

Weiterentwicklung von Werkzeugen zur Wissensauffindung im World-Wide-Web

Ergänzungen des Harvest-Systems im Hinblick auf Fehlertoleranz, Konfigurierbarkeit,
Keyword-Relevanz und Ranking

Diplomarbeit

an der

Technischen Universität Graz

vorgelegt von

Dietmar Neussl

Institut für Informationsverarbeitung und Computergestützte neue Medien (IICM),
Technische Universität Graz
A-8010 Graz

September 1998

© Copyright 1998, Dietmar Neussl

Betreuer: Dipl.-Ing. Christian Gütl

Begutachter: o.Univ.-Prof. Dr. Dr.h.c. Hermann Maurer

Kurzfassung

Das Internet stellt heute, wenige Jahre nach Einführung des World Wide Web, die weltweit größte Wissens- und Informationsdatenbank dar. Das Auffinden von relevanter, seriöser und qualitativ adäquater Information wird aufgrund seiner Unstrukturiertheit, seiner Dynamik und nicht zuletzt seiner Anonymität erheblich erschwert. Aufgabe der Suchdienste ist es, die derzeit schätzungsweise 300 Millionen Web-Seiten zu durchkämmen und dem Suchenden Verweise zu den gewünschten Themen bereitzustellen.

In der vorliegenden Arbeit wird, nach einer einführenden Begriffsbestimmung und Einteilung gebräuchlicher Suchdienste, das Harvest-System als ein Vertreter des Konzepts der verteilten Suche beschrieben. Untersucht werden die HTML-Konvertierung und der Suchindex. Dabei liegt das Hauptaugenmerk auf den Bereichen der automatischen Generierung von Schlüsselwörtern, der Keyword-Relevanz-Filterung sowie der Gewichtung von Suchresultaten. Die Ergebnisse dieser Untersuchung führen in weiterer Folge zu Modifikationen der entsprechenden Module des Harvest-Systems. Diese Änderungen werden, ebenso wie die dadurch erzielten Verbesserungen und neu entstandenen Möglichkeiten, detailliert diskutiert und dokumentiert.

Im Ausblick werden schließlich weiterführende Ergänzungen und Einsatzmöglichkeiten des neuen Systems erörtert. Die Verwendung als Hintergrundbibliothek für die WBT-Umgebung von Hyperwave (GENTLE) und als Basis eines *Information Reuse* Systems wird ebenso angesprochen, wie eine mögliche *Agent*-Anbindung zur Erreichung von plattformunabhängiger Zusammenarbeit.

Abstract

Only a few years after the introduction of the World Wide Web, the internet represents today's largest world wide knowledge- and database. The discovery of relevant, serious and adequate information is complicated enormously by its dynamics, lack of structure and last but not least, anonymity. The purposes of internet search facilities are to gather the approximately 300 million web pages, and to provide with links to specific areas of knowledge for the user.

In this thesis common search methods are explained. The Harvest-System, which represents the concept of distributed search, is described. Following this, the HTML conversion and search indices are investigated in more detail. Special attention is given to the fields of automatic generation of keywords, the relevance of keywords and the ranking of retrieved information. The results of this investigation lead to modifications in corresponding modules in the Harvest system. These changes, the ensuing improvements and new possibilities are documented in detail.

Finally, further extensions and possible applications are discussed. The system may be use as an background library for Hyperwave's WBT-environment (GENTLE) as well as a basis of an *Information Reuse* system. The implementation of *software agents* for platform independent co-operation is also mentioned.

Ich versichere hiermit, diese Arbeit selbständig verfaßt, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich auch sonst keiner unerlaubten Hilfsmittel bedient zu haben.

Inhaltsverzeichnis

1	Einleitung	1
I	Untersuchungsbereich	4
2	Suchdienste	5
2.1	Grundlegende Verfahren	5
2.1.1	Sammeln und Aufbereiten der Information	5
2.1.2	Bereitstellung der Information	6
2.2	Einteilung von gängigen Suchdiensten	8
2.3	Schwachpunkte gängiger Suchdienste	10
2.3.1	Netzwerk- und Serverbelastung	10
2.3.2	Vollständigkeit, Aktualität und Linkkonsistenz	10
2.3.3	Qualität und Zuverlässigkeit	11
2.4	Suche in Hyperwave	12
2.5	Zusammenfassung	13
3	Das Harvest-Suchsystem	14
3.1	Das Harvest-Konzept	14
3.2	Der Gatherer	18
3.2.1	Das Essence-Subsystem	19
3.2.2	Die Linkverfolgung	20
3.3	Der Broker	20
3.3.1	Das Indexinterface	22
3.3.2	Das Suchinterface	23
3.4	Vorteile der verteilten Suche	24
3.4.1	Netzwerk- und Serverbelastung	24
3.4.2	Vollständigkeit, Aktualität und Linkkonsistenz	24
3.4.3	Gliederung des Informationsraumes	25
3.4.4	Qualität und Zuverlässigkeit	25

3.4.5	Zusatzinformation über Server und Sites	25
3.5	Zusammenfassung	26
4	Verbesserungspotential am Harvest-System	27
4.1	Die HTML-SOIF-Konvertierung	27
4.1.1	Der HTML-Parser	27
4.1.2	Modifikation der SOIF-Attribute	29
4.2	Der Suchindex	31
4.2.1	Keyword-Relevanz-Filter	32
4.2.2	Gewichtung der Suchergebnisse - Ähnlichkeit von Dokument und Suchanfrage	32
4.2.3	Zusatzinformation über Server und Sites	36
4.2.4	Benutzerschnittstelle	37
4.3	Zusammenfassung	37
II	Gestaltungsbereich	39
5	Der HTML-nach-SOIF-Konverter	41
5.1	Zielsetzung und Übersicht	41
5.1.1	Konfigurierbarkeit	42
5.1.2	Autom. Zusammenfassung, Spracherkennung und Metadaten . .	44
5.1.3	Fehlertoleranz	45
5.1.4	Konvertierung nach ISO8859-1	45
5.2	Die Konfigurationsdatei	46
5.2.1	Die HTML-SOIF-Tabelle	46
5.2.2	Module und Filter	49
5.2.3	Separatoren	50
5.3	Filtermodule	51
5.3.1	Allgemeine Nachbearbeitung	52
5.3.2	Automatische Zusammenfassung	53
5.4	Installation	54
5.5	Zusammenfassung	56
6	Der Suchindex	57
6.1	Motivation und Zielsetzung	57
6.1.1	Keyword-Relevanz	57
6.2	Datenbankdesign	58
6.2.1	Theorie relationaler Datenbanken	58
6.2.2	Die Relationen des Suchindex	59

6.3	Indizierung	63
6.3.1	Einfügen	64
6.3.2	Löschen	66
6.4	Verwalten von Servern und Web-Areas	66
6.5	Suchabfragen	67
6.5.1	Prinzip der Suchabfrage	67
6.5.2	Sucheanfrage mit einem Wort	68
6.5.3	Boolsche Operatoren	69
6.5.4	Eingeschränkte Suche	70
6.5.5	Gewichtung gefundener Dokumente	71
6.6	Mehrstufige Suche	72
6.7	Einbindung in den Harvest-Broker	73
6.7.1	Die Indizierung	73
6.7.2	Die Suchabfragen	74
6.8	Implementierung	76
6.9	Zusammenfassung	76
7	Zusammenfassung und Ausblick	77
	Abbildungsverzeichnis	80
	Tabellenverzeichnis	81
	Größenverzeichnis	81
	Literaturverzeichnis	82

Kapitel 1

Einleitung

Das Niederschreiben und Aufnehmen von Wissen in Büchern, das Verwalten, Sammeln und Festhalten von Information in Archiven verliert mehr und mehr an Bedeutung. Der Prozeß des Durchsuchen übersichtlicher Ordnungssysteme wie Bibilotheken, Bücher und Zeitschriften wandelt sich in eine Auseinandersetzung mit komplexen Informationsfragmenten. Digitale Kommunikations- und Informationssysteme und globale Vernetzung bewirken eine Entfernung von existierenden Informationsstrukturen. Das Internet spannt einen Raum auf, der sich zunehmend mit dem realen gesellschaftlichen Lebensraum überschneidet. Die uneingeschränkte Informationsflut verlangt neue Mittel zur Suche und Auffindung von Inhalten.[Buc96]

In den USA wurden bereits in den 60er Jahren diverse Forschungsinstitute mit Hilfe digitaler Netze verbunden. Im Hintergrund des kalten Krieges wurde ein militärisches, dezentralisiertes Kommunikationsnetz, das ARPANET¹ eingeführt, das den Informationsaustausch unter den Stützpunkten ermöglichte. Neben diesem entstand das CS-NET², das Forschungseinrichtungen und Industriezentren vernetzte. Daraus entwickelte sich schließlich ein Netzwerk, das sämtliche Universitäten, Forschungs- und Entwicklungseinrichtungen miteinander verband. Die allgemeine Nutzung des Internet wurde erst in den Jahren um 1983 mit der Integration des Internet-Protokolls TCP/IP³ in die Berkley UNIX-Distribution eingeleitet. [HaE96]

1989 begann Tim Berners-Lee bei CERN⁴ in Genf mit der Entwicklung eines Informationssystems auf Hypertext- und Multimediabasis für das Internet. Er machte sich dabei das Internet als Kommunikationsmedium selbst zu Nutze, indem er seine Ideen, Entwürfe und lauffähige Versionen des Systems dem interessierten Publikum über diesen Weg zur Verfügung stellte. Er gewann dadurch Mitarbeiter und neue Ideen, weckte Interesse und schuf so gleichzeitig einen Markt für sein Produkt [BCL⁺94]. Das WWW nimmt seit der Verbreitung der ersten grafikfähigen Web-Browser zunehmend mehr Raum ein und dominiert das Netz mittlerweile in jeder Hinsicht [Net96].

Das Internet stellt heute, wenige Jahre nach Einführung des World Wide Web, die weltweit größte Wissens- und Informationsdatenbank dar. Zur Zeit spannen et-

¹Advanced **R**esearch **P**rojects **A**gency

²Computer **S**cience **N**etwork

³Transmission **C**ontrol **P**rotocol / Internet **P**rotocol

⁴Conseil **E**uropéenne pour la **R**echerche **N**ucléaire (dt. Europ. Organisation für Kernforschung)

wa 300 Million Dokumente und 300 Tausend Web-Server diesen Informationsraum auf [Sul98a]. Ein ungeordnetes und unübersichtliches Wissens- und Informationsangebot steht der ständig wachsenden Anzahl von Benutzern gegenüber. Jeusfeld und Jarke [JJ97] sprechen von einem großen, dynamischen und unstrukturierten Informationsmarkt. Die Dynamik äußert sich durch kontinuierlichen Wandel der Dokumente und ihrer Inhalte. Durchschnittlich wird ein Dokument alle 75 Tage geändert [Bra97]. Das *Georgia Institute of Technology* in Atlanta stellte in seiner jüngsten Studie [Gra97] fest, daß das Sammeln von Information die wichtigste Tätigkeit im Netz ist. An zweiter Stelle steht das Suchen von Information.

Gegenwärtige Suchdienste kommen mit den grundlegenden Eigenschaften dieses globalen Mediums kaum mehr zu Rande. Die enorme Netzwerklast von Suchdiensten verringert die Leistung für andere Dienste im Netz. Darüberhinaus werden Informationsserver durch permanentes “Gathern”⁵ in ihrer Leistungsfähigkeit geschwächt. Durch Einführung von Richtlinien, die das Ausschließen von Suchdiensten behandeln, versucht man dieses Problem in den Griff zu bekommen.⁶ [Bek96] Auch für den Benutzer wird es mit steigendem Angebot immer schwieriger, relevante und adäquate Information zu finden. Aspekte der Zuverlässigkeit und Qualität spielen eine immer größer werdende Rolle.

Die Quantität und die Unstrukturiertheit des Informationsangebotes verlangt neue Weg in der Auffindung und Bereitstellung von Information. Verteilte Suchsysteme passen sich der Topologie des Internets an. Durch die bekannte Methode “Teil-und-Herrsche” kann man lokal verschiedene Grundprobleme lösen und unter Einbindung vieler dieser Teilsysteme ein komplexes, strukturiertes Netz aufbauen, das in thematische oder regional gegliederte Suchdienste mündet. Zusammenarbeit statt gegenseitiges Blockieren. Das Harvest-System [BDH⁺94] der Universität von Colorado ist ein Vertreter dieses Konzeptes. Es ist eine Sammlung von Werkzeugen zur Auffindung und Bereitstellung von Information, mit denen flexible, verteilte Suchdienste aufgebaut werden können. Es unterstützt verschiedene Protokolle und Dokumenttypen.

In dieser Arbeit wird, nach einer einführenden Begriffsbestimmung und Einteilung gebräuchlicher Suchdienste, dieses System vorgestellt und bezüglich einiger spezieller Punkte untersucht. Das Hauptaugenmerk wird dabei auf folgende Bereiche gelegt:

- Die Fehlertoleranz und Konfigurierbarkeit des HTML-Parsers
- Automatische Generierung von Schlüsselwörtern und Zusammenfassungen
- Integration von Metadaten in die Dokumentenbeschreibung
- Schnittstellen für die Spracherkennung
- Berücksichtigung von Keyword-Relevanz bei der Suche im Index
- Gewichtung der einzelnen Dokumente im Suchergebnis in Abhängigkeit von der jeweiligen Bedeutung der Suchwörter im Dokument

⁵Abernten bzw. Durchsuchen

⁶Der Robot Exclusion Standard wurde 1994 ins Leben gerufen und wird von den meisten Web-Servern unterstützt. Er liegt zur Zeit nicht als RFC-Standard vor.

Die Ergebnisse dieser Untersuchung führen in weiterer Folge - im Gestaltungsbereich - zu Modifikationen der entsprechenden Module des Harvest-Systems:

- Ein fehlertoleranter, konfigurierbarer HTML-Konverter ersetzt den *Summarizer* des Harvest-Systems.
- Der neue Konverter besitzt Schnittstellen für Spracherkennung und der Erzeugung automatischer Zusammenfassungen.
- Metadaten können extrahiert und anschließend mit anderen Dokumentattributen verknüpft werden.
- Ein neuer Index auf Basis einer relationalen Datenbank bietet unter anderem die Möglichkeit der Relevanzberechnung von Schlüsselwörtern. Die Gewichtung der indizierten Wörter erlaubt es weiters, Suchabfragen mittels Ähnlichkeitsverfahren⁷ zu formulieren und die Ergebnisse zu bewerten.
- Zusätzlich zu den Dokumenten werden Informationen zum entsprechenden Web-Bereich indiziert. Es ist somit eine Suche in dieser Zusatzinformation möglich. Die Dokumente werden den weiteren Web-Bereichen (Informationsserver, Web-Area) zugeordnet. Dadurch läßt sich eine zweistufige Suche realisieren.

Im Ausblick werden schließlich weiterführende Ergänzungen und Einsatzmöglichkeiten des neuen Systems erörtert. So lassen sich zum Beispiel Spracherkennungsmodule in den HTML-Konverter integrieren. Die hyperrelationale Suche, die Suche nach Termen über die Dokumentgrenzen hinweg, wäre eine *Retrieval*-Methode, die in einer weiteren Ausbaustufe des Suchindexes realisiert werden könnte.

Die beschriebenen Eigenschaften des neuen Systems erlauben dessen Verwendung als Hintergrundbibliothek für die *Web-Based-Training*-Umgebung von Hyperwave (GENTLE) und als Basis von *Information Reuse* Systemen. Um plattformunabhängige Zusammenarbeit zwischen Teilsystemen zu erreichen, bieten sich *Agent*-Anbindungen an. Ideen zu diesen Themen bilden den Abschluß der vorliegenden Arbeit.

⁷siehe Abschnitt 4.2.2

Teil I

Untersuchungsbereich

Kapitel 2

Suchdienste

Nachdem in der Einleitung deutlich gemacht wurde, welche Anforderungen und Probleme eine Wissens- und Informationsdatenbank, wie sie das Internet heute darstellt, bezüglich der Suche von spezieller Information mit sich bringt, werden nachfolgend derzeit gängige Ansätze, Verfahren bzw. Systeme zur Wissensauffindung beschrieben.

2.1 Grundlegende Verfahren

2.1.1 Sammeln und Aufbereiten der Information

Die im folgenden angeführten Möglichkeiten zur Lokalisierung von Dokumenten sind nicht klar abgrenzbar. Kombinationen sind vielfach vorhanden. Eine Sonderstellung nehmen Informationssysteme ein, die ein integriertes Suchsystem beinhalten. [Koc96] So passiert die Aktualisierung des Suchsystems beim Hyperwave-Server (siehe Abschnitt 2.4) automatisch mit der Veränderung des Informationsangebotes.

Vollautomatische Auffindung

Hierbei navigiert ein Programm (*Robots*, *Bots* bzw. *Spiders*), wie der Benutzer mit Hilfe von Hyperlinks beim WWW, Menüs bei Gopher oder der Verzeichnisstruktur bei FTP, im Informationsraum. Die Information der Dokumente wird automatisch verarbeitet. Der Umfang kann dabei von einer Kurzbeschreibung (Autor, Titel, Schlüsselwörter, Erstelldatum etc.) bis zur Volltext-Information reichen. Zusätzlich können Wörter aufgrund ihrer Formatierung im Text als Schlüsselwörter eingestuft werden. Bei HTML-Dokumenten wird weiters die Metainformation aus den Meta-Tags gewonnen. Die so gesammelte Information dient als Basis einer Informationsstruktur (meist eine Datenbank) und wird, nach eventueller Nachbearbeitung (Stoppliste, *Stemming*, Kleinschreibung, Bedeutungssubstitution)¹ indiziert. [Koc96]

¹Auf diese Verfahren wird im Abschnitt 2.1.2 näher eingegangen

Anmelden von Informationseinheiten

Hier meldet der Autor seine Dokumente bei einem Suchdienst an. Dies geschieht unter Verwendung eines Formulars, das auch der Eingabe von Zusatzinformation dient. Diese angemeldeten Dokumenten können nun je nach Art des Suchdienstes als

- Startpunkt der automatischen Absuche (Linkverfolgung), oder als
- Eintrag in einem Suchkatalog

verwendet werden. [Koc96]

Redaktionelle Recherche

Die redaktionelle Recherche basiert üblicherweise auf Suchergebnissen einer vollautomatischen Auffindung. Redaktionsteams können auch von sich aus Hypertext-Strukturen zu bestimmten Themenkreisen verfolgen und passende Informationseinheiten aufnehmen. Darüberhinaus werden auch vorangemeldete Dokumente gesichtet, kategorisiert und aufgenommen. Mit dieser Methode lassen sich auch Bewertungen der Server- und Dokumenteninhalte realisieren. Die Beurteilungen können die Art, die Qualität, die Aktualität und die Leserzielgruppe der Dokumente beinhalten. [Koc96]

2.1.2 Bereitstellung der Information

Wie sooft entscheidet schließlich die Schnittstelle zum Anwender über die Brauchbarkeit eines Systems. Die zwei wesentlichen Kategorien werden nun kurz beschrieben [Koc96] [GAM98].

Suchkataloge

WWW-Kataloge werden von Systemen mit aktiver Anmeldung unterhalten. Sie bieten eine hierarchisch nach Sachgebieten gegliederte Suchstruktur, die das Navigieren innerhalb des Kataloges ermöglicht. Stichwortsuche ist ebenfalls oft integriert. [Koc96]

Suchindex

Ein Suchindex hält in einer Datenbank die ausgewerteten Informationen der aufgefundenen Dokumente bereit. Der Umfang variiert stark und reicht von Dokumentenname (WAIS), kurzer Beschreibung oder Keywordliste bis hin zum Volltext. Der Benutzer des Suchdienstes kann durch Eingabe von Suchbegriffen nach den gewünschten Dokumenten suchen. Suchsysteme mit Suchindex unterscheiden sich nun hinsichtlich der Detailliertheit der Indizierung, der Anfrageformate und der Darstellung der Ergebnisse. Folgende Kriterien sind hierbei zu beachten [WMB94] [FB92]:

boolsche Anfragelogik: Sie ist in den meisten Suchindexsystemen integriert, ist aber aus der Sicht des Benutzers nicht ideal, da sie Grundkenntnisse der Booleschen Algebra erfordert.

Quorum-Level: Dieser Ansatz kommt zunächst ohne boolescher Logik aus. Der Benutzer gibt lediglich die Suchbegriffe ein. Das *Retrieval*-System erzeugt aus diesen eine Menge von booleschen Anfragen, die von “sehr eng” (ausschließlich UND-Verknüpfungen) bis “sehr weit” (ausschließlich ODER-Verknüpfungen) reichen (siehe Abbildung 2.1). Diese Anfragen werden an den Suchindex gesandt. Die Ergebnisse werden entsprechend der zugrundeliegenden Anfrage gereiht und ausgegeben. Je mehr Konjunktionen die Disjunktionsterme enthalten, desto gewichtiger sind ihre Treffer im Gesamtsuchergebnis.

Gewicht	Query
4	$(A \wedge B \wedge C \wedge D)$
3	$(A \wedge B \wedge C) \vee (A \wedge B \wedge D) \vee (A \wedge C \wedge D) \vee (B \wedge C \wedge D)$
2	$(A \wedge B) \vee (A \wedge C) \vee (A \wedge D) \vee (B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$
1	$(A \vee B \vee C \vee D)$

Abbildung 2.1: Quorum-Level: Beispiel einer Query-Hierarchie

Der Benutzer braucht sich so nicht um die Formulierung kümmern. Da das Suchergebnis in der Regel hier besonders umfangreich sein wird, ist eine Schranke zur Reduktion der Anzahl der angezeigten Dokumente notwendig. Aufgrund der mit der Wortanzahl exponentiell steigenden Anzahl von zu generierenden Anfragen, wird der Suchaufwand sehr schnell nicht mehr bewältigbar.

Beziehung einzelner Suchwörter untereinander: Die Position jedes Wortes innerhalb eines Textes kann mitgespeichert werden. Dadurch lassen sich dann Anfragen wie folgt realisieren.

- Wort A vor/nach Wort B
- Wort A in Satz B
- Wort A in Kapitel B

Auch hier steigt der Aufwand mit steigender Auflösung der Wortposition. Dies betrifft die Größe des Index und die Dauer der Abfragenbearbeitung.

Nachbearbeitung²: Sie dient in erster Linie der Reduktion der Indexgröße. So läßt sich durch Kleinschreibung aller zu indizierenden Wörter, deren Anzahl auf etwa die Hälfte verkleinern. Andere Verfahren sind:

- Um häufige, nicht inhaltspezifische Wörter wie Artikel und Pronomen vor der Indizierung ausscheiden zu können, bedient man sich einer **Stoppliste**.
- Verschiedene Worte können die gleiche Bedeutung haben. Durch Wahl eines **Bedeutungsrepräsentanten** und **Substitution** der restlichen Wörter läßt sich die Indexgröße reduzieren und die Suche wird effizienter, da sie auf das Bedeutungsfeld eines Wortes ausgedehnt wird. Die Substitution der

²Die diese Verfahren sind sprachabhängig. Zum einem muß die Konfiguration an die jeweilige Sprache angepaßt werden. Zum anderen variiert deren Effizienz mit der zu indizierenden Sprache.

