, Vol 15. Knowledge Industrie.

[SR83] Smeaton, A. F. and van Rijsbergen, C. J. (1983). *The retrieval effects of query expansion on a feedback document retrieval system.* The Computer Journal, 26(3), 239-246.

[Sr90] Srinivasan, P. (1990). *A comparison of two-poisson, inverse document frequency and discrimination value models of document representation.* Information Processing and Management, 26(2), 269-278.

[St96] Stevens, R. (1996). *Advanced Programming in the UNIX Environment.* Addison-Wesley Professional Computing Series. 394-400.

[SY75] Salton, G., Yang, C.S. and Yu, C.T. (1975). *A theory of term importance in automatic text analysis.* Journal of the ASIS, 26(1), 33-44.

[Th93] Thompson,P. (1993). *Description of the PRC CEO algorithm for TREC.* In Donna K. Harman, editor, The First Text REtrieval Conference (TREC-1), pages 337-342, March 1993. NIST Special Publication 500-207.

[Tu91] Turtle, H.R. and Croft, W.B. (1991). *Evaluation of an Inference Network-Based Retrieval Model.* ACM Transactions on Information Systems, 9(3), 187-222.

[Tu92] Turtle, H.R. and Croft, W.B. (1992). *A Comparison of Text Retrieval Models.* The Computer Journal, 35(3), 279-290.

[WH89] Womser-Hacker, C. (1989). *Der PADOK-Retrievaltest.* Olms, 1989.

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst und ausschließlich die angegebenen Hilfsmittel und Quellen verwendet habe.

Kiel,den