Wörter muß vor dem Indizieren der Dokumente einerseits und vor dem Abschicken einer Suchanfrage andererseits passieren, damit die Wortmengen auf gleiche Weise eingeschränkt sind. Gibt man zum Beispiel das Suchwort “Kuh” ein, wird es zuerst in “Rind” umgewandelt, bevor die Abfrage gestartet wird. Im Index werden Wörter wie “Rind”, “Stier”, “Kalb” und “Ochse” durch “Rind” vertreten. Der Benutzer erhält somit auch Dokumente mit Wörtern der gleichen Bedeutung.

- Beim **Stemming** wird der Wortstamm von Endungen getrennt. Wie die Substitution wird auch das Stemming vor der Indizierung und vor der Suchabfrage durchgeführt. Somit wird nicht nur die Indexgröße reduziert, sondern auch die Formulierung von Anfragen vereinfacht, da Variationen eines Terms nun mitberücksichtigt werden.

Bereichssuche: Durch Verwendung von Bereichsoperatoren und Platzhaltern (*Wildcard*) lassen sich Anfragen leichter formulieren. Die Komplexität der unterstützten Ausdrücke reicht von einfachen Präfixen bis hin zu regulären Ausdrücken.

Gewichtung der Terme: Im Index wird für jedes Wort im Dokument ein Gewicht errechnet und gespeichert. Dieses kann von seiner Häufigkeit im Dokument und seiner Vorkommnis in speziellen Attributen (Titel, Autor, Überschrift, Schlüsselwort) abhängen. Bei der Methode von Salton, Fox und Wu [SFW83] kann der Benutzer zusätzlich jedes Suchwort innerhalb der Abfrage mit einem Gewicht versehen, das der relativen Wichtigkeit des Wortes in der gesamten Abfrage entspricht. Die erhaltenen Dokumente können nun bezüglich dieser Gewichtung gereiht, weiterverwendet oder verworfen werden. Es ist klar, daß die Suche im Index dadurch stark an Komplexität gewinnt.

2.2 Einteilung von gängigen Suchdiensten

Die folgenden Einteilung orientiert sich an der Arbeit von Koch [Koc96] und soll die unterschiedlichen Möglichkeiten beim Aufbau eines Suchsystems aufzeigen und zur Begriffsbestimmung dienen.

- Singuläre Suchdienste
 - Indexsuchdienste mit vollautomatischer Auffindung
 - * Volltext- und reduzierter Volltextindex
 - AltaVista <http://www.altavista.digital.com/>
 - HotBot <http://www.hotbot.com/>
 - Harvest <http://harvest.austria.eu.net/>
 - * Schlüsselwörter und Metadaten
 - Magellan <http://www.mckinley.com/>
 - WWW-Worm <http://www.cs.colorado.edu/www/>

- Katalogdienste mit aktivem Anmelden und Redaktionsteams

Yahoo	http://www.yahoo.com/
Web.de	http://www.web.de/
Dino.online	http://www.dino-online.de/
Henkel	http://www.henkel.co.at/henkel/ha_www_1.html
- Metasuchdienst und Kombinationen
 - Metasuchdienst durch Nutzung mehrerer Suchdienste

MetaCrawler	http://metacrawler.cs.washington.edu:8080/
IBM infoMarket	http://infomarket.ibm.com/
Inference Find	http://www.inference.com:8080/
 - Zusammenfassung mehrerer Katalogsuchdienste

Metaindex European Web	http://www.hj.se/hs/bibl/miewww/
------------------------	---
 - Kombination von Index- und Katalogsuchdiensten

Lycos	http://www.lycos.com/
-------	---
- Intelligent Agents

Verity	http://www.verity.com/
--------	---

Diese Einteilung soll die unterschiedlichen Möglichkeiten beim Aufbau von Suchsystemen aufzeigen. Jedes Verfahren bietet in Teilbereichen Vorteile. Diesen stehen mehr oder weniger ausgeprägte Schwachstellen gegenüber (siehe auch Abschnitt 2.3).

Die Unterschiede in der Gruppe der singulären Suchdienste sind in vielerlei Hinsicht erheblich. Zu den Vorteilen der Katalogsuchdienste, deren Daten “manuell” gesammelt werden, zählt vor allem die Auswahl der Dokumente. Autoren, Lektoren und Benutzer des Systems entscheiden, ob und in welchem Maß Informationsobjekte relevant sind, ob sie in den Katalog aufgenommen und welcher Kategorie sie zugeordnet werden. Dokumente und ganze Informationsserver können in bezug auf ihre Qualität bewertet werden. Die Suche in Themenbereiche basiert auf der Navigation in hierarchisch aufgebauten Kategorien und bietet so dem Benutzer die Möglichkeit, den Zielbereich seiner Suche schrittweise einzuschränken. Katalogsuchdienste können aufgrund ihres Konzeptes sich nicht mit Indexsuchdiensten mit automatischer Auffindung messen, was Größe, Aktualität und Vollständigkeit betrifft (siehe Abschnitt 2.3.2).

Die Vorgangsweise der Suchmaschinen mit vollautomatischer Auffindung (*Spiders*, *Robots*, *Wanderers*, *Worms*) macht diese empfindlich gegen Mißbrauch. So kann zum Beispiel durch verschiedene Tricks³ erreicht werden, daß Dokumente bei der Ergebnisanzeige unangemessen weit vorne platziert werden. Der große Vorteil dieser Dienste liegt in ihrem vergleichsweise hohen Grad an Aktualität und Vollständigkeit (siehe Abschnitt 2.3.2). Innerhalb dieser Gruppe existieren Unterschiede auf dem Gebiet der Indizierung (Volltext, Schlüsselwörter, Metadaten) und dem thematischen und geographischen Deckungsbereich. So indizieren *Magellan* und *WWW-Worm* keinen Volltext, *Web.de* und *Dino.online* nur deutsche Quellen.

³z.B. **Spamming**. Hierbei werden nicht sichtbaren Wörtern im HTML-Dokument zum Zwecke der Manipulation von Ranking-Verfahren untergebracht.

Durch Kombination verschiedener Konzepte wird versucht, für spezielle Anwendungsgebiete einen brauchbaren Kompromiß zu erzielen. Metasuchdienste nutzen selbst verschiedene andere Suchdienste, um qualitativ bessere Ergebnisse zu erzielen. Lycos hingegen ist ein eigenständiger Suchdienst, der aus einem Index- und Katalogsuchdienst besteht. *Agents* [NMW98] haben als Suchdienstkonzept in größeren Bereichen kaum Bedeutung, da nur wenige Informationsanbieter das *Agent*-Konzept unterstützen. So hat auch Verity⁴ nur eine begrenzte Reichweite (Intranet).

2.3 Schwachpunkte gängiger Suchdienste

2.3.1 Netzwerk- und Serverbelastung

Eines der Hauptprobleme der gegenwärtigen Suchsysteme mit vollautomatischer Auffindung ist das vielfache, unkoordinierte Durchsuchen des Netzes durch eine zunehmende Zahl an Suchmaschinen. Dazu kommt, daß sämtliche Rohdaten vom Server geladen, über das Netz gesandt und erst an zentraler Stelle analysiert werden. Um die Informationsserver und das Netz nicht zu überlasten, müssen die Updateintervalle entsprechend groß gewählt werden. Das geht auf Kosten der Konsistenz und Aktualität. In Tabelle 2.1 (aus [Sul98b]) sind u. a. die Größen der Auffindungsintervalle einzelner Suchdienste aufgelistet. Kooperierende, verteilte Systeme berücksichtigen beide oben

Name	Anzahl indizierter Seiten	Seiten pro Tag	Update-Rate ⁵
Alta Vista	140 Millionen	10 Millionen	1 Tag bis 1 Monat
HotBot	110 Millionen	bis 10 Millionen	1 Tag bis 2 Wochen
Lycos	30 Millionen	6 bis 10 Millionen	2 bis 3 Wochen

Tabelle 2.1: Suchdienste in Zahlen (aus [Sul98b] August 1998)

erwähnten Schwächen. Zum einen kann ein lokales Teilsystem direkt beim Informationsserver Daten auffinden, vorbereiten, komprimieren und bereitstellen. Zum anderen können mehrere Suchdienste von diesen aufbereiteten Daten Gebrauch machen. Der Server kann daher wesentlich öfter abgesucht werden, ohne daß es zu Überlastungen kommt. Auf die Vorteile der verteilten Suche wird in Abschnitt 3.4 genauer eingegangen.

2.3.2 Vollständigkeit, Aktualität und Linkkonsistenz

Vollautomatische Suchdienste streben zumindest teilweise Vollständigkeit an. Da selbst große Dienste nicht ständig alle Änderungen erfassen können und das WWW keinem zusammenhängenden Graphen entspricht, läßt sich Vollständigkeit nicht zentral von

⁴<http://www.vertiy.com/>

⁵Die Update-Rate hängt meist von der "Wichtigkeit" des Web-Bereiches ab, d.h. wie oft andere Dokumente darauf verweisen.

einer Stelle aus erzielen. Das gleiche gilt für die Aktualität von indizierten Dokumenten und in weiterer Folge auch für die Konsistenz von Verknüpfungen der Dokumente untereinander. [Bek96]

Um diesem Problemkreis zu begegnen, kann man sich bei der Auffindung zunächst auf einzelne Informationsserver beschränken. Suchprogramme, die am Rechner des Informationsservers im Hintergrund laufen und diesen periodisch absuchen, können so Vollständigkeit innerhalb ihres Bereiches erreichen. Die Aktualität und die Linkkonsistenz hängt von der Größe des Absuchintervalls im Verhältnis zur Änderungsrate am Server ab. Je öfter der Bereich abgesucht wird, desto konsistenter sind die Daten des Suchdienstes. Das Suchprogramm stellt weiters die Information des Servers für übergeordnete Suchdienste in geballter Form bereit. Dieses Konzept führt zur Architektur der verteilten Suche und wird im Abschnitt 3.1 anhand des Harvest-Suchsystems erläutert. [BDH⁺94]

Der Hyperwave-Server (siehe Abschnitt 2.4) verfügt über ein integriertes Suchsystem. Veränderungen des Datenbestandes wirken sich automatisch auf das Suchsystem aus. Die Aktualität des Suchindexes ist daher immer gegeben.

Um einen größeren Informationsraum zur Suche vorbereiten zu können, eignet sich das oben erwähnte Konzept der verteilten Suche in Verbindung mit einer hierarchischen Topologie. Auf unterster Ebene stehen die lokalen Server, die die vollständige Erfassung ihrer Daten sicherstellen müssen. Darüberliegende Suchdienste fassen Informationen mehrerer Server zu Wissensclustern zusammen. Dabei kann dieses Zusammensetzen nach geographischen oder thematischen Gesichtspunkten erfolgen. [GAM98] [GDN⁺98]

Suchkataloge sind aufgrund ihrer Konzeption nicht in der Lage Vollständigkeit zu erreichen. Da diese Suchdienste von Redaktionsteams betreut werden muß, ist der Umfang eines Suchkataloges deutlich kleiner als der eines automatisch erstellten Suchindexes. Die Aktualität hängt ab von [Bek96] :

Web-Autoren : Sie müssen die Dokumente beim Katalog anmelden und sind somit selbst verantwortlich für die Aktualität des Suchdienstes.

Redaktionsteams : Erst wenn die angemeldeten Dokumente von einem Redaktionsteam gesichtet und bewertet wurden, werden dies in den Katalog aufgenommen. Je öfter dieser Vorgang stattfindet, desto besser wird die Aktualität des Kataloges.

2.3.3 Qualität und Zuverlässigkeit

Qualität und Zuverlässigkeit⁶ von angebotener Information bestimmt die Verwertbarkeit für den Benutzer und werden mit der raschen Zunahme an Dokumenten immer wichtiger. Durch die mannigfaltigen Arten von Anbietern und deren Anonymität sind beide Forderungen von vornherein nicht gegeben. [Bek96]

Die Qualitätsangaben von ganzen Informationsservern und auch einzelnen Dokumenten bleibt vorerst Lektoren (manuelle Recherche, siehe Abschnitt 2.1.1) vorbehalten.

⁶Unter Zuverlässigkeit wird mitunter neben der Richtigkeit des Inhaltes auch die Verfügbarkeit (=Erreichbarkeit) der Information verstanden.

ten. Die Bereitstellung von Bewertungen von vorhandener Information könnte sich parallel zu den Suchdiensten als eigenständiger, kommerzieller Dienst entwickeln. Des weiteren könnten Bewertungen durch den Benutzer in das System rückfließen. [GDN+98]

2.4 Suche in Hyperwave

Im Gegensatz zum Webserver, wo ein Suchindex lediglich einen vom Administrator einzurichtenden Zusatz darstellt und externe Suchdienste die Daten über das Netz auffinden und weiterverarbeiten, ist die Suchmöglichkeit in Hyperwave bereits integriert. Nachfolgende Zusammenfassung orientiert sich an [Mau96] und [Hyp98].

Hyperwave setzt nicht wie gewöhnliche Server⁷ auf das Dateisystem, sondern auf eine objektorientiertes Datenbanksystem auf. Während ein Webserver eine unstrukturierte Sammlung von Dokumenten verwaltet, die einzig durch Hyperlinks zusammengehalten und zugänglich gemacht wird, bietet dieses System zusätzlich noch eine hierarchische Organisationsebene (strukturelle Links). Dies erleichtert die Navigation, die Zuordnung einzelner Dokumente u.v.m..

Alle Objekte (Dokumente, Kollektionen, Links) werden beim Anlegen in der Datenbank gespeichert. Durch die bidirektionale Verknüpfung der Hyperdokumente tritt das Problem der Linkinkonsistenz nicht auf. Beim Löschen eines Dokumentes werden aus der Datenbank alle ein- und ausgehenden Verknüpfungen ebenfalls entfernt. Aus dem bisher erwähnten läßt sich schließen, daß auch Aktualität und Vollständigkeit implizit gewährleistet sind.

Die oben genannten Eigenschaften erlauben es nicht nur, die gefundenen Dokumente in ihrer Hierarchie anzuzeigen, sondern auch die benachbarten Dokumente und ihre Verknüpfungsbeziehungen in Form eines mit Hyperlinks versehenen Graphen auszugeben. Suchen und Navigieren fließt ineinander über. Die Suche läßt sich auch auf einen Teilbaum der Dokumentenstruktur beschränken, wobei sich dieser auf verschiedenen Hyperwave-Server befinden kann. Eine Sucheabfrage kann auch auf die Ergebnismenge einer vorangegangenen Anfrage eingeschränkt werden. So kann man schrittweise die Anfrage präzisieren, ohne daß der ganze Bereiche neuerlich durchsucht werden muß.

Jedes Hyperwave-Objekt besitzt ein Vielzahl von Metainformation (Attribute). Die Indizierung der wichtigsten Attribute (Titel, Autor, Schlüsselwörter, Datum der Generierung etc.) macht eine schnelle Suche möglich. Boolesche Verknüpfungen werden bei der Abfrage ebenso unterstützt wie Präfix- und Bereichssuche. Nichtindizierte Attribute können nur in Verbindung mit einer Indexsuche verwendet werden; sie engen das Ergebnis ein. Reguläre Ausdrücke werden unterstützt. Weiters können beliebige Attribute von den Autoren hinzugefügt werden und diese vom System indiziert und somit suchbar gemacht werden.

Zusätzlich zur Suche über die Metainformation ist jedes Textdokument über Volltextsuche zugänglich. Die neueren Versionen ermöglichen auch das Suchen über mehrere Hyperwave-Server (*Serverpool*).

Hyperwave kann statt der eigenen Suchmaschine mit dem Verity-Suchsystem konfi-

⁷Die Autoren von Hyperwave sprechen von Webserver der ersten Generation

guriert werden. Verity verfügt über einen Thesaurus, um auch Wörter gleicher Bedeutung in die Suche miteinbeziehen zu können. Stemming wird ebenso angewandt wie ein Verfahren, das erlaubt, nach ähnlich klingenden Wörtern zu suchen. Dabei ist die Suche nicht auf HTML-Dokumente beschränkt. Formate wie *Word* und *Excel* werden unterstützt.

Das Hyperwave-Suchsystems ermöglicht die Verwendung sogenannter *Query Objects*. Diese Suchanfragen werden im Hyperwave-Server gespeichert und periodisch ausgewertet. Die Resultate werden via Email an den Benutzer versandt. Sie können auch durch vom Benutzer zu beliebiger Zeit ausgeführt werden (Konzept der vordefinierte Suche). Es wäre naheliegend, die vom System indizierten Daten in geeigneter Form⁸ auch anderen Suchsystemen zugänglich zu machen.

2.5 Zusammenfassung

Verschiedene Methoden der Lokalisierung von Dokumenten und der Aufbereitung deren Inhalts sind die Basis der Suchdienste. Durch aktives Anmelden können Autoren ihre Dokumente in einen Suchdienst aufnehmen lassen. Bei der redaktionellen Recherche verfolgen Lektoren die Hypertext-Strukturen und nehmen passende Dokumente in ihren Suchdienst auf. *Robots*, *Bots* bzw. *Spiders* navigieren mit Hilfe von Hyperlinks, Menüs und Verzeichnisstrukturen im Informationsraum und verarbeiten die aufgefundenen Dokumente automatisch.

Die Bereitstellung der Information wird durch thematische Kataloge, durch einen Suchindex oder durch Mischformen aus beiden realisiert. Während der Benutzer bei Verwendung eines Kataloges sich schrittweise der gewünschten Dokumentenmenge "nähert", muß er beim Suchindex durch Angaben von Suchwörtern seine Suche formulieren. Unterstützt wird er dabei durch verschiedene Verfahren (siehe Abschnitt 2.1.2).

Gängige Suchdienste bauen auf diese Basismethoden auf. Sie arbeiten meist unabhängig von einander und laden Daten vom Informationsserver über das Netz, um den Inhalt zu verarbeiten und bereitzustellen. Diese unkoordinierte, zentrale Vorgangsweise führt zu enormer Netz- und Serverlast. Weitere Probleme liegen zum Beispiel auf dem Gebiet der Aktualität und der Qualität der angebotenen Information. (siehe Abschnitt 2.3)

Der Hyperwave-Server (siehe Abschnitt 2.4) verfügt über ein integriertes Suchsystem. Veränderungen des Datenbestandes wirken sich automatisch auf das Suchsystem aus. So ist zum Beispiel die Aktualität des Suchindexes ist immer gegeben.

Das Harvest-System [BDH⁺94] der Universität von Colorado ist ein Vertreter des Prinzips der verteilten Suche. Viele der in diesem Kapitel beschriebenen Schwachpunkte gängiger Suchmaschinen können durch dieses Konzept vermieden werden. Dieses System wird im folgenden Kapitel vorgestellt.

⁸zum Beispiel im SOIF-Format des Harvest-Suchsystems (siehe Abschnitt 3.1) oder im XML-Format

Kapitel 3

Das Harvest-Suchsystem

Im folgenden Kapitel wird ein System der Universität von Colorado vorgestellt, das sich durch Vermeidung vieler der in Kapitel 2 erwähnten Schwächen auszeichnet. Das Harvest-System¹ [BDH⁺94] stellt eine Sammlung von Werkzeugen zur Auffindung, Aufbereitung, Verwaltung und Suche von Informationen im Internet dar. Es kann Daten aus den verschiedensten Dokumentformaten (HTML, \LaTeX , Tar- und Zipfiles, etc.) extrahieren. Zur Indizierung verwendet das Harvest-System externe Werkzeuge wie zum Beispiel Glimpse [WM93].

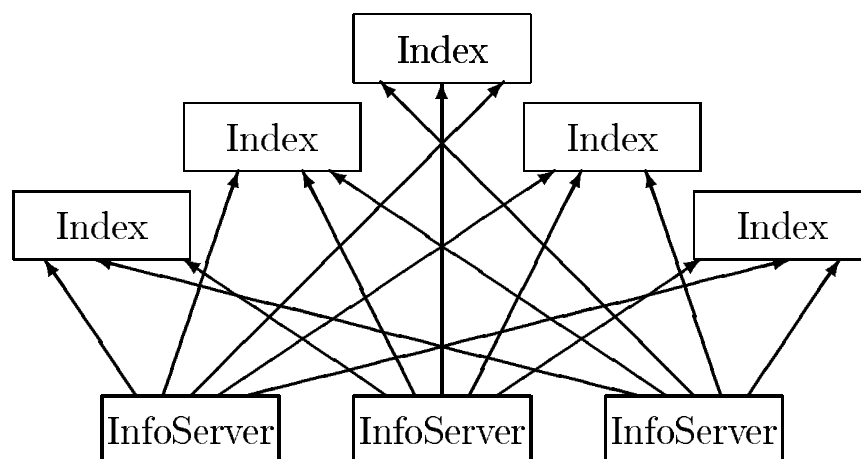
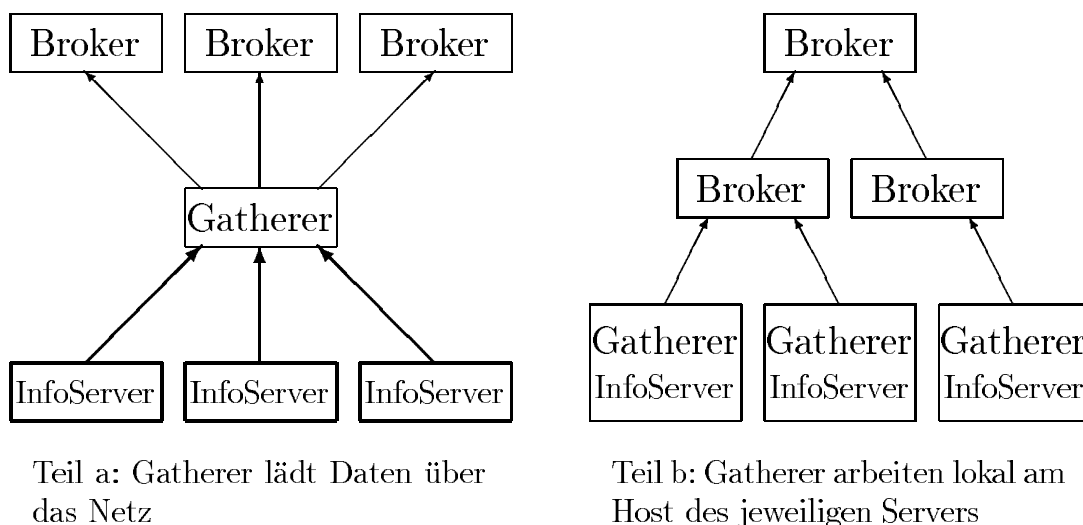


Abbildung 3.1: Ineffizienz gewöhnlicher Suchsysteme (aus [BDH⁺94])

3.1 Das Harvest-Konzept

Das Grundkonzept des Harvest-Systems [BDH⁺94] beruht auf der Idee der verteilten Informationsauffindung, Datenaufbereitung und Bereitstellung. Während zentrale

¹<http://www.harvest.cs.edu>

Abbildung 3.2: Lokal und über das Netz arbeitende Gatherer (aus [BDH⁺94])

Suchsysteme (siehe Abbildung 3.1) jedes für sich sämtliche gefundenen Dokumente über das Netz anfordern, dann analysieren und aufbereiten, geht Harvest einen anderen Weg. Ein Teil des Suchsystems ist für die Auffindung und Aufbereitung der Daten zuständig. Wie in Abschnitt 3.2 gezeigt wird, sollte dieser Teil möglichst lokal am Hostrechner des Informationsservers arbeiten. Der andere Teil des Systems bedient sich der so aufbereiteten Daten und stellt sie dem Suchenden zur Verfügung. Die Beschreibung im folgenden Abschnitt beruht auf [BDH⁺94] und [HSW96].

Das Harvest-Suchsystem bietet die Möglichkeit des Aufbaues eines verteilten, kooperativen Suchsystems mit Hilfe zweier unterschiedlicher Komponenten:

Der Gatherer sammelt und extrahiert periodisch verschiedenartige Dokumente von einem oder mehreren Informationsserver und generiert für jedes aufgefundene Dokument ein sogenanntes SOIF²-Objekt. Dieses wird bei Bedarf weitergeleitet. Der Gatherer läuft üblicherweise am Hostrechner des Informationsservers. Abbildung 3.2.b zeigt eine solche Konfiguration. Gatherer, die am selben Host wie der Informationsserver arbeiten, geben Information an übergeordnete Instanzen weiter. Pro Server ist genau ein Gatherer vorhanden. Es ist jedoch auch möglich, daß ein oder mehrere Informationsserver von einem Gatherer bearbeitet werden, der auf einem anderen Rechner läuft. Abbildung 3.2.a zeigt einen Gatherer, der Daten von drei entfernten Informationsservern über das Netz lädt und diese an drei übergeordnete Module weiterleitet.

Der Broker fragt die ihm zugeordneten Gatherer periodisch nach Änderungen des Informationsangebotes ab. Die erhaltenen SOIF-Objekte werden von ihm verwaltet und der Inhalt mittels externer Werkzeuge indiziert. Dem Benutzer stellt er über einen WWW-Client ein Suchinterface zur Verfügung.

²Das **S**ummary **O**bject **I**nterchange **F**ormat wird in später in diesem Abschnitt beschrieben.

Gatherer und Broker können nun auf unterschiedlichste Weise zusammenarbeiten. Der linke Teil der Abbildung 3.2 zeigt einen Gatherer, der über das Netz auf die Daten des Servers zugreift; der Vorteil einer lokalen Suche kommt nicht zum tragen. Dennoch hat diese Konfiguration ihre Berechtigung, da aufgrund des Zusammenarbeitens mit mehreren Brokern die Belastung für Netz und Server reduziert werden kann.³ Der rechte Teil der Abbildung 3.2 skizziert eine Struktur, bei der sich der Gatherer am Rechner des Informationsservers befindet. Wie durch die dünneren Pfeile und Umrandungen angedeutet wird, werden bei dieser Konfiguration Netz und Server wesentlich weniger beansprucht. Die einzelnen Methoden, durch die diese Effizienzsteigerung möglich ist, werden im Abschnitt 3.2 erläutert.

Der Broker kann Informationen von mehreren Gatherer beziehen. Ein Gatherer wiederum kann seine gesammelten Informationen an verschiedene Broker übermitteln. Schließlich können auch Broker ihre Informationen an andere Broker weiterreichen. Durch dieses Zusammenspiel der hierarchisch angeordneten Komponenten werden die Kosten für das Auffinden und Aufbereiten der Daten minimiert.

Effizienz ist nicht die einzige Stärke der verteilten Suche. Durch die Nähe des Gatherers zum Informationsserver können zum Beispiel Informationen über den Server übernommen und Konfigurationsparameter individuell angepaßt werden.⁴ Durch die Gliederung des Informationsangebotes lassen sich weitere wichtige Eigenschaften erreichen. Diese Vorteile werden in Abschnitt 3.4 erläutert.

Das Metadatenformat SOIF

Wie schon eingangs erwähnt, generiert der Gatherer für jedes aufgefundene Informationsobjekt eine Inhaltszusammenfassung (*Content Summary*). Diese enthält Attribut-Wert-Paare und wird im sogenannten **S**ummary **O**bject **I**nterchange **F**ormat (SOIF) gespeichert. SOIF basiert auf einer Kombination von IAFA-Templates und BibTeX.

SOIFSTROM	→	OBJEKT OBJEKT SOIFSTROM
OBJEKT	→	@TYPE {URL AW-LISTE}
AW-LISTE	→	AW-LISTE AW-PAAR AW-LISTE
AW-PAAR	→	ATTRIBUT {GRÖSSE} BEGRENZER WERT
ATTRIBUT	→	Zeichenkette
WERT	→	Zeichenkette
BEGRENZER	→	:<tab>

Abbildung 3.3: Das SOIF-Format: Formale Beschreibung in BNF (aus [HSW96])

Jedes SOIF-Objekt beinhaltet die URL des Dokumentes und eine Liste von Attribut-Wert-Paaren. Nach dem Namen des Attributes steht die Länge des Inhaltes. Dieses Format enthält aber nicht nur eine Teilmenge des Inhaltes des Originaldokumentes. Unter-

³Hier wird ein bestimmter Bereich, ein Infocuster, von einem Gatherer abgedeckt.

⁴Der lokale Administrator weiß im allgemeinen am besten, wie der Inhalt des Servers beschaffen ist, welches Layout verwendet wird, wie oft Dokumente verändert werden, etc.

<pre> <HTML> <HEAD> <META NAME="author" CONTENT="Dagobert Duck"> <META NAME="description" CONTENT="Mit Einfuehrung des Euro aendert sich die Bedeutung des Dollar als Weltwaehrung. Es werden die zu erwartenden wirtschaftlichen Auswirkungen auf Entenhausen untersucht."> <META HTTP-EQUIV="KEYWORDS" CONTENT="Dollar Euro Entenhausen"> <TITLE>Der Euro und seine Auswirkungen auf Entenhausen </TITLE> </HEAD> <BODY> <!-- der inhalt des folgenden textes ist frei erfunden und ist nicht ernst zunehmen. --> <H1>Der Euro</H1> Was noch vor ein paar Jahren niemand in Entenhausen fuer moeglich hielt, ist nun Gewissheit. Die Staaten der EU bekommen eine gemeinsame Waehrung. </BODY> </HTML> </pre>	<pre> @FILE { http://www.dagobert.eh/euro/euro.htm time-to-live{6}: 360000 last-modification-time{9}: 921234650 refresh-rate{6}: 241920 update-time{6}: 84600 type{4}: html file-size{4}: 2099 md5{32}: abcdef1234567890abcdef1234567890 gatherer-name{14}: Weltwirtschaft author{13}: Dagobert Duck body{155}: Der Euro Was noch vor ein paar Jahren niemand in Entenhausen fuer moeglich hielt, ist nun Gewissheit. Die Staaten der EU bekommen eine gemeinsame Waehrung. description{166}: Mit Einfuehrung des Euro aendert sich die Bedeutung des Dollar als Weltwaehrung. Es werden die zu erwartenden wirtschaftlichen Auswirkungen auf Entenhausen untersucht images{30}: /duck/bilder/duck_und_euro.jpg headings{8}: Der Euro keywords{49}: Auswirkungen Dollar EU Entenhausen Euro Waehrung title{47}: Der Euro und seine Auswirkungen auf Entenhausen url-reference{43}: www.entenhausen.org www.eu.gv } </pre>
--	---

Abbildung 3.4: Eine HTML-Datei und ein daraus generiertes SOIF-Objekt

schiedlichste Zusatz- und Metainformationen können mit Hilfe der SOIF-Attribute festgehalten werden. Welche Metainformation im SOIF-Objekt steht, hängt beim Harvest-System vom *Summarizer* des entsprechenden Dokumentformates ab. Abbildung 3.4 zeigt ein HTML-Dokument und ein mögliches korrespondierendes SOIF-Objekt. Bei diesem Beispiel werden unter anderem fett gedruckte Textteile als Schlüsselwörter interpretiert und dem **keywords**-Attribut zugeordnet. Die Information des Meta-Tags fließt ebenfalls in die entsprechenden Attribute des SOIF-Objektes. Links werden erkannt und die Zieladresse im Attribut **url-reference** abgelegt. Ähnlich wird mit den Inhalten anderer HTML-Tags und HTML-Attributen verfahren. [HSW96]

Das SOIF-Format beschreibt keine starre Konvertierung von Dokumenten. Es stellt viel mehr einen Rahmen dar, der es erlaubt, den Inhalt unterschiedlichster Originaldokumente in ein einheitliches Objekt abzubilden, wobei die Zuordnung zu den Attributen individuell konfigurierbar ist.

Wesentlicher Vorteil dieses Formates ist die einfache Verarbeitung. SOIF läßt sich “on-the-fly” untersuchen und da es ohne spezielle Zeichen auskommt, braucht es keinen Escape-Mechanismus. Eine Menge von SOIF-Objekten läßt sich zu einem Paket zusammenstellen, das als Strom übertragen wird. Nachteilig wirkt sich aus, daß ein Fehler innerhalb des SOIF-Stroms dessen gesamten restlichen Inhalt unbrauchbar macht. [NK94]

Die meisten Module des Harvest-Systems befassen sich unmittelbar mit dem SOIF-Format [Wes95]:

- Die Module des Essence-Subsystems (*Summarizer*) geben SOIF-Objekte aus (siehe Abschnitt 3.2.1).
- Die “Datenbank”⁵ des Gatherer beinhaltet SOIF-Objekte.
- Der Gatherer exportiert den Inhalt dieser Datenbank in Form eines SOIF-Stromes.
- Der Broker empfängt, indiziert und speichert SOIF-Objekte im Dateisystem.
- Der Broker exportiert wieder SOIF-Ströme.

3.2 Der Gatherer

Der Gatherer des Harvest-Systems [BDH⁺94] sammelt periodisch Dokumente innerhalb eines definierten Bereiches und extrahiert deren Inhalt. Der Auffindungsbereich wird durch Start-URLs und maximalen Suchtiefen festgelegt und durch konfigurierbare URL-, Server- und IP-Domainfilter eingeeengt. Die Linkverfolgung wird in Abschnitt 3.2.2 erläutert. Neben verschiedenen Zugriffsarten wie FTP, Gopher, NetNews, HTTP und lokales Dateisystem unterstützt das Harvest-System mannigfaltige Dateiformate. Auf diese und deren Aufbereitung wird im Abschnitt 3.2.1 eingegangen. Die Flexibilität des Gatherer erlaubt es, auf das Harvest-System die unterschiedlichsten Suchdienste aufzusetzen.

Um einen Index von zum Beispiel HTML-Objekten erstellen zu können, muß bei zentralen Suchsystemen jedes einzelne Objekt über das Netz angefordert und geladen werden. Anschließend erfolgt das Extrahieren der Hyperlinks. Mit deren Zieldokumenten wird gleich verfahren. Diese Methode ist äußerst ineffizient, da so für jedes Objekt eine TCP/IP-Verbindung errichtet und ein neuer Prozeß gestartet werden muß.

Ein Gatherer, der resident am Rechner des Providers ist, vermeidet diesen Overhead. Er greift im allgemeinen über das lokale Dateisystem auf die Daten zu und sucht die HTML-Objekte periodisch ab. Dabei werden nur Dokumente neu untersucht und in einen lokalen Cache aufgenommen, die sich seit dem letzten Mal verändert haben. Wie die Konsistenz zwischen Informationsserver und Broker erreicht wird, wird in Abschnitt 3.3 gezeigt. Bei einer Anfrage eines Broker an einen Gatherer können die extrahierten Daten somit auf einmal geladen werden. Dies führt zu enormen Verringerung der Serverlast. Konkrete Messungen und Zahlen werden in [BDH⁺94] erläutert. Um die Netzwerklast zu minimieren, verwendet der Harvest im wesentlichen drei Verfahren:

- Das **Essence-Subsystem** (Abschnitt 3.2.1) des Gatherers extrahiert aus dem Inhalt der Informationsobjekte sogenannte *Content Summaries* (SOIF-Objekte), die dann an einen oder mehrere Broker geschickt werden. Diese Objekte sind, abhängig von der Konfiguration, im wesentlichen kleiner als die ursprünglichen Dokumente.

⁵Der Gatherer speichert die SOIF-Objekte im Filesystem.

- Alternativ zum normalen Format können die Daten auch komprimiert übertragen werden.
- Je größer die Update-Rate des Index im Vergleich zur mittleren Zeitspanne der Veränderungen der Dokumente ist, desto sinnvoller sind **inkrementelle Verfahren** zur Indexerstellung. Hierbei wird ein Objekt nur im Falle einer inhaltlichen Veränderung neuerlich übertragen. Im Fall von HTTP verwendet der Gatherer den “if-modified-since”-Headereintrag in der Anfrage. Greift der Gatherer direkt auf ein Dateisystem zu, entnimmt er aus dem Dateieintrag den Zeitpunkt der letzten Änderung. Zusätzlich bildet er eine MD5-Prüfsumme über die einzelnen Dokumente und kann gegebenenfalls mit ihrer Hilfe entscheiden, ob ein Dokument neu aufgenommen werden muß. Durch dieses inkrementelle Verfahren spart man sich einerseits das Extrahieren des Originaldokumentes und andererseits die Übertragung zum Broker und die anschließende Indizierung.

3.2.1 Das Essence-Subsystem

Das Essence-System [HS95] des Gatherer teilt die Aufgabe der Informationsextraktion auf mehrere Komponenten auf. Der Typ des Informationsobjektes muß erkannt, eventuelle Einkapselung aufgelöst und die zur Indizierung verwendbaren Dateien ausgewählt werden. Erst dann folgt der eigentliche Prozeß des Extrahierens. Speziell für den entsprechenden Dokumenttyp entwickelte Module filtern Information aus dem Objekt und stellen sie in einem SOIF-Objekt zusammen. Alle typenspezifischen Methoden können außerhalb des Essence-Systems definiert, ergänzt und konfiguriert werden.

Typerkennung: Sie wird mit Hilfe der Erweiterung des Dateinamens (zum Beispiel *.html für HTML-Dateien) und durch implizite Erkennung, indem der Dokumentinhalt selbst untersucht wird (*Magic Number Test*), durchgeführt. Bei typenbehafteten Dateisystemen (zum Beispiel OLE⁶ bei Microsoft Windows) wird die Typinformation aus dem Dateieintrag selbst gewonnen.

Entkapselung: Die Typenerkennung kann eine Einkapselung (Komprimierung, Gruppierung, Verschlüsselung) feststellen. Diese wird nach Möglichkeit aufgelöst. Das Ergebnis ist im allgemeinen eine Menge von Dateien.

Auswahl: In diesem Schritt werden Vertreter uninteressanter Objekttypen ausgeschieden. Dieses Verhalten ist natürlich konfigurierbar. Falls durch eine Entkapselung aus einem Objekt mehrere Teile entstanden sind, wird zusätzlich die Redundanz berücksichtigt und Einzelobjekte entsprechend verworfen. So können zum Beispiel alte Dateiversion (zum Beispiel *.bak) eliminiert werden, wenn eine neue Version existiert und Objektcode ausgeschieden werden, wenn der entsprechende Quellcode vorhanden ist.

Inhaltszusammenfassung: Basierend auf der Typinformation wird in diesem Schritt eine passende Extraktionsprozedur (*Summarizer*) zur Verarbeitung des Objektes

⁶Object Link Embedded

aufgerufen. Jeder Dokumenttyp hat einen eigenen *Summarizer*, der wiederum individuell vom Administrator des Gatherer angepasst werden kann. Aufgrund des objektorientierten Ansatzes des Essence-Systems, lassen sich Methoden eines *Summarizer* an andere vererben. So ist der *HTML-Summarizer* ein Spezialfall des *SGML-Summarizer*. Dieser hält sich genau an die Spezifikationen von HTML. Für gängige, selbsterstellte HTML-Dokumente ist dieses Verfahren zu wenig fehler-tolerant.

Wie dieser Teil des Essence-Systemes für den Fall von HTML-Dateien funktioniert und welche Probleme die Verwendung des *SGML-Summarizer* mit sich bringt, wird im Abschnitt 4.1 diskutiert.

3.2.2 Die Linkverfolgung

Um den Auffindungsbereich im Informationsraum zu bestimmen, hat der Administrator mehrere Möglichkeiten. Er kann die URL der Objekte individuell festlegen (*leave node*) oder einen Startpunkt (*root node*) mit einer maximalen Verzweigungstiefe angeben. Zusätzlich können Stopplisten mit URLs von Dokumenten und Sides geführt werden. So läßt sich zum Beispiel erreichen, daß sich der Gatherer auf Objekte innerhalb eines bestimmten Bereiches beschränkt. Die Verwendung von regulären Ausdrücken in den Konfigurationsdateien erlaubt eine Vielfalt an Einstellungen. [HSW96]

Damit der Vorteil eines lokalen Gatherer zum tragen kommt, müssen die URLs, die er von den Konfigurationsdateien liest und im Laufe des Auffindungsprozesses extrahiert, in eine URL des Dateisystems umgewandelt werden. Im allgemeinen werden die Dokumente eines Informationsservers in einem eigenen Verzeichnis verwaltet. Dieses Verzeichnis kann für den Zugriff über den Server neu benannt werden. Somit gelangt keine Information über die tatsächliche Verzeichnisstruktur nach außen. Der Administrator kennt beide Bezeichnungen, den lokalen Pfad und die Entsprechung für HTTP oder FTP. In der Konfigurationsdatei kann er für beliebige URLs den lokalen Pfad vermerken. Extrahiert der Gatherer eine nicht-lokale URL, kann er diese, falls ein Eintrag in der Konfigurationsdatei existiert, in einen Pfad des Dateisystems umwandeln und über diesen Weg lokal auf die Datei zugreifen. [HSW96]

Der Gatherer untersucht nun zu festgesetzten Zeiten die Startdokumente und ermittelt, nachdem der Inhalt behandelt wurde (siehe Abschnitt 3.2.1), die darin enthaltenen Links. Mit den Zieldokumente der Links wird nun auf gleiche Weise verfahren. Beendet wird dieser Prozeß, wenn alle Links im definierten Bereich verfolgt wurden.

3.3 Der Broker

Der Broker des Harvest-Systems [BDH⁺94] ist die Schnittstelle zwischen Gatherer und Index einerseits und zwischen aufgefundenem Wissen und Benutzer andererseits. Er verwaltet die vom Gatherer bereitgestellten SOIF-Objekte und veranlaßt die Indizierung durch externe Werkzeuge. Dem Benutzer stellt er mit Hilfe eines Web-Clients und eines CGI-Skripts eine Schnittstelle zur Kommunikation mit dem Suchsystem bereit.

Des weiteren kann er seine SOIF-Objekte einem anderen Broker zur Verfügung stellen. Folgende Teilmodule führen die Arbeit des Broker aus [Cam94]:

Der Registry-Manager wartet eine Liste der im Broker existierenden SOIF-Objekte. Der *Registry*-Eintrag setzt sich aus einem Ablaufdatum- und einem ID-Feld, das aus URL und Gatherer-Information gebildet wird, zusammen. Das Ablaufdatum wird aus der Summe des “update-time”⁷- und des “time-to-live”⁸-Wertes gebildet. Wenn dieser Zeitpunkt erreicht ist, wird das Objekt aus der *Registry* entfernt. Bei jedem Update des Broker wird so die Lebensdauer der noch am Informationsserver befindlichen Objekte verlängert.⁹

Der Storage-Manager archiviert die Sammlung von SOIF-Objekten, indem er sie im darunterliegenden Dateisystem speichert. Er eliminiert mehrfach vorhandene Objekte. Zum Vergleich bedient er sich der MD5-Signatur und der Gatherer-ID.

Der Collector fragt periodisch die Gatherer und andere Broker nach Updates ab. Diese antworten mit einem SOIF-Strom, der die in ihrem Bereich aufgefundenen Dokumente beschreibt. Objekte, die seit der letzten Brokeranfrage neu erstellt oder inhaltlich verändert wurden, müssen zum *Storage Manager* weitergeleitet und in der *Registry* neu aufgenommen werden. Um die neuen Dokumente zu indizieren, ruft der *Collector* anschließend über das Indexinterface den externen Indexer auf. Die Lebensdauer der Objekte, die vor der letzten Anfrage vorhanden waren und deren Inhalt sich nicht verändert hat, wird in der *Registry* verlängert, indem die neue “update-time” in die Berechnung einfließt.

Der Query-Manager empfängt Anfragen von einem Suchclient (Web-Client via CGI), übersetzt diese in ein Format, das von der angebunden Suchmaschine akzeptiert wird, und schickt es dieser. Die Suchmaschine antwortet mit einer Liste von “Identifiern” von Objekten, die der Suchanfrage genügen. Der *Query Manager* erzeugt nun eine kurze Beschreibung dieser Objekte und schickt sie dem Suchclient zurück. Im Fall, daß der Broker von einem darüberliegenden Broker abgefragt wird, werden alle entsprechenden SOIF-Objekte als Ganzes, und nicht deren Kurzbeschreibung, übertragen. Dies geschieht mit einem eigenen Protokoll (*Bulk Transfer*). So ist es für einen Broker machbar, eine Teilmenge des Inhaltes eines anderen Broker aufzunehmen.

Wenn ein Dokument als SOIF-Objekt im Broker erstellt wird, wird eine Objekt-ID in der *Registry* hinzugefügt; das *Summary*-Objekt wird vom *Storage Manager* archiviert und von der Suchmaschine indiziert. Befindet sich ein Objekt beim nächsten Update des Broker noch immer am Informationsserver, wird seine Lebensdauer neu berechnet. Falls dieser Eintrag abläuft, weil das korrespondierende Dokument am Informationsserver

⁷Zeitpunkt des Auffindens am Informationsserver. Wird vom Gatherer im SOIF-Objekt bei jedem Auffinden vermerkt.

⁸Zeitspanne innerhalb der das Objekt im Broker existieren kann, ohne das das entsprechende Dokument am Server aufgefunden wurde. Kann aus dem gleichnamigen SOIF-Attribut entnommen oder direkt vom Broker-Administrator festgelegt werden.

⁹Aspekte der Konsistenz zwischen Broker und Gatherer werden am Ende dieses Abschnittes diskutiert.

entfernt wurde, wird das Objekt lediglich aus der *Registry* gelöscht. So kommt es, daß im Laufe der Zeit die Inkonsistenz zwischen Inhalt des *Storage Manager* und der *Registry*-Einträge zunimmt. Daher wird periodisch eine Bereinigung dieses Zustandes (*Garbage Collection*) durchgeführt.

Das oben erwähnte “time-to-life”-Attribut legt die Zeitspanne fest, innerhalb derer ein Objekt im Broker existieren kann, obwohl der zuständige Gatherer das entsprechende Dokument nicht mehr aufgefunden hat. Dies hat den Vorteil, daß im Fall von kurzzeitiger Außerbetriebnahme des Informationsservers oder von Wartungen einzelner Dokumente, die Einträge im Index nicht gelöscht werden müssen, um beim nächsten Update wieder neu generiert zu werden.

Die Konsistenz zwischen Suchindex des Broker und Dokumenten am Informationsserver sinkt also mit der Länge des Abfrageintervalls und der Differenz zwischen “time-to-live”-Eintrag sowie der tatsächlichen Lebensdauer. Die Lebensdauer in der *Registry* kann fix vom Administrator des Broker vergeben oder aus dem SOIF-Objekt entnommen werden. Der Gatherer kann sie aus den Metadaten¹⁰ des Dokuments berechnen.

Die Administration des Broker wird über den *Query Manager* abgewickelt. Viele Konfigurationsparameter können so manipuliert werden. Dazu gehören die Liste der Gatherer und Broker, die als Informationsquelle dienen, die Standardlebensdauer der Objekte, die Abfragerate und die Rate der *Garbage Collection*.

3.3.1 Das Indexinterface

Damit der Broker verschiedensten Ansprüchen gerecht werden kann, wird eine flexible Schnittstelle zwischen Broker und Indexer definiert, nicht aber ein spezieller Indexer spezifiziert. Unterschiede im Design von Indexern erschweren dabei das Erstellen eines Interface. Einige Indexer indizieren objektweise, andere arbeiten besser, wenn sie viele Objekte gleichzeitig indizieren können. Einige Indexer können keine einzelnen Objekte löschen, was eine Reindizierung notwendig macht. Aufgrund dieser Differenzen werden vom Broker vier Interfaceoperationen bereitgestellt [Cam94] [HSW96]:

Update_Object wird vom *Collector* bei jedem Update im *Storage Manager* pro Objekt einmal aufgerufen.

Index_Flush wird vom *Collector* nach jedem Update gestartet.

Garbage_Collect wird vom Broker zum Löschen von nicht mehr aktuellen Einträgen im Index verwendet. Falls der Indexer dies nicht unterstützt (Glimpse), wird von hieraus die Reindizierung eingeleitet.

Resolve_Query generiert aus dem brokereigenen Abfrageformat das Format des jeweiligen Indexers. Dieses wird dem Indexer übermittelt, das Resultat analysiert. *Resolve_Query* gibt die IDs der im Ergebnis vorkommenden Objekte zurück.

¹⁰zum Beispiel bei HTML: <META NAME="expires" CONTENT="DEC 09 1998">

Zum Beispiel kann ein Indizerwerkzeug, das Objekte einzeln bearbeitet, dies mit *Update_Object* tun; *Index_Flush* wird nicht gebraucht. Für einen andere Indizierer, der Objekte gebündelt indiziert, werden mit *Update_Object* die einzelnen Update-Anfragen zwischengespeichert und am Ende mit *Index_Flush* gesammelt indiziert.

3.3.2 Das Suchinterface

Damit der Benutzer über das Web mit dem Broker kommunizieren kann, wurde ein Webinterface implementiert. Dieses besteht im wesentlichen aus [HSW96]:

- HTML-Dateien, die Forms¹¹ verwenden. Sie stellen die grafische Benutzerschnittstelle (GUI¹²) dar.
- einem CGI¹³-Programmteil, das die Daten aus den HTML-Forms liest, daraus eine Anfrage formuliert und diese schließlich an den *Query Manager* des Broker leitet.
- einem CGI-Programmteil, das das Resultat der Anfrage in ein HTML-Dokument bettet und dessen Darstellung am Webbrowser veranlaßt. Beide CGI-Programmteile sind somit die Verbindung zwischen GUI und der Funktionalität des *Query Manager*.

Das Suchinterface unterstützt, abhängig von der verwendeten Suchmaschine, unterschiedlichste Anfragetypen. Die Anfragesyntax ist einheitlich, da erst im Indexerinterface das interne Anfrageformat in das entsprechende Format des Indexers umgewandelt wird. Kann ein spezieller Broker eine Anfrage nicht weiterbehandeln, weil der Indexer deren Typ nicht unterstützt, wird eine Fehlermeldung ausgegeben. Glimpse¹⁴ [WM93] kann folgende Anfragetypen verarbeiten:

- wahlweise Berücksichtigung der Groß- und Kleinschreibung
- Wildcards
- Suche nach Phrasen
- boolsche Verknüpfungen
- Begrenzung der Suche auf spezielle SOIF-Attribute
- Möglichkeit der Angabe der maximalen Anzahl der Resultate, die ausgegeben werden sollen

¹¹Mit dem **FORM**-Tag wird ein Formular innerhalb eines HTML-Dokumentes zur Eingabe von Daten durch den Anwender definiert. Verschiedene Feldtypen wie Textfeld, Checkbox und Radiobutton können in die Formulare integriert werden.

¹²Graphical User Interface

¹³Common Gateway Interface

¹⁴<http://glimpse.cs.arizona.edu/>

3.4 Vorteile der verteilten Suche

Aufgrund der Topologie des Gesamtsystems und der Nähe zur Wissensquelle ergeben sich Vorteile, die nachfolgend beschrieben werden und die Ursache für weitere Effizienzsteigerungen sind. Speziellen Verfahren des Harvest-Suchsystems erzielen weitere Verbesserungen.

Es sei hier angemerkt, daß das System der verteilten Suche auch einen großen Nachteil hat. Mit der Verteilung der Aufgaben geht zwangsläufig ein Mehraufwand an Organisation einher. So ist man in erster Linie auf die Mitarbeit der Informationsanbieter angewiesen, will man einen verteilten Suchdienst ins Leben rufen. Da diese aber im allgemeinen an der Verbreitung ihrer im Netz befindlichen Dokumente interessiert sind, dürften sich die Widerstände in Grenzen halten.

3.4.1 Netzwerk- und Serverbelastung

Wie in Abschnitt 3.2 schon erwähnt wurde, ist die Reduktion der Last für Netz und Server beim Harvest-Verfahren mehrfach begründet:

- Die Auffindung im lokalen Dateisystem ist wesentlich effizienter als die Kommunikation via HTTP.
- Durch Übertragung einer Sammlung von vielen Objekten statt Einzelübertragung, werden aufwändige Betriebssystemaufrufe (UNIX-Fork) gespart.
- Die Erstellung von Inhaltszusammenfassungen (*Content Summaries*) und deren mögliche Komprimierung verringert die zu sendende Datenmenge.
- Inkrementelles Update des Broker macht nur die Übermittlung der Veränderungen notwendig.
- Schließlich reduziert die Zusammenarbeit vieler Dienste den Aufwand der Informationsauffindung und Vorverarbeitung.

3.4.2 Vollständigkeit, Aktualität und Linkkonsistenz

Das System der verteilten Suche beschränkt sich bei der Erreichung der Vollständigkeit auf Teilbereiche. Die kleinste Einheit ist durch den Informationsserver gegeben. Mittels lokalem Gatherer kann, aufgrund der Nähe und geringerem Aufwand, der Informationsserver wesentlich häufiger als bei zentralen Suchmodellen abgesucht werden. Dies führt zu einem guten Maß an Aktualität und somit zu Konsistenz innerhalb des Untersuchungsgebietes. [BDH⁺94]

Die Absuchrate kann für jeden Server individuell angepaßt werden. So werden Server mit dynamischem Informationsangebot öfter untersucht werden müssen, als solche, deren Inhalt sich kaum verändert. [HSW96]

Die Zusammenfügung von mehreren Teilbereichen eines betrachteten Informationsraumes erweitert die Vollständigkeit auf den gesamten Bereich, ohne daß sich die Aktualität verschlechtern würde.

3.4.3 Gliederung des Informationsraumes

Wie in Abschnitt 3.2.1 und Abschnitt 3.2.2 beschrieben wurde, kann ein Administrator die Gatherer in seinem Bereich individuell anpassen. So kann dieser aufgrund seiner Kenntnis, in welchen Verzeichnissen welche Themen behandelt werden, Gatherer für unterschiedliche Wissensgebiete einrichten.

Die Broker können sich nun in zweierlei Hinsicht spezialisieren. Entweder kontaktieren sie nur Gatherer und Broker, die sich innerhalb eines geographischen Gebietes beziehungsweise einer Domain befinden, oder sie wählen nur themenspezifische Quellen aus. Natürlich setzt dies das Wissen um die Gatherer und Broker und deren Angebot voraus. Dieses Wissen kann als Zusatzinformation (siehe Abschnitt 3.4.5) über den Server und die Website vom Gatherer zur Verfügung gestellt werden. Kooperation braucht Organisation.

Die Möglichkeit der Zusammenarbeit einzelner Suchdienste kann in einer hierarchischen Struktur erfolgen. Bei dieser Topologie nimmt der Spezialisierungsgrad nach oben hin ab, der Umfang an Dokumenten nimmt zu.

3.4.4 Qualität und Zuverlässigkeit

Aus dem oben erwähnten läßt sich schon ersehen, wie ein verteiltes Suchsystem die Güte von Information bewerten kann. Eine Möglichkeit besteht darin, daß der Broker mittels eines lokalen Gatherer eine Beschreibung des Informationsservers 3.4.5 ermittelt und so auf die Qualität des Inhaltes schließt. Es ist auch denkbar, daß ein Broker nicht nur themenspezifisch sondern auch qualitätsspezifisch einer Gruppe angehört.

Ein anderer Weg wäre die Verwendung von Experten, die die Dokumente an ihrem Server beurteilen. Diese Bewertung könnte als Metainformation im Dokument vermerkt und in weiterer Folge als Attribut im SOIF-Objekt aufgenommen werden.

3.4.5 Zusatzinformation über Server und Sites

Die Nähe des Gatherer zur Informationsquelle erlaubt es, auch über diese selbst Informationen zu sammeln und bereitzustellen. Von Interesse sind Eigenschaften wie Betreibername, Qualitätsmaß, thematische Kategorie, Größe, Richtlinien für das Dokumentenlayout und statistische Werte wie zum Beispiel die durchschnittliche Änderungsrate der Dokumente. Diese Zusatzinformationen ermöglichen erst die Zusammenarbeit einzelner Teilsuchsysteme, die anhand dieser Eigenschaften gruppiert werden können.

Die Serverinformationen verbessern auch, unabhängig von Zusammenschlüssen, die Brauchbarkeit der Suchergebnisse. So können zum Beispiel Fehler, die auf Mehrdeutigkeit des Suchbegriffes beruhen, durch Berücksichtigung von thematischen Kategorien vermieden werden. Bei der zweistufigen Ergebnispräsentation (siehe Abschnitt 4.2.4) dient die Serverinformation als Beschreibung und somit als Hilfe für den Benutzer, der entscheiden muß, welche Server genauer durchsucht werden sollen.

3.5 Zusammenfassung

Das Grundkonzept des Harvest-Systems beruht auf der Idee der verteilten Informationsauffindung, Datenaufbereitung und Bereitstellung. Ein Harvest-Gatherer sammelt periodisch Dokumente innerhalb eines definierten Bereiches und extrahiert deren Inhalt. Neben den verschiedenen Zugriffsarten (HTTP, FTP) und lokales Dateisystem unterstützen die Verarbeitungsmodule des Gatherer eine Reihe von Dateiformaten. Ein Teilsystem des Gatherer erzeugt aus dem Dokumentinhalt sogenannte SOIF-Objekte. Diese werden dann zum Zwecke der Bereitstellung über das Netz weitergeleitet.

Der Harvest-Broker ist die Schnittstelle zwischen Gatherer und Index einerseits und zwischen Index und Benutzer andererseits. Er verwaltet die vom Gatherer bereitgestellten SOIF-Objekte und veranlaßt deren Indizierung durch externe Werkzeuge. Dem Benutzer stellt er mit Hilfe eines Web-Clients und eines CGI-Skripts eine Schnittstelle zur Kommunikation mit dem System bereit. Damit verschiedene Indexer mit dem Broker zusammenarbeiten können, ist eine flexible Schnittstelle zwischen Broker und Indexer definiert.

Broker können ihre Daten austauschen. So kann ein kooperierendes verteiltes Suchsystem aufgebaut werden. Einzelne Broker können speziellen Themen zugeordnet werden und ihre Daten an übergeordnetere Broker weiterreichen.

Durch das Konzept der verteilten Suche ergeben sich eine Reihe von Vorteilen, wie zum Beispiel die Reduktion der Netzwerk- und Serverlast, die das Harvest-System im Vergleich zu anderen Suchsystemen auszeichnet. Weiters führen die Nähe des Gatherer zur Informationsquelle und die mögliche Zusammenarbeit einzelner Teilsysteme zu einem vergleichsweise hohen Grad an Effizienz und Aktualität.

Im folgenden Kapitel werden einzelne Module des hier vorgestellten Harvest-Systems im Detail untersucht und daraus Verbesserungs- und Erweiterungsvorschläge abgeleitet.

Kapitel 4

Verbesserungspotential am Harvest-System

Auf den folgenden Seiten werden die entsprechenden Module des Harvest-Systems zum Parsen von HTML-Dokumenten und zum Indexaufbau auf Verbesserungsmöglichkeiten untersucht. Die tatsächliche Umsetzung von Verbesserungen werden im Abschnitt Gestaltungsbereich (Kapitel 5 und 6) ausführlich beschrieben.

4.1 Die HTML-SOIF-Konvertierung

4.1.1 Der HTML-Parser

Das Essence-Subsystem (siehe Abschnitt 3.2.1) des Harvest-Gatherer beinhaltet u.a. einen *Summerizer*, der die Konvertierung der HTML¹-Seiten in SOIF-Objekte durchführt. Bevor auf die Problematik der Konvertierung eingegangen wird, werden zuerst die wesentlichen Eigenheiten von HTML beschrieben.

HTML - Hypertext Markup Language

HTML [Rag] ist eine SGML²-Anwendung [Cov], die mittels einer DTD³ definiert ist [Gro]. Die Markup-Befehle (Tags) werden von spitzen Klammern⁴ umschlossen, um sie vom eigentlichen Text zu unterscheiden. Manche dieser Tags haben Parameter (so genannte Attribute), die in der Form `<TAG ATTR1=VALUE1 ATTR2=VALUE2>` angegeben werden.

Bei Tags und Attributen wird die Groß- und Kleinschreibung nicht berücksichtigt. Bei den Attributwerten hingegen wird manchmal, abhängig vom jeweiligen Attribut, die Schreibweise übernommen. Die Argumente müssen zwischen Anführungszeichen

¹Hyper Text Markup Language

²Standard Generalized Markup Language

³Dokument Typ Definition

⁴Größer-Kleiner-Zeichenpaar (<,>)

eingeschlossen werden, wenn diese Sonderzeichen beinhalten oder die Schreibweise explizit beachtet werden soll.

Einfache oder mehrfache Leerstellen oder Zeilenwechsel wirken (von Spezialfällen wie Text innerhalb des `<PRE>`-Tags abgesehen) jeweils so wie ein Leerzeichen. Somit läßt sich der Text im HTML-Dokument optisch gliedern, ohne das Aussehen des Dokumentes davon beeinflußt wird.

Die meisten HTML-Befehle bestehen aus einem Start-Tag der Form `<TAG>` und einem End-Tag der Form `</TAG>`. Diese Befehlspaare legen jeweils die Bedeutung des dazwischen liegenden Textes fest. So umschließt zum Beispiel das Tag-Paar `<TITLE>` und `</TITLE>` den Titel eines HTML-Dokumentes.

Einzeln auftretende HTML-Befehle (ohne End-Tag) bezeichnen bestimmte Elemente, die an der entsprechenden Stelle eingefügt werden. So bewirkt zum Beispiel `
` einen Zeilenumbruch im HTML-Dokument.

Die paarweisen HTML-Befehle müssen immer richtig geschachtelt werden. Nicht erlaubt sind Verschränkungen. Die gängigen Web-Browser (Netscape, Internet Explorer) tolerieren allerdings solche syntaktisch falschen Definitionen (siehe nächster Absatz). Kommentare können in der Form `<!-- Kommentar -->` eingefügt werden. Damit der Kommentartext tatsächlich von allen Browser-Versionen als Kommentar erkannt wird, sollte er weder `--` noch `>` enthalten.

Anpassung an das Verhalten der Web-Browser

Der HTML-Parser des Gatherer hält sich strikt an die DTD-Spezifikationen von HTML [HSW96]. Die Web-Browser (Netscape, Internet Explorer) verhalten sich wesentlich toleranter. Testläufe am IICM⁵ zeigten zum Teil große Unterschiede zwischen SOIF-Objekt und angezeigten HTML-Text. Hier einige Beispiele des nicht HTML-konformen Verhaltens der Web-Browser:

- Text vor dem `<BODY>`-Tag führt zu keinem Fehler und wird sogar angezeigt.
- Das `<HEAD>`-Tag hat keinen Einfluß. Es wird ignoriert, unabhängig von der Platzierung im Dokument.
- Verschränkte Definitionen werden von den Web-Browser aufgelöst. Un zwar wird bei einem End-Tag die innerste Definition geschlossen, gleichgültig welches Tag diese geöffnet hat. In HTML sind sie nicht erlaubt.
- Kommentare werden wie folgt akzeptiert: `<!--Kommentar-->`
- Steht zwischen den spitzen Klammern ein Tag, das syntaktisch richtig, aber kein HTML-Tag ist, so wird es ignoriert.
- Steht zwischen den spitzen Klammern etwas anderes als ein syntaktisch richtiges Tag, wird der Text mit den Klammern als Text interpretiert und angezeigt.

⁵Institut for Information Processing and Computer Supported New Media

Die Autoren von Web-Seiten überprüfen ihre Dokumente mit Hilfe der Web-Browser. Das Erscheinungsbild am Browser ist somit das entscheidende Kriterium bei der Dokumentenerstellung. Damit Konsistenz zwischen SOIF-Objekt und für den Benutzer sichtbarer Information herrscht, wäre eine Angleichung des HTML-Parsers an die Eigenschaften der Web-Browser notwendig.

Konfigurierbarkeit und Fehlertoleranz

Der Parser kann so konfiguriert werden, daß er einzelne Tag-Inhalte ignoriert [HSW96]. `<CODE>` wäre ein sinnvolles Beispiel dafür. Der Code kommt so nicht in den Volltext. Textteile die innerhalb dieser Definition durch ein anderes Tag, zum Beispiel als Schlüsselwort, gekennzeichnet sind, werden natürlich auch nicht in die Menge der Schlüsselwörter aufgenommen, da der ganze Bereich ignoriert wird. Beinhaltet der Code einen Hyperlink, also ein `<A>`-Tag mit einer Referenz als Attribut, so wird dieser ebenfalls ignoriert. Da der Benutzer allerdings einen Link sieht und ihn benutzen kann, sollte das SOIF-Objekt den Link auch aufnehmen. Daher darf sich das Ignorieren von HTML-Inhalten nicht auf die HTML-Attribute enthaltener HTML-Definitionen auswirken.

Des weiteren kommt es bei fehlerhaften HTML-Dokumenten oft zu unvorhersehbaren Reaktionen des Parsers. Sobald ein Fehler erkannt wird, ist das resultierende SOIF-Objekt unbrauchbar. Wünschenswert wäre hier ein toleranteres Verhalten. So sollten verschränkte Definitionen bearbeitet und nicht in HTML definierte beziehungsweise nicht benötigte Tags ignoriert werden.

4.1.2 Modifikation der SOIF-Attribute

Das SOIF-Format (siehe Abschnitt 3.1) hat keinen festen Attributsatz. Es lassen sich somit neue Attribute für die Generierung der SOIF-Objekte selbst definieren. Auch der *HTML-Summerizer* des Gatherer ist in weiten Bereichen konfigurierbar. So lassen sich nach Abschluß des Parsens und der Erstellung des SOIF-Objektes bestehender SOIF-Attribute im Zuge einer Nachbearbeitung⁶ modifizieren. Zum Beispiel können Überschriften nachträglich sortiert und doppelt vorhandene Schlüsselwörter löschen. Dennoch bleiben diesbezüglich einige Wünsche offen:

- Bestehende SOIF-Attribute lassen sich zwar im Objekt nachbearbeiten, neue Attribute können aber nicht mehr eingefügt werden. So läßt sich zum Beispiel mittels eines externen Moduls ein SOIF-Attribut "Inhaltszusammenfassung" nur dann in das SOIF-Objekt integrieren, wenn bereits ein solches existiert.
- Inhalte von SOIF-Attributen können nicht in Abhängigkeit des Inhaltes eines anderen SOIF-Attributes modifiziert werden. Zum Beispiel wäre es wünschenswert, nur dann die aus dem Text extrahierten Schlüsselwörter zu verwenden, wenn keine durch das `<META>`-Tag gewonnen wurden.

⁶Nachbearbeitung bezeichnet hier die Modifikation der SOIF-Attribute, nachdem diese über den Weg des Parsens aus dem HTML-Quelldokument erstellt wurden.

- Es ist nicht möglich, den Inhalt eines SOIF-Attributes unter Berücksichtigung der HTML-Tags zu gliedern, da die Information der HTML-Formatierung nach dem Parsen nicht mehr zur Verfügung steht. Der Mechanismus einer Gliederung wäre zum Beispiel im Volltextattribut ein Weg, um Absätze und Überschriften zu kennzeichnen.
- Standardwerte können nicht vergeben werden. So kann zum Beispiel beim Auftreten eines `<SCRIPT>`-Tags ohne `Language`-Attribut der Standardwert `JavaScript` im entsprechenden SOIF-Attribut nicht vermerkt werden.
- Da sich die Nachbearbeitung des SOIF-Objektes immer nur auf ein Attribut pro Arbeitsgang bezieht, können keine Abhängigkeiten zwischen zwei oder mehreren Attributen in die Modifikation einfließen und Verschmelzungen von Attributinhalt nicht durchgeführt werden. So wäre es zum Beispiel wünschenswert, nach der Generierung des SOIF-Objektes externe Module entscheiden zu lassen, ob und welche SOIF-Attribute zusammengefügt werden sollen. Diese Möglichkeit ist vor allem dann sinnvoll, wenn der SOIF-Attributname mittels `<Meta>`-Tag des HTML-Codes bestimmt wird (siehe nächsten Abschnitt). Zum Beispiel können Schlüsselwörter sowohl aus dem Textteil extrahiert als auch mittels `<META>`-Tag gewonnen werden. Im Zuge der Nachbearbeitung könnte nun entschieden werden, welche Schlüsselwortmenge ins SOIF-Attribut `keywords` gelangt. So könnte auch die Vereinigungsmenge beider das SOIF-Attribut ergeben.

Metadatengenerierung

Metadaten können über das `<META>`-Tag in ein SOIF-Attribut abgelegt werden. Der *HTML-Summerizer* des Gatherer kann nicht zwischen statischen SOIF-Attributnamen und solchen, die mittels `<Meta>`-Tag erzeugt wurden, unterscheiden. Wäre dies der Fall, könnte der *Summerizer* in der Nachbearbeitungsphase entscheiden, ob diese Metadaten zum Inhalt des gleichlautenden SOIF-Attributes, dessen Inhalt aus dem Text ermittelt wurde, hinzugefügt werden, ob sie dieses ersetzt oder ob sie gelöscht werden. Metadaten und Daten aus dem Inhalt könnten auch, abhängig von der jeweiligen Konfiguration, verknüpft werden. So wäre es dann zum Beispiel machbar, Schlüsselwörter aus dem Text und aus den Metadaten entweder zusammenzulegen oder, abhängig von der Konfiguration, die einen oder die anderen zu verwerfen.

Automatische Zusammenfassung

Eine Kurzzusammenfassung des Dokumentinhaltes ist in erster Linie eine Orientierungshilfe für den Benutzer. Der Harvest generiert keine automatische Zusammenfassung des Inhaltes. Die Integration von Modulen zur Erstellung von Zusammenfassungen ist zwar mit Hilfe der Nachbearbeitung denkbar; brauchbare Ergebnisse erzielt man aufgrund der oben erwähnten Einschränkungen im ursprünglichen Harvest-System kaum.

Folgende neue Wege und deren Kombination sind vorstellbar:

- Verwendung von HTML-Metaattributen (z.B.: `ABSTRACT`, `DESCRIPTION`, etc.)
- Inhalte verschiedener SOIF-Attribute (z.B.: `title`, `headings` etc.)
- Suche nach Absatz im Volltextattribut, der mit “abstract”, “Inhaltsangabe”, “Zusammenfassung” etc. beginnt. Dies ist nur möglich, wenn der Volltext im SOIF-Objekt nach Abschnitten gegliedert ist.
- Wahl des ersten oder des letzten Absatzes.

Die Art und Weise, wie eine Inhaltszusammenfassung erzeugt wird, müßte individuell vom Administrator des Informationsservers bestimmt werden können, da er über die Richtlinien des Dokumentenlayouts in seinem Bereich verfügt. So beinhalten wissenschaftliche Publikationen in der Regel eine Zusammenfassung. Diese könnte dann vom Administrator mittels Konfiguration für den Eintrag im SOIF-Objekt bestimmt werden.

Spracherkennung

Der Name WORLD WIDE WEB impliziert schon Mehrsprachigkeit. Die Sprache ist eines der wesentlichen Merkmale eines Dokumentes. Da von der Möglichkeit, die Sprache mittels `<Meta>`-Tag im HTML-Header anzugeben, selten Gebrauch gemacht wird, sind Spracherkennungsmodule notwendig. Im Gatherer fehlt jedoch eine Schnittstelle zur Spracherkennung. Auch die oben beschriebene Methode der Nachbearbeitung der SOIF-Objekte ist aufgrund der oben beschriebenen Schwachpunkte dafür nicht geeignet. Spracherkennungsmodule müssen auf mehrere Attribute schreibend und lesend zugreifen können. Dies ist bei *HTML-Summerizer*, wie schon erwähnt, nicht machbar.

Ein Spracherkennungsmodul müßte den Inhalt des Volltext-SOIF-Attributes lesen, und das Resultat seiner Untersuchung - die Sprache - in Form einer Kennung in ein entsprechendes SOIF-Attribut ablegen können.

4.2 Der Suchindex

Das Harvest-System verwaltet seinen Suchindex mittels externer Werkzeuge wie zum Beispiel Glimpse (siehe Abschnitt 3.3). Über das Indexinterface können auch andere Indexer an den Broker angeschlossen werden (siehe Abschnitt 3.3.1).

Eine wesentliche Schwäche der Indizierung liegt darin, daß die Suchmaschinen den Inhalt der SOIF-Dateien indizieren, ohne zwischen Metadaten und tatsächlichem Inhalt zu unterscheiden. Attributnamen werden ebenso indiziert wie Einträge, die Lebensdauer, URL etc. beinhalten. Das hat zur Folge, daß zum Beispiel bei einer Suche nach “md5” alle indizierten Dokumente als Ergebnis ausgegeben werden, weil jedes SOIF-Objekt ein Attribut dieses Namens enthält.

Es lassen sich auch keine statistischen Aussagen über die Häufigkeit einzelner Wörter und deren SOIF-Attribut machen. Diese Information ist notwendig, wenn man

häufig vorkommende Wörter ausscheiden und das Suchergebnis anhand einer Gewichtung der Wörter sortieren will.

4.2.1 Keyword-Relevanz-Filter

Ein Keyword-Filter hat die Aufgabe, nicht gewünschte Schlüsselwörter auszuschneiden und so die Menge der indizierten Wörter zu verkleinern und damit die Brauchbarkeit der Schlüsselwörter des Index als Unterscheidungskriterium der Dokumente zu erhöhen [GDN⁺98]. Statisch kann dies mittels Stoppliste geschehen. Dieser Mechanismus ist im Gatherer integriert [HSW96]. Ein Relevanzfilter hingegen mißt die Häufigkeit des Auftretens eines Schlüsselwortes, vergleicht diesen Wert mit einem voreingestellten Schwellwert und entscheidet dann, ob dieses auch tatsächlich aufgenommen wird. Dem Harvest-Systems fehlt ein solches Verfahren.

Der Index unterliegt ständigen Veränderungen und so kann die Worthäufigkeit eines Wortes einmal unterhalb und einmal oberhalb des Grenzwertes liegen. Daher müssen alle Worteinträge mitgeführt werden und entsprechend der Worthäufigkeit ein- und ausgeblendet werden. Da ein Schlüsselwort als Unterscheidungsmerkmal einzelner Dokumente dienen soll, spielt die Häufigkeit des Auftretens innerhalb eines Dokumentes für die Relevanzbetrachtung keine Rolle. Mit steigender Anzahl der Dokumente, in denen das Schlüsselwort enthalten ist, nimmt dessen Brauchbarkeit als Suchkriterium in Form eines Keywords ab. Es kann kaum als Merkmal innerhalb des Informationsraumes⁷ dienen.

Die Wichtigkeit eines Wortes ist zeitabhängig. So ist das Schlüsselwort "Schifahren" im Winter weniger relevant als im Sommer. Sie ist aber auch abhängig vom Grad der Spezialisierung des Suchdienstes. Das Keyword "Medizin" kann für einen allgemeinen Suchdienst relevant sein, für einen speziellen Suchdienst für Mediziner ist es dies nicht. Die Zeitabhängigkeit läßt sich insofern berücksichtigen, als daß bei jedem Update auch die Häufigkeit neu ermittelt wird. Da der Broker in ein hierarchisches System eingebunden sein kann, und sich bei übergeordneten Brokern andere Worthäufigkeiten ergeben, muß er alle Schlüsselwörter verwalten; für sich selbst scheidet er jene aus, deren Häufigkeit einen bestimmten Schwellwert überschreiten.

Die im gefundenen Dokument enthalten relevanten Schlüsselwörter könnten als Zusatzinformation beim Suchergebnis zu jedem Dokument ausgegeben werden (siehe Abschnitt 4.2.3). Es läßt sich aber auch die Suche entsprechend einschränken.

4.2.2 Gewichtung der Suchergebnisse - Ähnlichkeit von Dokument und Suchanfrage

Boolsche Anfragesysteme haben sich vor allem bei Systemen bewährt, die einen eingeschränkten, geschulten Benutzerkreis haben. Benutzer, die ein solches System selten verwenden, haben Schwierigkeiten mit der Syntax und Semantik der Abfragen [Har92].

⁷zum Beispiel innerhalb eines Brokerbereiches

Boolsche Systeme erzielen gute Ergebnisse in bezug auf *Recall*⁸ und *Precision*⁹, wenn die Anfrage exakt formuliert wird.

Folgende Schwächen dieser Systeme werden in [ESM92] erläutert:

- Die boolschen Operatoren sind strikt. Zum Beispiel liefern Suchabfragen der Form **A and B and C and D and E** nur Dokumente, in denen alle Terme enthalten sind. Der Benutzer hat kaum die Möglichkeit, dieses Verhalten “aufzuweichen”¹⁰.
- Das boolsche Modell gibt keine Auskunft über die Wichtigkeit der einzelnen gefundenen Dokumente.
- Das boolsche Modell bietet keine Möglichkeit, einzelne Suchterme nach ihrer Bedeutung zu bewerten.

Das nachfolgend beschriebene Ranking-Verfahren für das Retrieval verzichtet auf boolsche Operatoren und scheinen somit benutzerfreundlicher zu sein [Har92].

Prinzip des Rankings

Worthäufigkeiten und unterschiedliche Formatierungen innerhalb eines Dokuments geben Auskunft über die Bedeutung der einzelnen Wörter im Dokument. Ein Wert, der diese Bedeutung widerspiegelt, wird beim Ranking im Index mitgespeichert. Für jedes Dokument, das zum Suchergebnis einer Anfrage gehört, wird in weiterer Folge ein Wert errechnet, der dessen Wichtigkeit im Gesamtergebnis wiedergibt, und als ein weiteres Kriterium für die Suchergebnispräsentation dient. Dokumente, die das Suchwort im Titel oder mehrmals enthalten, werden höher bewertet als solche, in denen es nur einmal im Textbereich vorkommt.¹¹

Zusätzlich zur Bedeutung eines Suchterms im Dokument wird bei den Ranking-Verfahren auch dessen Bedeutung innerhalb der Suchanfrage selbst berücksichtigt. Statt boolscher Operatoren gibt der Benutzer das relative Gewicht der einzelnen Suchbegriffe an [Har92].

Ein Beispiel:

gegeben seien zwei Dokumente A und B (siehe Abbildung 4.1). Der Benutzer gibt zusätzlich zu jedem Suchterm eine Zahl an, die dessen Bedeutung in der Anfrage repräsentiert. Diese Zahlen werden einzeln mit dem Gewicht des jeweiligen Terms im Dokument multipliziert. Durch Addition dieser Produkte liefert einen Wert, der als Maß für die Wichtigkeit des Dokuments in bezug auf die Suchanfrage dient. Die Anfrage nach ((Zug, 0.7) (Wagen, 0.1)) ergibt so (A, 0.35); (B, 0.22). Das Dokument A ist somit relevanter. Verschiedene Ranking-Algorithmen werden in [Har92] beschrieben.

⁸*Recall* gibt das Verhältnis zwischen gefundenen relevanten und alle relevanten Dokumenten an.

⁹*Precision* gibt das Verhältnis zwischen relevanten gefundenen und alle gefundenen Dokumenten an.

¹⁰Durch Umwandlung in die Form **(A and B and C and D) or (A and B and C and E) or (A and B and D and E) ...** ließe sich dies auf sehr komplizierte Weise erreichen.

¹¹Dieses Verhalten kann von Web-Autoren mißbraucht werden, indem sie Wörter nur zum Zwecke des besseren Ranking oft sinnlos aneinander reihen (*Spamming*).

Dokument A		Dokument B	
Wort	Bedeutung	Wort	Bedeutung
Auto	0.7	Wagen	0.1
Zug	0.5	Zug	0.3
Straße	0.3	Straße	0.9
Anzug	0.1	Haus	0.4

Abbildung 4.1: Zwei Dokumente: Wörter und ihre Bedeutung für das Dokument

Das bisher beschriebene Konzept läßt sich nun um die Verwendung von boolschen Operatoren erweitern. Es stellt sich die Frage, wie sich das Gewicht eines gefundenen Dokumentes aus den Gewichten der Teilterme ermitteln läßt, damit es vergleichbar wird. In [ESM92] wird unter anderem das P-Norm-Modell vorgestellt.

Das P-Norm-Modell

Das P-Norm-Modell [SFW83] [ESM92] erlaubt es, sowohl den Wörtern im Index ein dokumentspezifisches Gewicht zu geben, als auch die einzelnen Termen in der Anfrage, abhängig von ihrer Wichtigkeit, unterschiedlich zu bewerten. Hier werden Abfragen und Dokumente als n -dimensionale Vektoren betrachtet, wobei die Dimension gleich der Anzahl der Suchterme der Anfrage ist. Die Koordinaten entsprechen dem Gewicht der Terme¹². Die Koordinaten liegen im Intervall $[0, 1]$. Ein Dokument, das durch den Punkt $(0, 0, \dots, 0)$ repräsentiert wird, enthält keinen Suchterm. Der Punkt $(1, 1, \dots, 1)$ steht hingegen für ein Dokument, das alle Suchterme mit dem maximalen Gewicht enthält. Somit ist es möglich, gefundene Dokumente bezüglich ihrer Distanz zu diesen beiden Punkten zu reihen. Während die boolsche Konjunktion verlangt, daß alle Terme im Dokument auftreten (Optimum $(1, 1, \dots, 1)$), werden Dokumente in diesem Modell um so besser bewertet, je näher sie am Optimum liegen. Das Vorkommen aller Terme ist daher nicht zwingend. Ein Dokument, das im Resultat einer disjunktiven Suchabfrage enthalten ist, ist im P-Modell dann am wichtigsten, wenn sein Abstand vom Ursprung (kein Suchwort im Dokument) am größten ist, weil mit steigendem Abstand vom Ursprung die Anzahl bzw. die Gewichtung der Wörter eines Dokuments zunimmt.

Es seien nun A_1, A_2, \dots, A_n die Suchwörter und a_1, a_2, \dots, a_n deren, vom Benutzer vergebenen Gewichte. Durch p sei angegeben, wie genau sich der boolsche Operator auf die Berechnung des Gewichtes (siehe Gleichungen 4.6 und 4.5) auswirken soll. Der Parameter p ist ein Maß für die unterschiedliche Bewertung der beiden boolschen Operatoren (\wedge, \vee) bei der Berechnung der Ähnlichkeit und liegt im Intervall $(1 \dots \infty)$. Er wird bei der Konfiguration des Ranking-Systems festgelegt. Suchanfragen (Q) mit $n - 1$ Konjunktionen ($Q_{\wedge p}$) beziehungsweise $n - 1$ Disjunktionen ($Q_{\vee p}$) haben in diesem Modell folgende Form:

$$Q_{\vee p} = (A_1, a_1) \vee_p (A_2, a_2) \vee_p \dots \vee_p (A_n, a_n) \quad (4.1)$$

¹²Suchwörter der Anfrage bzw. Wörter im Index

$$Q_{\wedge p} = (A_1, a_1) \wedge_p (A_2, a_2) \wedge_p \cdots \wedge_p (A_n, a_n) \quad (4.2)$$

Die Negation der boolschen Anfrage (Q) aus den Gleichungen 4.2 und 4.2 wird wie folgt dargestellt:

$$Q_{\neg p} = \neg Q \quad (4.3)$$

Es seien nun weiters $d_{A1}, d_{A2}, \dots, d_{An}$ die Gewichte der Wörter im Index. Die Ähnlichkeit ($SIM(Q, D)$) zwischen Anfragen ($Q_{\wedge p}, Q_{\vee p}, Q_{\neg p}$) und Dokument (D) errechnet sich im P-Modell aus:

$$SIM(Q_{\vee p}, D) = \sqrt[p]{\frac{a_1^p d_{A1}^p + a_2^p d_{A2}^p + \cdots + a_n^p d_{An}^p}{a_1^p + a_2^p + \cdots + a_n^p}} \quad (4.4)$$

$$SIM(Q_{\wedge p}, D) = 1 - \sqrt[p]{\frac{a_1^p (1 - d_{A1})^p + a_2^p (1 - d_{A2})^p + \cdots + a_n^p (1 - d_{An})^p}{a_1^p + a_2^p + \cdots + a_n^p}} \quad (4.5)$$

$$SIM(Q_{\neg p}) = 1 - SIM(Q, D) \quad (4.6)$$

und liegt im Intervall $[0, 1]$.

Der so errechnete Wert für die Ähnlichkeit zwischen einem Dokumenten und einer Anfrage gibt an, wie gut das Dokument zu Anfrage paßt. Durch unterschiedliche Bewertung der Suchterme (a_1, a_2, \dots, a_n) in der Anfrage gehen die einzelnen Koordinaten der Dokumente ($d_{A1}, d_{A2}, \dots, d_{An}$) mehr oder weniger stark in die Distanzberechnung ein. Im Falle, daß alle Suchwortgewichte gleich sind, fallen sie aus den Gleichungen heraus. Je größer p , desto größer wird der Unterschied zwischen Konjunktion und Disjunktion. Für den Fall $p = 1$ ergeben sich, wie man leicht aus den Gleichungen 4.5 4.6 entnehmen kann, zwei idente Gleichungen. Je größer die Ähnlichkeitswerte zwischen gefundenem Dokument und Suchanfrage sind, desto wichtiger ist dieses im Gesamtergebnis.

Beispiel 1:

Die Suchabfrage Q ($(\text{Zug}, 0.4) \wedge (\text{Straße}, 0.3)$) sei nun durch den Vektor $\begin{pmatrix} 0.4 \\ 0.3 \end{pmatrix}$ dargestellt. Es sei weiters $p = 2$. Gesucht wird in den Dokumenten nach Abbildung 4.1. Für die Ähnlichkeiten der Anfrage mit den Dokumenten D_A und D_B ergibt sich nun unter Verwendung der Gleichung 4.6:

$$SIM(Q, D) = 1 - \sqrt[2]{\frac{0.4^2(1 - d_{Zug})^2 + 0.3^2(1 - d_{Straße})^2}{0.3^2 + 0.4^2}}$$

Werden für d_{Zug} und $d_{Straße}$ die entsprechenden Werte aus der Tabelle aus Abbildung 4.1 eingesetzt, ergeben sich schließlich folgende Resultate :

$$SIM(Q, D_A) = 0.5348; SIM(Q, D_B) = 0.7136$$

Dokument D_B wird hier vor D_A gereiht.

Beispiel 2:

Gesucht wird wieder in den Dokumenten nach Abbildung 4.1. Die Suchabfrage Q sieht wie folgt aus: $((\text{Zug}, 0.5) \wedge (\text{Straße}, 0.1))$. $p = 2$. Das Wort "Zug" ist nun in der Abfrage wichtiger als das Wort "Straße". Für die Ähnlichkeiten dieser Abfrage mit den

gefundenen Dokumenten ergeben sich wieder unter Verwendung der Gleichung 4.6:

$$SIM(Q, D) = 1 - \sqrt{\frac{0.5^2(1 - d_{Zug})^2 + 0.1^2(1 - d_{Straße})^2}{0.5^2 + 0.1^2}}$$

und somit

$$SIM(Q, D_A) = 0.7148; SIM(Q, D_B) = 0.6495$$

Dokument D_A ist in diesem Fall wichtiger als D_B .

Mit Hilfe eines Index, der ein Datenbanksystem als Basis hat, ließen sich die für dieses Verfahren notwendigen Daten verwalten. Dies Suchmaschinen des Harvest-Systems sind dazu nicht in der Lage.

Die hier eingeführte Ähnlichkeit zwischen einem Dokumenten und einer boolschen Anfrage gibt Auskunft darüber, wie gut ein Dokument zu einer Anfrage paßt. Der Ähnlichkeitswert kann als Kriterium für die Reihung im Suchergebnis dienen. Gefundene Dokumente, deren Ähnlichkeitswert einen konfigurierbaren Schwellwert unterschreiten, könnten aus dem Gesamtergebnis ausgeschieden werden. Unter Verwendung des P-Modells ist es auch möglich, einzelnen Suchtermen in der Anfrage mehr oder weniger wichtig einzustufen.

4.2.3 Zusatzinformation über Server und Sites

Die verteilte Suche hat unter anderem den wesentlichen Vorteil der Nähe zum Informationsangebot. Im Abschnitt 3.4.5 wurde auch die daraus resultierende Möglichkeit erläutert, Informationen über den Server oder einer Site bereitzustellen. Das Harvest-System macht davon allerdings keinen Gebrauch.

Folgende Zusatzinformationen scheinen nützlich:

Thematische Kategorien und Qualitätsbezeichnung können einerseits als erste Orientierungshilfe für den Benutzer dienen. Andererseits lassen sich Gatherer, die Informationsdiensten der selben Kategorie bzw. Qualität zugeteilt sind, anhand dieser Information gruppieren und zu einem gemeinsamen, umfassenderen Suchdienst zusammenfügen.

Namen der Autoren und Herausgeber geben u.a. Auskunft über Qualität und Seriosität der angebotenen Information.

Kurzbeschreibung des Gesamtinhaltes kann ebenfalls als Orientierungshilfe verwendet werden und in einer zweistufigen Suche (siehe Abschnitt 4.2.4) als Informationsbasis dienen.

Statistische Werte : Zum Beispiel kann die durchschnittliche Lebensdauer der Dokumente die Update-Rate des Suchsystems beeinflussen. Je kürzer die Lebensdauer desto öfter muß der Informationsbereich untersucht werden. Die durchschnittliche Anzahl der "Besucher" kann ein Maß für die Bedeutung des Dokumentes sein.

Der Gesamtindex müßte über einen Mechanismus verfügen, der eine Zuordnung von Dokumenten zu Server (bzw. Sites) erlaubt, damit die Serverzusatzinformationen bei

bekanntem Dokument ermittelt werden kann. Zusätzlich sollten auch in den Serverzusatzinformationen gesucht werden können, was deren Indizierung notwendig machen würde.

4.2.4 Benutzerschnittstelle

Multimediaobjekte und Skripts

Oft ist es aus Sicht des Benutzers wünschenswert, zusätzlich zu den eigentlichen Suchbegriffen, noch andere Kriterien anzugeben. So könnte er das Suchergebnis auf Dokumente mit oder ohne eingebettete Multimedia-Objekte beschränken. Der Gatherer des Harvest-System kann zwar so konfiguriert werden, daß er eingebettete Objekte in einem SOIF-Attribut vermerkt, als Suchkriterium kann es aber nicht ohne weiters dienen, da ja nicht der Inhalt des Attributes sondern allein dessen Auftreten im SOIF-Objekt maßgeblich ist.

Es wäre auch wünschenswert, das Vorhandensein und den Typ eingebetteter Objekte im Suchergebnis darzustellen.

Zweistufige Ergebnisaufbereitung

Ist der Suchterm einer Suchanfrage zu unspezifisch, werden zu viele Dokumente als Ergebnis geliefert. Der Aufwand der Suche steigt während die Brauchbarkeit des Resultats sinkt. Ein möglicher Ausweg wäre es, in solchen Fällen nicht nach einzelne Dokumente, sondern nach Informationsserver zu suchen und deren Namen als Suchergebnis zu liefern. Die Suche könnte somit auf Benutzerwunsch oder automatisch "gröber" ausgelegt werden. In einem zweiten Schritt wäre dann eine genauere Suche innerhalb der ausgewählten Server notwendig, um eine Dokumentenliste als Resultat zu erhalten.

Damit nicht alle Dokumente im Index überprüft werden müssen, kann man sich zuerst auf die indizierte Zusatzinformation des Servers beschränken. Ein Stichwortkatalog oder eine thematische Beschreibung des Informationsangebotes wäre denkbar. Auch wenn dies nicht möglich ist, hat diese Zweistufigkeit ihre Berechtigung. Zwar bliebe der Aufwand unverändert, da in den Dokumenten gesucht werden muß, die Suchergebnisdarstellung wäre aber wesentlich kompakter und strukturierter. Alle gefundenen Dokumente eines Servers beziehungsweise einer Site wäre nur durch einen Eintrag (URL und Kurzbeschreibung des Servers (Site), Anzahl der gefundenen Dokumente) in der Ergebnisdarstellung repräsentiert.

4.3 Zusammenfassung

In diesem Kapitel wurden Verbesserungsmöglichkeiten am Harvest-System im Bereich der HTML-Konvertierung und des Suchindexes aufgezeigt.

Es wurden Verbesserungsvorschläge in Bereich der Konfigurierbarkeit und der Anbindung an externe Module gemacht. Die Fehlertoleranz der HTML-Parsers ist zu

gering sollte, so wurde vorgeschlagen, an das Verhalten der Web-Browser angepaßt werden.

Die Suchindex des Harvest-Systems wird mittels externer Suchmaschinen (Glimpse) bereitgestellt und verwendet. Diese eignen sich nicht für komplexere Aufgaben auf dem Gebiet der Suchabfrage und Suchergebnispräsentation. Einige wünschenswerten Methoden, wie zum Beispiel der Keyword-Relevanz-Filterung und das Ranking des Suchergebnisses, wurden beschrieben.

Die Erkenntnisse dieses Kapitels sind Ausgangspunkte für die Veränderungen am Harvest-System, die im wesentlichen aus zwei Teilen bestehen:

- Ein neuer HTML-nach-SOIF-Konverter und seine Eigenschaften werden in Kapitel 5 beschrieben.
- Ein Suchindex auf Basis relationaler Datenbanken und die sich daraus ergebenden neuen Möglichkeiten für die Suche und Ergebnisdarstellung sind Thema des Kapitels 6.

Teil II

Gestaltungsbereich

Gestaltungsbereich

Der zweite Teil dieser Arbeit setzt sich detailliert mit den Verbesserungsmöglichkeiten des Harvest-Systems auseinander und stellt Lösungen einzelner Problemkreise vor. Die im Kapitel 4 aufgezeigten Schwachstellen des Harvest-Systems bilden die Basis für diesen zweiten Teil.

Zwei wesentliche Systemteile sind von Veränderungen betroffen:

- Die HTML-nach-SOIF-Konvertierung des Harvest-Gatherer wurde neu implementiert. Die in Abschnitt 4.1 beschriebenen Mängel werden so durch den neuen Konverter beseitigt. In Kapitel 5 wird dieses Modul und seine Konfigurationsmöglichkeiten beschrieben.
- Die externe Suchmaschine des Harvest-Broker kann nicht für anspruchsvollere *Retrieval*-Methoden verwendet werden. Ein neuer Suchindex auf Basis einer relationalen Datenbank bietet viele neue Möglichkeiten in Bezug auf Suche und Ergebnispräsentation. Kapitel 6 handelt von diesem Suchindex.

Kapitel 5

Der HTML-nach-SOIF-Konverter

Das Essence-System (Abschnitt 3.2.1) des Harvest-Gatherer (Abschnitt 3.2) beinhaltet u.a. einen *Summerizer*, der die Konvertierung der HTML-Dateien in SOIF¹-Dateien durchführt. Das Ergebnis ist aber in den seltensten Fällen zufriedenstellend, da sich der SGML-Parser des *Summerizers* genau an die DTD²-Spezifikationen hält. Dies steht im Gegensatz zum Verhalten der gängigen Webbrowser. Darüber hinaus kommt es zu Problemen bei fehlerhaften HTML-Texten. In Abschnitt 4.1.1 sind diese Schwierigkeiten im Detail beschrieben.

Gesucht ist also ein fehlertolerantes Modul, dessen Ergebnis so weit wie möglich dem der Webbrowser entspricht. Weitere wesentliche Forderungen sind die hohe Konfigurierbarkeit, die flexible Auswertung der Metadaten, sowie die Aufnahme von eingebetteten Objekten (Multimedia, Java-Applets, Scripts etc.) in das SOIF-Format. Weiters sollen Schnittstellen für Spracherkennungsmodule geschaffen und ein Modul für die Generation von Inhaltszusammenfassungen entworfen werden.

5.1 Zielsetzung und Übersicht

Das SOIF-Objekt beschreibt den Inhalt des Originaldokumentes. Im Fall von HTML wird aus den Markup-Definitionen auf die Bedeutung von Textteilen geschlossen. HTML-Definitionen und HTML-Attributen³ werden SOIF-Attributen zugeordnet. Es fließt aber nicht nur der Inhalt eines Dokumentes in die Beschreibung ein, sondern auch Information, die in den HTML-Attributen enthalten ist. So können zum Beispiel Ziel-URLs, eingebettete Objekte und Metadaten⁴ ebenfalls im SOIF-Objekt aufgenommen werden.

Der hier beschriebene Konverter generiert, wie der ursprüngliche HTML-*Summerizer* des Harvest-Systems, kein vollständiges SOIF-Objekt. Er extrahiert lediglich den Gesamtinhalt einer HTML-Seite und erzeugt entsprechende Attribut-Wert-Paare im SOIF-Format. Übergeordnete Harvest-Module ergänzen dann diese Liste von SOIF-

¹siehe Abschnitt 3.1

²Document Type Definition

³HTML-Attribute werden in Abschnitt 4.1.1 erklärt

⁴mit Hilfe der Attribute des Meta-Tags `<META NAME="Attributname" CONTENT="Attributwert">`

Attributen zu einem vollständigen SOIF-Objekt. Der Konverter besteht aus drei Dateien:

Perl-Skript: Dieses Programm liest eine HTML-Datei, deren Name als Parameter übergeben wurde, parst den HTML-Code durch, ordnet Textteilen ein oder mehrere SOIF-Attribute zu und gibt diese Attribut-Wert-Paare in der SOIF-Syntax über die Standardausgabe aus. Auf die Standardfehlerausgabe werden HTML-Dateiname, Dauer der Konvertierung und Anzahl der geparsten Tags geschrieben. Da der Konverter das HTML-Dokument “one-the-fly” verarbeitet, und der Umweg über die DTD-Spezifikation wegfällt, ist er durchschnittlich um den Faktor 10 schneller als das Original.

Konfigurationsdatei: Sie enthält unter anderem eine Tabelle, die den Inhalt der HTML-Definitionen und die Werte der HTML-Attribute den SOIF-Attributen zuordnet. Die verschiedenen Konfigurationsmöglichkeiten werden im Abschnitt 5.2 beschrieben.

Moduldatei: Die Liste von SOIF-Attributen kann, bevor sie vom Konverter ausgegeben wird, noch verändert werden. Neben Sortierung, Ausschluß und Verknüpfung von Einträgen, können durch die Nachbearbeitung auch neue Attribute erzeugt und eingefügt werden. Die Moduldatei enthält einige der dazu nötigen Prozeduren. Sie kann auch beliebig ergänzt werden.

5.1.1 Konfigurierbarkeit

Die Notwendigkeit einer hohen Konfigurierbarkeit hat mehrere Gründe. Der Administrator soll die Möglichkeit haben, HTML-Tags individuell zu behandeln, neue SOIF-Attribute einzuführen und andere, extern ermittelte, Inhalte in die SOIF-Objekte zu integrieren. Der Konverter soll auch an zukünftige HTML-Versionen angepaßt werden können und als Experimentierumgebung für Zusatzmodule dienen.

HTML-SOIF-Tabelle

In dieser Tabelle werden den HTML-Tags und den HTML-Attributen ein oder mehrere SOIF-Attribute zugeordnet. Zusätzlich zu dieser absoluten Zuordnung kann der Inhalt einer Definition auch der nächstäußeren Definition zugewiesen werden. Der Inhalt bestimmter Definitionen kann auch ignoriert werden. Dieses Ignorieren wirkt sich auf alle Definitionen innerhalb des Bereiches aus; lediglich die darin enthaltenen HTML-Attribute werden untersucht. So bleibt zum Beispiel der Inhalt der Definition `Start`, “Start”, innerhalb eines zu ignorierenden Textbereiches unbeachtet; der Wert des HTML-Attributes, “../index.html” wird hingegen entsprechend der HTML-SOIF-Tabelle SOIF-Attributen zugeordnet, da auch die Web-Browser diesen Link anzeigen und der Benutzer diesen verwenden kann. Zu einer Besonderheit zwang das `<META>`-Tag. Hier werden vom Autor des Dokumentes mit Hilfe des “CONTENT-NAME”-Paares attributbehaftete Werte definiert, die auch als SOIF-Attribut-Wert-Paar verwendet werden können. Somit sind die SOIF-Attribute

dynamisch erweiterbar. Dies kann in der Tabelle durch eine Referenz ausgedrückt werden. Es besteht auch die Möglichkeit, nicht den Inhalt einer Definition, sondern das Auftreten eines HTML-Tags in einem SOIF-Attribut zu erfassen. Dies dient in erster Linie zur Berücksichtigung von HTML-Default-Attributen. So hat zum Beispiel das `<SCRIPT>`-Tag "JAVASCRIPT" als Default-Attribut. Mit diesem neuen Mechanismus kann nun bei Auftreten eines attributlosen `<SCRIPT>`-Tags einem SOIF-Attribut der Wert "JavaScript" zugeordnet werden. Die HTML-SOIF-Tabelle ist Teil der Konfigurationsdatei und wird im Abschnitt 5.2.1 genau beschrieben.

Konfigurationen innerhalb des Perl-Skripts

Einige, besonders kritische Eigenschaften des Konverters werden im Perl-Skript selbst festgelegt. Sie brauchen vom Administrator nicht manipuliert werden. Damit man dennoch das Verhalten, zum Beispiel zu Testzwecken, verändern kann, sind die entsprechenden Konstanten am Beginn des Skripts zusammengestellt.

Folgende Bereiche sind so prinzipiell konfigurierbar, sollten aber im allgemeinen nicht verändert werden:

- Die Pfadkonstante `$HTML2SOIF_HOME` enthält den Namen des Verzeichnisses, in dem alle weiteren Dateien erwartet werden. Ist die Umgebungsvariable `$HARVEST_HOME` gesetzt, hat sie den Wert `$HARVEST_HOME/lib/gatherer/`. Andernfalls wird im aktuellen Verzeichnis gesucht. Dieses Verhalten kann verändert werden, indem die Pfadkonstante `$HTML2SOIF_HOME` editiert wird.
- Separatoren dienen zur Gliederung des Inhaltes im Volltext-SOIF-Attribut. So können zum Beispiel Überschriften (`<H1>...<H6>`), Absatzwechsel (`<P>`) und Zeilenumbrüche zu solchen Trennzeichen führen. Somit bleibt ein Teil der HTML-Formatierung für die spätere Weiterverwendung des SOIF-Objektes erhalten. Welche HTML-Tags wieviele Trennzeichen in welchem Volltext-Attribut hervorrufen, wird in einer eigenen Sektion (Abschnitt 5.2.3) der Konfigurationsdatei (Abschnitt 5.2) bestimmt. Welche Zeichen hierzu dienen, kann im Perl-Skript angegeben werden. Standardmäßig werden Zeilenwechsel (`"\n"`, *Newline*) verwendet.
- Einige Tags sollen erzwingen, daß sämtlicher Inhalt zwischen Start- und End-Tag ignoriert wird. Im Gegensatz zum Ignorieren, wie es mittels der HTML-SOIF-Tabelle (Abschnitt 5.2.1) einstellbar ist, bezieht sich dieses auch auf HTML-Attribute innerhalb der Definition. Würde zum Beispiel der Administrator dem HTML-Tag `<SCRIPT>` das IGNORE-Attribut in der HTML-SOIF-Tabelle zuordnen, blieben zwar die HTML-Tags unberücksichtigt, nicht aber deren HTML-Attribute. So würden also HTML-Linkdefinitionen innerhalb der `<SCRIPT>`-Definition als Links des HTML-Dokumentes verarbeitet. Daher überspringt der Parser sämtlichen Inhalt zwischen Start- und End-Tag der `<SCRIPT>`-Definition. Welche Tags ein solches Verhalten des Parsers hervorrufen sollen, kann im Perl-Skript im Abschnitt "IGNORE" bestimmt werden (z.B. Kommentarauszeichnungen).

- In HTML-Dokumenten können Zeichen auf unterschiedliche Weise angegeben werden. Zum Beispiel läßt sich jedes Zeichen mittels der Escape-Sequenz “&#...;” durch seinen ASCII-Code angeben. Dies kann vor allem bei Leerzeichen, die als Trennzeichen zwischen den Worten dienen, zu Problemen führen. Wie der Konverter diesen begegnet, und welche Konfigurationsmöglichkeiten bestehen, wird in Abschnitt 5.1.4 gezeigt.

Nachbearbeitung durch Filter

Nachdem der HTML-Text geparkt wurde, kann eine Nachbearbeitung der Inhalte einzelner SOIF-Attribute durchgeführt werden (Entfernen von Duplikaten, Sortieren etc.). Attribute können aber auch zusammengelegt, gelöscht oder neu eingefügt werden. So können zum Beispiel eine Zusammenfassung und das Erstelldatum in das SOIF-Objekt integriert und die Metadaten mit den entsprechenden Daten aus dem Dokumentinhalt verschmolzen werden. Die Filter, die dies bewerkstelligen, haben Zugriff auf beliebige Attribute und deren Inhalt. Die Filterfunktionen werden in der Konfigurationsdatei als Perl-Skript angegeben. Diese können komplexere Perl-Funktionen beinhalten, die dann beim Programmstart aus ein oder mehreren separaten Dateien eingebunden werden müssen.⁵ Innerhalb dieser Perl-Funktionen können beliebige andere Programme aufgerufen werden. Die Parameter der Filter sind die Namen der SOIF-Attribute, die für die Erstellung des Resultats benötigt werden. Das Resultat ist ebenfalls ein SOIF-Attribut. Um diese Aufgabe zu erleichtern, die Darstellung zu vereinfachen und das Lösen von komplexeren Bearbeitungen zu ermöglichen, können Module in den Konverter geladen werden, die umfangreichere Filterfunktionen als Prozedur beinhalten. Statt eines mehrzeiligen Perl-Skripts braucht so nur der Name der Filterfunktion samt Parameter angegeben werden (siehe Abbildung 5.2).

Wie Filter in den Konverter integriert werden können, wird in Abschnitt 5.2.2 beschrieben. Im Abschnitt 5.3 werden einzelne Filter und ihre Funktion vorgestellt.

5.1.2 Automatische Zusammenfassung, Spracherkennung und Metadaten

Mit den mächtigen Möglichkeiten der Nachbearbeitung der SOIF-Attribut-Liste lassen sich auch neue Attribute erzeugen und einfügen. Er eignet sich auch als Schnittstelle zur Spracherkennung und Generierung von Inhaltszusammenfassungen. Entsprechende Module werden in Abschnitt 5.3 vorgestellt.

Durch vorübergehend getrenntes Behandeln von Textdaten und Metadaten lassen sich diese nach dem Parsen des Originaldokumentes individuell weiterverarbeiten. Sie können umbenannt, mit anderen Attributen verschmolzen oder gelöscht werden. Nachdem die Nachbearbeitung abgeschlossen ist, haben Metadaten Priorität gegenüber den extrahierten Daten. Haben daher unmittelbar vor der Ausgabe der SOIF-Attributliste zwei Attribute den gleichen Namen - ein Attribut wurde über die Metadaten, das andere über die Textdaten generiert - so wird der Inhalt des Textdatenattribut verwor-

⁵siehe Abschnitt 5.2.2

fen. Wurden zum Beispiel beim Parsen Schlüsselwörter aus dem HTML-Text und aus dem `<META>`-Tag gewonnen, kann nun, abhängig von den Konfigurationseinträgen, beide Schlüsselwortmengen zusammengefügt oder aber nur eine Menge weiterverwendet werden. Steht bezüglich der Schlüsselwörter kein Eintrag in der Konfigurationstabelle, werden die Schlüsselwörter des `<META>`-Tags in das entsprechende SOIF-Attribut übernommen. Die anderen Schlüsselwörter gehen verloren.

5.1.3 Fehlertoleranz

Die Ausgangsmotivation der Fehlertoleranz ist dadurch bestimmt, daß sich das Verhalten des Parsers großteils mit jenem der gängigen Webbrowser decken soll. Der hier vorgestellte Parser des Konverters ist insofern fehlertolerant, als er syntaktisch falsche Tags und Attribute ignoriert, verschränkte Definitionen akzeptiert und sich nur um Definitionen kümmert, die in der Konfigurationsdatei angegeben sind. So kann zum Beispiel das `<HEAD>`-Tag an beliebiger Stelle vorkommen - es wird ignoriert, wenn es in der Konfigurationsdatei nicht eingetragen wurde. Das `<TITLE>`-Tag braucht andererseits nicht in einer `<HEAD>`-Definition stehen, um korrekt ausgewertet zu werden.

- Text vor dem `<BODY>`-Tag wird von den Browsern angezeigt. Daher kann dieser auch vom Konverter in ein, in der Konfigurationstabelle anzugebendes SOIF-Attribut geschrieben werden. Somit kann auch ein beliebiger Text korrekt in eine SOIF-Datei umgewandelt werden, sofern er keine HTML-Tags enthält.
- Steht zwischen den spitzen Klammern ein Tag, das syntaktisch richtig ist, nicht aber in der Tabelle steht, so wird es ignoriert. Der bis zum nächsten HTML-Tag folgende Text wird der letzten offenen HTML-Definition zugewiesen.
- Steht zwischen den spitzen Klammern etwas anderes als ein syntaktisch richtiges Tag, werden die Klammern samt Inhalt als Text interpretiert.
- Der Inhalt von Kommentaren wird solange überlesen, bis das entsprechende End-Tag gelesen wird.
- Verschränkte Definitionen werden strenger als von den Webbrowsern behandelt. Dies soll anhand des nachfolgenden Beispiels erläutert werden:

`<TAG1>text1<TAG2>text2</TAG1>text3</TAG2>`

Während Netscape und Internet Explorer nach dem Lesen von "`</TAG1>`" die innere Klammer in den meisten Fällen schließen, schließt der Parser das passende Tag und alle darin enthaltenen Definitionen. Damit wird also auch "`<TAG2>`" geschlossen. Trifft der Parser schließlich auf "`</TAG2>`", so wird dieses End-Tag ignoriert. Dieses Verhalten ist sicherer und führt zu weniger ausgezeichnetem Text, da so vergessene Definitionen automatisch geschlossen werden.

5.1.4 Konvertierung nach ISO8859-1

Die HTML-Spezifikation (siehe Abschnitt 4.1) legt den ISO8859-1-Zeichensatz für die Erstellung von HTML-Dokumenten fest. Jedes Zeichen - zum Beispiel "ß" - kann auch

durch seinen Namen “ß” oder seinen dezimalen Code “ß” im HTML-Dokument repräsentiert werden, damit auch Zeichen, die nicht über die Tastatur erreichbar sind, in ein Dokument aufgenommen werden können.⁶ Diese verschiedenartigen Schreibweisen führen zu zwei Problemen beim Parsen und beim Indizieren der HTML-Dokumente:

1. *Whitespaces* (Leerzeichen, Tabulatoren etc.), die als Trennzeichen zwischen Wörtern dienen, können, falls sie in der oben gezeigten Namens- oder Code-Form vorliegen, nicht erkannt werden.
2. Unterschiedliche Schreibweisen führen dazu, daß gleiche Wörter nicht als solche erkannt werden. Dies bedeutet in weiterer Folge, daß im Index mehrere Einträge für das gleiche Wort angelegt werden.

Der Konverter wandelt standardmäßig nur *Whitespaces* beliebiger Form in das entsprechende Zeichen um, damit einzelne Wörter erkannt werden. Sollen dies für alle Zeichen (z.B. “ß”, “Ü”) gelten, muß im Perl-Skript der Eintrag `$charkonvert=wspaces` durch `$charkonvert=all` ersetzt werden.

5.2 Die Konfigurationsdatei

Die Konfigurationsdatei ist in vier Abschnitte (HTML-SOIF-Tabelle, Module, Filter und Separatoren) gegliedert. Jeder Abschnitt beginnt mit seinem Namen in eckigen Klammern. Der Abschnitt endet mit einer Leerzeile. Kommentare werden durch “#” gekennzeichnet.

5.2.1 Die HTML-SOIF-Tabelle

Dieser Bereich wird mit “[SOIF]” überschrieben. Hier wird angegeben, welche HTML-Definitionen wie extrahiert und unter welchen Namen die Inhalte zusammengefaßt werden sollen. Die HTML-Muster stehen großgeschrieben in spitzen Klammern am Beginn einer Zeile. Nach einem Trennzeichen (Tabulator) folgen die zugeordneten SOIF-Attribute, die untereinander durch ein Leerzeichen getrennt sind. Die Tabelle hat somit zwei Spalten, Quell- und Zielspalte.

Die folgenden HTML-Muster bezeichnen HTML-Markups im Quelldokument:

<TAG> bezeichnet den Inhalt einer HTML-Definition. Beispiel:

```
<H1>Ueberschrift</H1>
```

<TAG:ATTRIBUTE>: HTML-Definitionen können auch mit Attributen versehen sein. Mit diesem Muster wird der Wert eines HTML-Attributes beschrieben.

```
<A HREF="http://www.tu-graz.ac.at/">
```

Dieses Muster steht für den Wert “http://www.tu-graz.ac.at/” der angegebenen Tag-Attribute-Kombination “A HREF”

⁶ISO SGML Specification <http://www.w3.org/TR/references.html>


```

[SOIF]
#Text innerhalb der Definitionen werden Attribute zugeordnet.
<BODY>          body
<NOTAG>         body
<TITLE>         title

#Text innerhalb der Definitionen und der nächstäußeren Definition werden Attributen
zugeordnet.
<H1>            headings PARENT
<A>             keywords PARENT
<B>             keywords PARENT

#Der Inhalt des HTML-Attributes wird einem SOIF-Attribut zugeordnet.
<A:HREF>        url-references

#Text innerhalb von <CODE> und </CODE> werden ignoriert.
<CODE>          IGNORE

#Defaultattribut: auf das SOIF-Attribut script wird der Wert JavaScript geschrieben.
<SCRIPT:*>      script=JavaScript

#Referenz: Hier wird dem Attribut CONTENT ein SOIF-Attribut zugewiesen, dessen Name
der Wert von NAME ist.
<META:CONTENT>  &NAME

```

Abbildung 5.1: Konfigurationsdatei: Ausschnitt SOIF-Tabelle

- Referenz
`<meta name="keywords" content="Wald">` → `keywords{4}: Wald`
- Das Schlüsselwort PARENT
`<body><h1>Der Wagen</h1> Finster war es...` → `headings{9}: Der Wagen`
`body{...}: Der Wagen`
`Finster war es...`
- Das Schlüsselwort IGNORE
`<body><code>if x=1...Willkommen</code>Hallo</body>` → `url-references{12}: welcome.html`
`body{5}: Hallo`
- Das Schlüsselwort <NOTAG>
`Finster <body>war es...` → `body{...}: Finster war es...`
- Fixe Zuweisung
`<script>...</script>` → `script{10}: JavaScript`

```

[MODULE]
#Namen der Perl-Module, die inkludiert werden sollen.
standard.pl
#Die Funktionen unify, separate und makeauthor sind im Modul standard.pl ent-
halten.
[FILTER]
#Doppelte Überschriften werden gelöscht. Die Prozedur "unify" ist im Modul
"standard.pl" enthalten.
headings          "unify \@{X1}"

#Leerzeichen werden in der URL ersetzt, Labels gelöscht.
pictures          "foreach (@{X1}) {s/ /%20/g; s/#.*//;}"

#Jedes Wort innerhalb eines Listenelementes wird zu einem eigenen Listenelement. An-
schließend können doppelte Elemente gelöscht werden. Die Prozedur "separate" ist im
Modul "standard.pl" enthalten.
keywords          "separate \@{X1}; unify \@{X1}"

#Die Inhalte zweier SOIF-Attribute werden verschmolzen. Anschließend wird das erste
Attribut gelöscht.
METAkeywords keywords "push(@{X2},@{X1}); delete X1"

#Falls es noch keine Autoreneintrag durch einen Meta-Tag gibt, wird eine Prozedur
gestartet, die einen solchen generiert. Anschließend befindet sich in beiden Fällen das Er-
gebnis im Attribut "author" ! Die Prozedur "makeauthor" ist im Modul "standard.pl"
enthalten.
METAauthor author "if (! exists X1) makeauthor \@{X2}"

```

Abbildung 5.2: Konfigurationsdatei: Filter und Module

5.2.2 Module und Filter

In der Sektion der Konfigurationsdatei, die durch "[MODULE]" gekennzeichnet ist, werden die Namen der Perlskripts angegeben, die als Module für die Nachbehandlung geladen werden sollen. Die Dateien müssen sich im Verzeichnis `$HARVEST_HOME/lib/gatherer/` befinden, falls der Konverter des Harvest-Gatherer verwendet werden soll.⁷

Der Abschnitt, der mit "[FILTER]" überschrieben ist, betrifft die Nachbearbeitung der SOIF-Attribut-Liste. Die Filter-Tabelle besteht aus zwei Spalten. Um die Lesbarkeit zu erhöhen, werden zuerst alle SOIF-Attribute angeführt (durch Leerzeichen getrennt), auf die der Filter zugreift. Falls ein Attribut mit dem angegebenen Namen nicht existiert, wird es angelegt, wenn der Filter dieses beschreibt. Nach dem Trennzeichen (Tabulator) steht das zu exekutierende Perlskript unter Anführungszeichen. Die SOIF-Attribute werden innerhalb der Anführungszeichen entsprechend ihres Auftretens in der rechten Spalte mit "X1", "X2" etc. bezeichnet. Zu beachten ist, daß der Kontext der Variablen im Perlskript angegeben werden muß (zum Beispiel "@{X1}" bei

⁷Die Datei **standard.pl** enthält einige Filterfunktionen.

[SEPARATOR]	
#Name des SOIF-Volltext-Attributes, das gegliedert werden soll.	
body	
#Vor der Überschrift werden zwei Trennzeichen ins Volltext-Attribut geschrieben.	
<H1>	2
#Nach der Überschrift wird ein Trennzeichen ins Volltext-Attribut geschrieben.	
</H1>	1
#Bei Auftreten einer horizontalen Linie wird ein Trennzeichen ins Volltext-Attribut geschrieben.	
<HR>	1
#Vor Beginn jedes Absatzes wird ein Trennzeichen ins Volltext-Attribut geschrieben.	
<P>	1

Abbildung 5.3: Konfigurationsdatei: Separatoreinträge

Listenoperationen).⁸

Werden Namen von SOIF-Attributen mittels Referenz zur Laufzeit generiert (zum Beispiel mit der **<Meta>**-Definition), wird ihnen vom Konverter der Präfix “**META**” vorangestellt, damit gleichbenannte Attribute nicht überschrieben werden. Die Filter können also in diesem Fall mit Hilfe dieses erweiterten Namens auf das Meta-Attribut zugreifen. Die Konfigurationsdatei in Abbildung 5.2 behandelt zum Beispiel Daten in den SOIF-Attributen **keywords** und **METAkeywords**. Der Inhalt von **METAkeywords** wurde aus einem **<META>**-Tag gewonnen, dessen **NAME**-Attribut den Inhalt “**keywords**” hat. Hätte der Konverter nicht zwischen denn zwei Arten von “**keywords**” unterschieden, könnte anschließend nicht auf die beiden Inhalte getrennt eingegangen werden. In der Konfigurationsdatei in Abbildung 5.2 werden die Schlüsselwörter des **<Meta>**-Tags dem SOIF-Attribut **keywords** hinzugefügt (**push(@{X2},@{X1})**). Nach Abschluß der Nachbearbeitung wird der Präfix des Attributnamens vom Konverter wieder entfernt. Andere gleichnamige Attribute werden zu diesem Zeitpunkt gelöscht. Metadatenattribute haben Vorrang vor Daten, die aus dem Text extrahiert wurden. Um diese Daten zu erhalten, muß das Meta-Attribut gleichen Namens entfernt werden. Der Inhalt kann natürlich vorher im Zuge der Nachbearbeitung beliebig manipuliert und verschoben werden. Damit der Inhalt von **keywords** nicht verloren geht, wurde in Abbildung 5.2 **METAkeywords** entfernt.

Um neue Filter entwickeln zu können, ist die Kenntnis der internen Struktur der SOIF-Attribute notwendig. Diese wird im Abschnitt 5.3 erklärt.

5.2.3 Separatoren

Einige Tags sollen innerhalb des SOIF-Volltext-Attributes⁹ Trennzeichen erzwingen. In dieser Sektion werden, nach der Bezeichnung “[SEPARATOR]” und einer Zeile mit

⁸Da Perl keine Variablendeklaration braucht, ist bei der Parameterübergabe an eine Prozedur eine Kontextangabe notwendig. Sie gibt Auskunft über den Typ des Parameters.

⁹In der Regel wird es als “**body**” bezeichnet.

dem Namen des Volltext-Attributes, die Tags (auch End-Tags) in spitzen Klammern angegeben, die zu einem solchen Verhalten führen sollen. Nach dem Tabulator folgt die Anzahl der Trennzeichen im SOIF-Volltext-Attribut. Welche Zeichen oder Zeichenkette zur Trennung vorgesehen sind, wird im Skript des Konverters selbst bestimmt. Standardmäßig werden Zeilenwechsel (“\n”, *Newline*) verwendet.

Mit diesem Mechanismus läßt sich der Inhalt genau eines SOIF-Attributes gliedern. So können zum Beispiel Absatzmarken (“<P>”, “</P>”), Zeilenumbrüche (“
”) und Überschriften (“<H1>”... “<H6>”, “</H1>”... “</H6>”) durch eine unterschiedliche Anzahl von Trennzeichen im Volltext-Attribut repräsentiert werden. Diese Gliederung machen sich dann Filtermodule zu Nutze.

5.3 Filtermodule

Um neue Filter zu entwerfen und diese in den Konverter zu integrieren, bedarf es neben einiger Perl-Kenntnisse auch des Wissens um die interne Datenstruktur der SOIF-Attribute. Die Inhalte der SOIF-Attribute werden als Listen von Zeichenketten gespeichert. Jedesmal, wenn der Parser während der Abarbeitung eines HTML-Dokumentes ein SOIF-Attribut neu beschreibt, weil eine weitere korrespondierende HTML-Definition im Text gefunden wurde, wird ein neues Listenelement angelegt. Wird zum Beispiel jedes Wort einzeln als “fett” markiert, wird auch jedes Wort in jedem direkt zugeordneten SOIF-Attribut in ein eigenes Listenelement geschrieben.¹⁰ Befinden sich hingegen mehrere Wörter innerhalb einer Definition, werden sie gemeinsam in einem Listenelement abgelegt. Wird hingegen ein Textteil mittels “PARENT”-Schlüsselwort an das nächstäußere SOIF-Attribut übergeben, wird es in diesem ans Ende des letzten Listenelementes angehängt. Jedes Listenelement wird im SOIF-Objekt durch einen Zeilenumbruch beendet.

Warum wird nun nicht einheitlich jedes Wort entweder an das Ende einer Zeichenkette oder als eigener Eintrag an eine Liste angehängt ?

Die Einträge unterschiedlicher SOIF-Attribute haben unterschiedliche Eigenschaften. So beinhaltet das Attribut “**keywords**” eine Liste von Wörtern, das Attribut “**headings**” eine Liste von Sätzen. Würde jedes Wort als Listeneintrag behandelt oder an eine Zeichenkette angehängt werden, wäre der Satz nicht mehr als solcher vorhanden. Daher wird jeder durch Start- und End-Tag markierte Textteil in jedem direkt zugeordnetem SOIF-Attribut als neuer Listeneintrag angelegt. Wird aufgrund des “PARENT”-Schlüsselwortes ein SOIF-Attribut beschrieben, wird der Text nur an den letzten Eintrag angehängt, damit ebenfalls die Satzstruktur nicht verloren geht. Daten aus den HTML-Attributen werden jeweils getrennt in Listeneinträgen gespeichert.

Erläuternde Beispiele

Die folgenden Beispiele sollen diesen Sachverhalt verdeutlichen. Berücksichtigung findet hierbei wieder die Tabelle aus Abbildung 5.1. Da die interne Struktur der SOIF-

¹⁰In HTML werden Textteile mit dem -Tag markiert, wenn sie “fett” dargestellt werden sollen. In der Regel wird diesem Tag das SOIF-Attribut “**keywords**” zugeordnet.

Attribute im SOIF-Objekt nicht mehr erkennbar ist, werden die Inhalte einzelner Listen zur Veranschaulichung umrandet dargestellt.

- Satzstruktur bleibt erhalten

```
<H1>Dies ist ein Satz</H1><H1>Dies ist ein weiterer Satz</H1>
```

→

```
headings{...}: Dies ist ein Satz
               Dies ist ein weiterer Satz
```

- Das “PARENT”-Schlüsselwort

```
<BODY>Dies ist <B>ein Satz</B></BODY>
```

→

```
body{...}: Dies ist ein Satz
keywords{...}: ein Satz
```

- <NOTAG> und <BODY>

Die SOIF-Tabelle aus Abbildung 5.1 ordnet sowohl dem <NOTAG>- als auch dem <BODY>-Tag das SOIF-Attribut “body” zu. Daher stehen Textteil vor dem <BODY>-Tag in einem anderen Listenelement als diejenigen danach. Im SOIF-Attribut sind diese Teil schließlich durch einen Zeilenumbruch getrennt.

```
Der <body>Wagen</body>
```

→

```
body{...}: Der
           Wagen
```

Da die Web-Browser in solchen Fällen aber keine Trennzeichen anzeigen, ist es sinnvoller, den gesamten Volltext in ein Listenelement zu schreiben. Dies gelingt, indem man sich die im vorigem Beispiel beschriebene Eigenschaft des “PARENT”-Schlüsselwortes zu Nutze macht. Für das folgende Beispiel gilt die SOIF-Tabelle aus Abbildung 5.4:

```
Der <body>Wagen</body>
```

→

```
body{...}: Der Wagen
```

- HTML-Attribute

```
<a href="1.htm">I</a>
<a href="2.htm">II</a>
```

→

```
keywords{...}: I
               II
url-references{...}: 1.html
                   2.html
```

5.3.1 Allgemeine Nachbearbeitung

Wie eingangs erwähnt, bedarf es bei verschiedenen SOIF-Attributen einer individuellen Nachbehandlung der Inhalte. Soll ein Attribut sich nur auf Wörter beziehen, müssen die Wörter innerhalb eines Listenelementes in einzelne Listeneinträge zerteilt werden. Anschließend können diese Wortlisten sortiert und mehrfach vorkommende Einträge gelöscht werden. Neben allen Perl-Funktionen stehen folgende Filter mittels Einbindung von `standard.pl` zur Verfügung:

Trennen: Dieser Filter verteilt Wörter innerhalb eines Listenelementes in einzelne auf. Dies ist notwendig, um zum Beispiel Schlüsselwörter sortieren und doppelt vorhandene Wörter löschen zu können. Beispiel:

```
keywords "separate \@{X1}"
```

Sortieren: Dieser Filter sortiert die Einträge im Attribut alphabetisch. Unter Umständen ist es notwendig, die Wörter innerhalb eines Listenelementes vorher aufzutrennen. Im Beispiel ist dies schon passiert:

```
keywords "sortbyname \@{X1}"
```

Löschen mehrfacher Einträge: Bei manchen Attributen spielt die Häufigkeit des Auftretens eines Wortes keine Rolle. Mehrfach vorkommende Einträge können gelöscht werden. Dieser Filter sortiert und löscht die mehrfachen Vorkommnisse. Auch hier gilt, daß bei manchen Attributen vorher eine Trennung in Wörter durchgeführt werden muß. Im folgenden Beispiel war dies nicht nötig, da die Einträge im Attribut “picture” ja nur als Ganzes eine Bedeutung haben.

```
pictures "unify \@{X1}"
```

RFC1738: Einige Zeichen, wie zum Beispiel Leer- und Fragezeichen dürfen nicht in URLs vorkommen. Um diese unsichere Zeichen zu vermeiden, können diese nach RFC1738 [BMM94] konvertiert werden. So wird zum Beispiel das Leezeichen in “%20” umgewandelt. Im folgenden Beispiel wird der Inhalt des SOIF-Attributes “url-references” nach diesen Zeichen untersucht und unsichere Zeichen in ihre Entsprechung laut RFC1738 übersetzt:

```
url-references "enc1738 \@{X1}"
```

5.3.2 Automatische Zusammenfassung

Neben der Möglichkeit, aus den Metadaten¹¹ eine Inhaltszusammenfassung zu gewinnen, kann auch aus dem Inhalt eine solche extrahiert werden. Unterschiedliche Filtermodule verwenden unterschiedliche Strategien. Die folgenden Perl-Prozeduren sind in der Moduldatei `standard.pl` enthalten und können in der Konfigurationsdatei angegeben werden:

Absatz mit Überschrift: Dieser Filter sucht im Volltextattribut nach einen Absatz, dessen Überschrift im Übergabeparameter (Liste) enthalten ist.

```
body abstract "abs_head(\@{X2},\@{X1},'Kurzfassung','abstract')"
```

Absatz ohne Überschrift: Dieser Filter sucht im Volltextattribut nach einen Absatz, in dem mindestens ein Wort des Übergabeparameter (Liste) enthalten ist.

```
body abstract "abs_text(\@{X2},\@{X1},'Kurzfassung','abstract')"
```

Bestimmter Absatz: Dieser Filter sucht im Volltextattribut nach einen Absatz, dessen Reihung im Übergabeparameter angegeben ist.

```
body abstract "abs_numb(\@{X2},\@{X1},3)"
```

In der Konfigurationsdatei des Konverters lassen sich nun beliebige, bedingte Kombinationen dieser Filter angeben. Darüber hinaus lassen sich Inhalte beliebiger SOIF-Attribute zur Generierung der Zusammenfassung heranziehen.

¹¹z.B.: `<META NAME="description" CONTENT="In diesem Dokument wird ...">`

Erklärendes Beispiel

Das folgende Beispiel zeigt, wie verschiedene Methoden kombiniert werden können. Hier werden die einzelnen Verfahren nach der Reihe ausgeführt, bis eines erfolgreich war. Somit läßt sich eine Prioritätenliste verwirklichen. Im Beispiel hat das Meta-Datum die höchste Priorität.

Existiert das SOIF-Attribut `METAabstract`, enthält es eine Zusammenfassung. Das SOIF-Attribut `abstract`, das durch ein anderes Verfahren beschrieben wurde, wird gelöscht, der Inhalt von `METAabstract` auf `abstract` kopiert. Damit am Ende der Nachbearbeitung der Inhalt von `METAabstract` nicht neuerlich auf `abstract` kopiert wird, wird es entfernt:

```
METAabstract abstract "(exists \@X1) &&
{delete X2; push(@{X2},@{X1}); delete X1}"
```

Falls `abstract` nicht existiert, wird nun versucht, mit dem Filter `abs_head` eine Zusammenfassung zu generieren:

```
body abstract "(!exists \@X2) && abs_head(\@{X2},\@{X1},'abstract')"
```

Falls `abstract` immer noch nicht existiert, wird versucht, mit dem Filter `abs_text` eine Zusammenfassung zu generieren:

```
body abstract "(!exists \@X2) && abs_text(\@{X2},\@{X1},'abstract')"
```

Falls es immer noch keine Zusammenfassung gibt, werden schließlich alle Überschriften in das SOIF-Attribut `abstract` kopiert:

```
abstract headings "(!exists \@X1) && push(@{X1},@{X2})"
```

5.4 Installation

Voraussetzung für die Inbetriebnahme des Konverters ist ein installierter Perl-Interpreter der Version 5.0 oder höher. Unter einer UNIX-Shell läßt sich dann der Konverter wie eine ausführbare Datei starten. Perlskript und Konfigurationsdatei müssen sich im selben Verzeichnis befinden.

Um den Konverter in den Harvest-Gatherer zu integrieren und somit den *HTML-Summerizer* zu ersetzen, muß der gesamte Konverter in das Verzeichnis `$HARVEST_HOME/lib/gatherer/` kopiert werden. Hier befindet sich die Datei `HTML.sum`. In ihr muß nun der Eintrag

```
exec SGML.sum HTML $ *
```

durch

```
exec html2soif.pl $ *
```

ersetzt werden. Von nun an verwendet der Gatherer den neuen HTML-SOIF-Konverter.

[SOIF]		
<NOTAG>	body	
<BODY>	PARENT	
<TITLE>	title keywords	
<H1>	headings PARENT	
<H2>	headings PARENT	
<H3>	headings PARENT	
	keywords PARENT	
	keywords PARENT	
	keywords PARENT	
<CODE>	IGNORE	
<A:HREF>	url-references	
<LINK:HREF>	url-references	
<FRAME:SRC>	url-references	
<IMG:SRC>	pictures	
<META:CONTENT>	&NAME &HTTP-EQUIV	
<APPLET>	IGNORE	
<APPLET:CODE>	applets	
<SCRIPT:*>	scripts=JavaScript	
<SCRIPT:LANGUAGE>	scripts	
<STYLE>	IGNORE	
<MAP>	IGNORE	
<OBJECT>	IGNORE	
<OBJECT:CODE>	activeX	
[MODULE]		
standard.pl		
[FILTER]		
METAschlagwort keywords	"push(@{X2},@{X1}); delete X1"	
headings	"unify \@{X1}"	
url-references	"foreach (@{X1}) {s/ /%20/g; s/#.*//;}; unify \@{X1}"	
pictures	"foreach (@{X1}) {s/ /%20/g; s/#.*//;}; unify \@{X1}"	
keywords	"separate \@{X1}; unify \@{X1}"	
METAauthor author	"if (! exists X1) makeauthor \@{X2}"	
[SEPARATOR]		
body		
<H1>	2	
</H1>	1	
<H2>	2	
</H2>	1	
<H3>	2	
</H3>	1	
<H4>	2	
</H4>	1	
<P>	1	
</P>	1	
<HR>	1	
<DIV>	1	
	1	
<TABLE>	1	

Abbildung 5.4: Beispiel einer kompletten Konfigurationsdatei

5.5 Zusammenfassung

In diesem Kapitel wurde ein neuer HTML-nach-SOIF-Konverter vorgestellt. Neben der Fehlertoleranz, die sich an das Verhalten gängiger Web-Browser orientiert, ist seine Konfigurierbarkeit die wesentliche Verbesserung gegenüber dem ursprünglichen Konverter des Harvest-Gatherer.

Die konfigurierbare Zuordnung der HTML-Tags und -Attribute wurde ebenso überarbeitet wie der Mechanismus der Nachbearbeitung. Dieser ermöglicht unter anderem nun auch Schnittstellen zu externen Modulen, wie zum Beispiel Spracherkennungssysteme und Module zur automatischen Generierung von Inhaltszusammenfassungen.

Durch den neuen Konverter wurde ein Teil des Gatherer ersetzt. Im folgenden Kapitel wird nun ein neuer Suchindex, ein Teilsystem des Broker, beschrieben.

Kapitel 6

Der Suchindex

Der Broker des Harvest-Systems beinhaltet keine eigene Suchmaschine. Eine definierte Schnittstelle (siehe Abschnitt 3.3.1) ermöglicht die Anbindung verschiedener Indexer. So wird der Harvest-Broker standardmäßig mit Glimpse als Index konfiguriert. Dieses Kapitel beschreibt einen neuen Suchindex, dessen Entwicklung auf den Ergebnissen des Kapitels 4 basiert.

6.1 Motivation und Zielsetzung

In Abschnitt 4.2 wurde die Einschränkungen der Funktionalität des Suchindexes des Harvest-Systems beschrieben. Folgende Forderungen machten die Neuentwicklung eines Indexes notwendig:

Keyword-Relevanz-Filter: Er soll entscheiden, welche Schlüsselwörter aufgrund von Worthäufigkeiten auszuschneiden sind. Dieser Filter läßt sich für die Suche und für die erweiterte Dokumentenbeschreibung verwenden.

Ranking der Suchergebnisse: Je nach zu errechnendem Gewicht sollen die gefundenen Dokumente gereiht werden. Dieses Gewicht hängt von der Formatierung und Häufigkeit des Suchwortes im jeweiligem Dokument ab.

Zweistufige Suche: Bevor in den Dokumenten nach Suchbegriffen gesucht wird, soll die Anzahl der Ergebnisse vom System abgeschätzt werden und gegebenenfalls automatisch die Suche auf Server- und Site-Ebene durchgeführt werden. Dies bringt einerseits eine Effizienzsteigerung mit sich. Andererseits ist die Ergebnisliste kürzer und somit übersichtlicher und brauchbarer.

6.1.1 Keyword-Relevanz

Ein Keyword-Filter hat die Aufgabe, nicht gewünschte Schlüsselwörter auszuschneiden (siehe auch Abschnitt 4.2.1). Statisch kann dies mittels Stoppliste geschehen. Ein Relevanzfilter hingegen zählt die Dokumente, in denen das jeweilige Schlüsselwort auftritt und entscheidet dann, ob dieses auch tatsächlich als solches verwendet wird.

Solange ein Broker nur von einem Gatherer bedient wird, könnte der Gatherer die Relevanzberechnung lokal vornehmen. Er verfügt ja über alle Dokumente. Der Broker ist somit speziell für einen Server zugeschnitten. Wenn ein Broker Informationen von mehr als einen Gatherer bekommt, ist dieses Verfahren nicht mehr zielführend.

- Ein Wort, das oft auftritt, aber gut über die einzelnen Server verteilt ist, würde aufgenommen; ein geballt auftretendes Wort mit gleicher Häufigkeit nicht.
- Suchwörter, die speziell auf einem Infoserver vermehrt vorhanden sind, könnten nicht verwendet werden, weil diese vom Keyword-Filter des lokalen Gatherer ausgeschlossen würden. Zum Beispiel müßte der Gatherer eines Grazer Informationsservers das Schlüsselwort “Graz” ausscheiden, obwohl es für einen Österreich-Broker relevant wäre.

Daher scheint es am sinnvollsten, den Keyword-Filter in den Broker zu integrieren. Da der Broker in ein hierarchisches System eingebunden werden soll und sich bei übergeordnete Broker andere Worthäufigkeiten ergeben, muß er alle Schlüsselwörter verwalten; für sich selbst scheidet er sie entsprechend der Häufigkeit in seinem Bereich aus.

6.2 Datenbankdesign

Relationale Datenbanksysteme [Dat86] [YC95] verfügen mittlerweile über Zugriffsmechanismen¹, die es erlauben, effizient nach Textteilen zu suchen. SQL-Datenbanken sind weit verbreitet und bietet sich somit als Basis für den Suchindex an.

Bevor auf die Details der Index-Datenbank eingegangen wird, werden im folgende Abschnitt die Grundprinzipien relationaler Datenbanken kurz erklärt.

6.2.1 Theorie relationaler Datenbanken

Bei relationalen Datenbanken werden Daten in Tabellen abgelegt. Dabei entspricht jede Zeile einem Objekt. Die Spalten bezeichnen die Attribute der Objekte. [Dat86]

Damit Datenbank-anomalien² vermieden werden, müssen die Relationen (Tabellen) einer Datenbank den Normalformen entsprechen. Die erste Normalform besagt, daß die Attributtypen elementar sein müssen, das heißt, zusammengesetzte Typen wie Felder oder Strukturen sind nicht erlaubt. Da aber Dokumente im Allgemeinen eine beliebige Anzahl von Wörtern beinhalten, können diese nicht als Attribute direkt dem Dokument zugeordnet werden. Zuordnungen von Dokument zu enthaltenen Wörtern sind nicht durch eine Zeile realisierbar. Pro Wort im Dokument ist jeweils ein eigener Eintrag in einer Tabelle notwendig (siehe Abbildung 6.1). [Dat86]

¹*Btrees* in Verbindung mit dem *Like*-Vergleichsoperator [FB92]

²Zum Beispiel bezeichnet die Update-Anomalie das Problem, das auftritt, wenn Attribute eines realen Objektes in mehreren Tabellen vorhanden sind. Wird bei einer Änderung eines Attributes diese nicht in allen Tabellen vollzogen, kommt es zu Inkonsistenz der Daten. [Dat86]

Würden nun pro Zuordnung sämtliche Dokument- und sämtliche Wortattribute in einer Tabelle angegeben, hätte dies einige Nachteile. Nicht nur, daß unnötig viel Speicher verbraucht würde, es müßten auch bei einer Änderung eines Dokumentattributes alle dazugehörenden Zeilen geändert werden (“Update”-Anomalie). Abbildung 6.1 zeigt eine solche Relation.

Dokumentname	Wortanzahl	Wort	Gesamthäufigkeit	Häufigkeit im Dokument
...
SQL.html	3	Daten	24	2
SQL.html	3	Bank	7	1
index.html
...

Abbildung 6.1: Beispiel für die Verletzung der zweiten Normalform

Abhilfe schafft hier die Einführung der zweiten Normalform:

Unter einem Schlüssel versteht man die kleinste Attributmenge, die ein Objekt einer Tabelle eindeutig bestimmt. Die zweite Normalform besagt nun, daß kein Nichtschlüsselattribut nur von einem Teil des Schlüssels abhängig sein darf. In unserem Fall wäre der Dokumentname und das Wort gemeinsam der Schlüssel, die Dokumentattribute sind aber nur vom Dokumentnamen, die Wortattribute nur vom Wort abhängig. Lediglich das Attribut “Häufigkeit im Dokument” hängt vom gesamten Schlüssel ab. Es sei noch angemerkt, daß insgesamt fünf Normalformen beim Datenbankdesign zu berücksichtigen sind. [Dat86] Diese spielen aber bei der weiteren Betrachtung des Suchindex keine Rolle. Durch Aufspalten der Tabelle aus Abbildung 6.1 in drei Tabellen, erreicht man die zweite Normalform:

Dokumenttabelle			Worttabelle			Wort-Dokument-Tabelle		
DID	D.-name	W.-anzahl	WID	Wort	G.-häufigkeit	DID	WID	Häufigkeit im D.
...
1	SQL.html	3	1	Daten	24	1	1	2
2	index.html	...	2	Bank	7	1	2	1
...

Abbildung 6.2: Beispiel für die Erlangung der zweiten Normalform

6.2.2 Die Relationen des Suchindex

In diesem Abschnitt werden die einzelnen Tabellen des Suchindex und ihrer Bedeutung beschrieben. Wie die Indizierung und die Verwendung der Tabellen funktioniert, zeigt dann Abschnitt 6.3.

Prinzipiell wurde jedes Objekt der Problemstellung (Wort, Dokument, Site bzw. Server) in eine Relation abgebildet, damit die in Abschnitt 6.2.1 beschriebenen Normalformen gelten. Weitere Relationen werden benötigt, um Beziehungen und Beziehungseigenschaften zwischen zwei Objekten zu realisieren. So steht in einer solchen Tabelle zum Beispiel welches Wort in welchem Dokument wie oft vorkommt.

Damit auch die in Abschnitt 4.2.4 beschriebene zweistufige Suche möglich wird, muß weiters das Objekt “Site” eingeführt werden. Damit die Beziehung zwischen Wörtern

und Site realisiert werden kann, muß wieder eine zusätzliche Relation eingeführt werden. Welche Objekte sollen nun aber mit dem Site-Objekt verknüpft werden?

- Werden Dokumenten und Sites verknüpft, wäre eine hierarchische Struktur geschaffen. Wörter wären Dokumenten und Dokumente wären Sites zugeordnet. Es ließe sich somit auch feststellen, welche Wörter in welchen Sites vorhanden sind. Allerdings hätte diese Suche den entscheidenden Nachteil, daß obwohl nur “grob” in den Sites gesucht wird, zuerst alle Dokumente durchsucht werden müßten. Die zweistufige Suche brächte so keine Effizienzsteigerung.
- Werden Wörter direkt der Site zugeordnet, in der sie vorkommen, kann ohne Umweg über die Dokument-Relation direkt die Sites gesucht werden. Nachteilig wirkt sich bei dieser Strategie der größere Speicherbedarf aus. Da es im Allgemeinen mehr Wörter als Dokumente auf einer Site gibt, wären hier wesentlich mehr Einträge in der Beziehungsrelation zu speichern.

Der Speicherbedarf einer Wort-Site-Beziehungsrelation ist maximal gleich³ groß der Wort-Dokument-Beziehungsrelation. Im Allgemeinen ist sie wesentlich kleiner, da alle Wort-Dokument-Beziehungen eines Wortes zu Dokumenten einer Site nur zu einer Zeile in der Wort-Site-Tabelle führen. Weil sich dieser Mehraufwand in Grenzen hält, wurde die Wort-Site-Relation eingeführt. So kann direkt, ohne den Umweg über die Dokumententabelle, in den Sites gesucht werden.

Worttabelle

Die Worttabelle (siehe Abbildung 6.3) enthält neben dem eigentlichen Wort (**WORD**) noch zusätzliche Information das Wort betreffend, und eine Wort-ID (**WID**). Beim Indizieren werden die Wörter eines Dokumentes in diese Tabelle eingetragen und eine eindeutige ID (Schlüssel) vergeben. In den restlichen Attributen (**ABODY**, **AKEYWORD**, **ATITLE**) wird festgehalten, in wie vielen Dokumenten das entsprechende Wort im Volltext, als Schlüsselwort und im Titel vorkommt. Ist dieses Wort schon aufgenommen, werden nur diese Zähler erhöht.

WID	WORD	ABODY	AKEYWORD	ATITLE
...
405	jupiter	23	4	1
406	mars	27	2	1
407	sonne	29	5	3
...

Abbildung 6.3: Worttabelle - WORT-T

Bei Löschen eines Dokumentes werden die Inhalte der Zähler bei Bedarf entsprechend reduziert. Haben alle Zähler den Wert Null, ist dieses Wort in keinem Dokument

³Für den Fall, daß jede Site genau ein Dokument enthält, gibt es in beiden Relationen gleich viel Einträge.

mehr vorhanden. Der Eintrag bleibt jedoch bestehen. Dies erspart den Aufwand des Löschens und des evetuell neuen Anlegens eines Worteintrags.

Dokumenttabelle

Sie enthält neben der URL (**URL**) und dem Dateinamen (**FILENAME**) des korrespondierenden SOIF-Objektes noch ein Attribut (**LTIME**), das den Zeitpunkt⁴ der letzten Änderung des Dokumentes enthält und eine Dokument-ID (siehe Abbildung 6.4). Auch in der Dokumenttabelle werden Attributzähler mitgeführt (**WBODY**, **WTITLE**, **WKEYWORD**). Sie enthalten die Anzahl der Wörter im Dokument getrennt nach Attributzugehörigkeit. Beim Indizieren werden diese Daten des Dokumentes in die Tabelle eingetragen und eine eindeutige ID (Schlüssel) vergeben.

IID	URL	FILENAME	LTIME	WBODY	WTITLE	WKEYWORD
...
5	planet.html	OBJ1907865	9400023	20	15	1
6	index.html	OBJ6898324	9400059	124	27	5
7	stars.html	OBJ8989896	9400099	6755	152	12
...

Abbildung 6.4: Dokumenttabelle - DOC-T

Wort-Dokument-Tabelle

Diese Tabelle (siehe Abbildung 6.5) stellt die Verbindung zwischen Wort und Dokument dar. Jeder Eintrag enthält eine Wort-ID (**WID**) und die Dokument-ID (**IID**) jenes Dokumentes, in dem das Wort vorkommt. Beide IDs bilden den Schlüssel für diese Relation. Zusätzliche Attribute (**DBODY**, **DKEYWORD**, **DTITLE**) halten fest, wie oft das Wort im entsprechenden Dokument im Volltext, als Schlüsselwort und im Titel vorkommt.

WID	IID	DBODY	DKEYWORD	DTITLE
...
405	5	9	0	1
405	68	6	2	1
406	5	55	12	0
...

Abbildung 6.5: Wort-Dokumenttabelle - WD-T

⁴Sekunden seit 1. Januar 1970

Site-Tabelle

Sie enthält neben der URL (**URL**) den Dateinamen (**FILENAME**) einer Site-Beschreibung und eine Site-ID (**IID**). Weiters wird die Anzahl der enthaltenen Dokumente in einem Attribut (**DOKCOUNT**) festgehalten (siehe Abbildung 6.6). Ein Eintrag in dieser Tabelle erfolgt durch eine Site-Anmeldung. Die Dokumente werden dann bei der Indizierung mitgezählt.

IID	URL	FILENAME	DOKCOUNT
...
17	www.iicm.edu	SITE089	988
18	www.ml.org	SITE164	2045
19	styria:9552/TouchTheCity/	SITE091	109
...

Abbildung 6.6: Site-Tabelle - SITE-T

Wort-Site-Tabelle

Diese Tabelle (siehe Abbildung 6.7) stellt die Verbindung zwischen Wort und Site dar. Jeder Eintrag enthält eine Wort-ID (**WID**) und die Site-ID (**IID**) jener Site, in der das Wort vorkommt. Beide IDs bilden den Schlüssel für diese Relation. Zusätzliche Attribute (**SBODY**, **SKEYWORD**, **STITLE**) halten fest, in wie vielen Dokumenten der Site das Wort im Volltext, als Schlüsselwort und im Titel vorkommt. Beim Indizieren eines Dokumentes wird zuerst anhand der URL festgestellt, zu welcher Site es gehört. Die Wörter werden dann durch einen Eintrag in dieser Tabelle auch der entsprechenden Site zugeordnet, falls diese angemeldet wurde und somit in der Site-Tabelle enthalten ist.

Wird, nachdem bereits Dokumente eines ganzen Servers indiziert wurden, ein neuer Teilbereich dieses Servers angemeldet, müssen die Wörter der Dokumente, die diesem neuen Web-Bereich angehören, neu zugeordnet werden. Ähnliches muß beim Abmelden eines Servers oder Site erfolgen. Die Details werden im folgenden Abschnitt beschrieben.

IID	WID	SBODY	SKEYWORD	STITLE
...
5	19	19	2	2
6	18	60	2	1
5	18	55	12	0
...

Abbildung 6.7: Wort-Site-Tabelle - WS-T

Wort-Site-Info-Tabelle

Den Sites (Server, Web-Bereiche) werden nicht nur die Wörter der enthaltenen Dokumente zugeordnet, sondern auch Worte, die in einer Site-Beschreibung extrahiert werden. Diese wird indiziert, wenn der Systemadministrator einen zuvor angemeldeten Web-Bereich in den Broker aufnimmt. Tabelle (siehe Abbildung 6.8) stellt die Verbindung zwischen Wort und Site dar. Jeder Eintrag enthält eine Wort-ID (WID) und die Site-ID (IID) jener Site, in der das Wort vorkommt. Da hier nicht nach Attributen unterschieden wird, werden keine Attributzhälter verwendet.

IID	WID
...	...
5	4149
6	34148
5	138444
...	...

Abbildung 6.8: Wort-Site-Info-Tabelle - WS-T

6.3 Indizierung

Nachdem im vorangegangenen Abschnitt die logische Struktur des Indexes aufgezeigt wurde, wird nun gezeigt, wie diese Struktur für die Indizierung verwendet wird. Es werden die einzelnen Datenbankoperationen beschrieben.

Wichtig hierbei ist das Konzept der Transaktion, welches es möglich macht, daß komplexere, zusammengesetzte Datenbankmanipulationen entweder als Ganzes oder gar nicht durchgeführt werden. Wird zum Beispiel ein Dokument entfernt, müssen auch alle Wort-Dokument-Beziehungen gelöscht werden. Tritt während dieser Operationen ein Fehler auf, wird vom Datenbanksystem der alte Zustand vor Beginn der Transaktion wieder hergestellt. Mit Hilfe des Transaktionskonzeptes werden zusammengesetzte Operationen somit "atomar".

Vor dem eigentlichen Indizieren werden eine Reihen von Prozeduren durchlaufen, um die weiteren Vorgänge optimieren zu können:

- Die häufigsten Wörter werden samt Wort-ID in einen Cache (Binärbaum) geladen, um die Suche nach bereits vorhandenen Wörtern zu beschleunigen. Dieser Cache wird auch während des Einfügens neu aufgebaut, wenn die Anzahl der Wörter, die nicht im Cache gefunden wurden, einen Schwellwert übersteigt.
- Die größte Wort-ID wird ermittelt und gespeichert. Wird ein neues Wort angelegt, wird dieser Wert inkrementiert und als neue Wort-ID verwendet.
- Wenn ein Dokumenteintrag gelöscht wird, wird ein Dokument-ID frei. Diese freien IDs werden ermittelt und stehen dann beim Einfügen eines Eintrages in der Dokumententabelle zur Verfügung.

6.3.1 Einfügen

Das Einfügen von Dokumenten besteht aus mehreren Schritten, die, abhängig vom Status der Datenbank, verzweigen können.

Einfügen in die Dokumententabelle :

1. Es wird überprüft, ob das zu indizierende Objekt ein gültiges SOIF-Objekt ist. Nur im positiven Fall wird mit dem Einfügen begonnen.
2. Falls ein Dokument mit der gleichen URL schon im Index ist, wird die Zeit der letzten Änderung (**Last-Modification-Time, LTIME**) untersucht. Nur wenn das vorhandene Dokument älter ist, wird dieses Dokument aus der gesamten Datenbank entfernt und dann durch das neue ersetzt. Es werden somit also auch die Wort-Dokument-Beziehungsrelationen entfernt und die entsprechenden Zähler in den dazugehörenden Einträgen der Worttabelle je nach Vorkommen in **body**, **title** oder **keyword** um eins verkleinert (siehe Abschnitt 6.3.2). Nach dem Löschen wird die alte Dokument-ID verwendet (**IID**). Es wird so der bestehende Eintrag in der Dokumententabelle erneuert, indem die Zeit der letzten Änderung (**LTIME**) und ein eventuell neuer Dateiname des SOIF-Objektes überschrieben werden.
3. Falls das Dokument noch nicht vorhanden ist, wird nach der kleinsten freien ID⁵ (**IID**) gesucht. Wurde eine solche gefunden, werden die Daten des neuen Dokuments in die entsprechende Zeile geschrieben. Nur wenn keine freie Dokument-ID zur Verfügung steht, wird ein neuer Eintrag mit einer neuen ID erzeugt. Wurde keine Manipulation in der Dokumententabelle vorgenommen, weil die aktuelle Version des Dokuments bereits im Index enthalten ist, wird die Indizierung dieses Objektes abgebrochen.

Ermittlung der dazugehörenden Site : Mit Hilfe der URL des einzufügenden Dokumentes wird in der Site-Tabelle nach einer dazugehörenden Site gesucht. Falls eine solche vorhanden ist, wird der Dokumentenzähler um eins erhöht, da ein weiteres Dokument der Site zugeordnet wurde. Beim anschließenden Einfügen der Wörter werden dann zusätzlich Einträge in der Wort-Site-Tabelle vorgenommen.

Einfügen der Wörter :

1. Die SOIF-Attribute **body**, **keywords** und **title** des zu indizierenden SOIF-Objektes werden ausgelesen.
2. Bevor die Wörter eines Dokumentes in die Datenbank eingefügt werden, werden sie sortiert. Anschließend werden die Häufigkeit ihres Auftretens getrennt nach den drei SOIF-Attributen ermittelt und in der Dokumententabelle in die drei Spalten **WBODY**, **WTITLE** und **WKEYWORD** vermerkt.

⁵Solche entstehen, wenn Dokumente aus dem Index gelöscht werden

3. Nun werden die Wörter einzeln in die Datenbank eingefügt. Dabei wird zuerst überprüft, ob das Wort bereits in der Wort-Tabelle existiert:
 - (a) Im Wort-Cache wird nach der Wort-ID des Wortes gesucht.
 - (b) Falls das Wort nicht im Wort-Cache vorhanden ist, wird in der Wort-Tabelle gesucht.
 - (c) Falls ein Worteintrag gefunden wurde, werden die jeweiligen Attribut-Zähler (**ABODY**, **ATITLE**, **AKEYWORD**) des Eintrages um eins vergrößert, da dieses Wort nun einem weiteren Dokument zugeordnet wird. Kommt zum Beispiel das Wort im Dokument nur im Titel vor, wird das Tabellenattribut **ATITLE** inkrementiert, unabhängig von der Häufigkeit im Dokument.
 - (d) Falls das Wort nicht gefunden wurde, wird ein neuer Eintrag in der Wort-Tabelle erstellt. Da diese Einträge nicht mehr gelöscht werden, können keine “ID-Löcher” entstehen. Es wird die größte Wort-ID um eins vergrößert und verwendet. Die Attribut-Zähler (**ABODY**, **ATITLE**, **AKEYWORD**) des neuen Eintrages werden auf Eins beziehungsweise Null gesetzt, je nachdem ob das Wort im entsprechenden SOIF-Attribut vorkommt. Kommt zum Beispiel das Wort fünfmal im Volltext und dreimal als Schlüsselwort im Dokument vor, wird der Wert von **ABODY** und **AKEYWORD** auf Eins, der Wert von **ATITLE** auf Null gesetzt.
 - (e) Ein neuer Eintrag in der Wort-Dokument-Tabelle wird erstellt. Er enthält die Dokument- und die Wort-ID und die Anzahl des Auftretens des Wortes im betreffenden Dokument getrennt nach SOIF-Attributen. Hier wird also die Häufigkeit im Dokument vermerkt. Kommt ein Wort dreimal im Volltext und einmal im Titel vor, wird der Wert von **DBODY** auf “drei” und **DTITLE** auf Eins gesetzt.
 - (f) In der Wort-Site-Tabelle wird, falls die Site im Index existiert und das Wort nicht gerade neu in die Wortatabelle aufgenommen wurde, nach einem bereits vorhandenen Eintrag gesucht.
 - Falls ein solcher Eintrag existiert, enthält ein bereits indiziertes Dokument dieser Site das Wort. Es brauchen nur mehr die jeweiligen Attribut-Zähler des Eintrages entsprechend um eins vergrößert werden, da diese Tabellenattribute die Dokumente zählen, in denen das Wort vorkommt.
 - Andernfalls wird ein neuer Eintrag mit Wort-ID und Site-ID angelegt und die jeweiligen Attribut-Zähler des Eintrages werden auf eins gesetzt, da das einzufügende Dokument das erste ist, das auf dieser Site dieses Wort enthält.

Dokumente im Index sind voneinander unabhängig, das heißt, ein Neueintrag beeinflusst die anderen Dokumenteinträge nicht. Anders ist dies bei Site-Einträge. Wird eine Site in den Index aufgenommen, muß überprüft werden, ob nicht bereits eine übergeordnete Site im Index enthalten ist. Ist dies der Fall, müssen alle Zuordnungen der Wörter der neuen Site zur übergeordneten Site in Zuordnungen zur neuen Site umgewandelt werden. Auf diese Problematik wird in Abschnitt 6.4 genauer eingegangen.

6.3.2 Löschen

Die Löschoperation kann explizit durch den Broker aufgerufen werden. Sie wird auch dann verwendet, wenn ein Dokument mit gleicher URL und jüngerem Erstelldatums eingefügt werden soll (siehe Abschnitt 6.3.1). Folgende Schritte werden hierbei durchlaufen:

Behandlung des Eintrages in der Dokumententabelle :

1. Mit Hilfe der URL wird überprüft, ob das zu löschende Dokument im Index vorhanden ist.
2. Ist dies der Fall, wird nur die URL aus dem Eintrag entfernt. Somit kann auf einfache Weise die Dokument-ID wiederverwendet werden (siehe 6.3.1).

Ermittlung der dazugehörenden Site : Mit Hilfe der URL des zu löschenden Dokumentes wird in der Site-Tabelle nach einer dazugehörenden Site gesucht. Falls eine solche vorhanden ist, wird der Dokumentzähler um eins verkleinert, da ein Dokument der Site entfernt wurde. Bei der anschließenden Behandlung der Wörter werden dann zusätzlich Löschungen in der Wort-Site-Tabelle vorgenommen.

Behandlung der Wörter : In der Wort-Dokument-Tabelle wird nach allen Einträgen des zu löschenden Dokumentes gesucht. Für jedes gefundene Wort wird nun folgende Prozedur durchlaufen:

1. Der Eintrag in der Wort-Dokument-Tabelle wird gelöscht.
2. War im Eintrag der Wort-Dokument-Tabelle ein Attribut-Zähler größer als Null⁶, wird der jeweilige Attribut-Zähler im Eintrag der Worttabelle des Wortes um eins verkleinert. Hat zum Beispiel der Attribut-Zähler DBODY in der Wort-Dokument-Tabelle den Wert "5", kommt dieses Wort im Volltext des zu löschenden Dokumentes vor. Daher muß der Attribut-Zähler ABODY in der Worttabelle genau im eins verkleinert werden. Das Wort kommt in einem Dokument weniger im Volltext vor.
3. In der Wort-Site-Tabelle wird, falls die Site im Index existiert, nach dem Eintrag des zu behandelnden Wortes gesucht.
4. War im Eintrag der Wort-Dokument-Tabelle ein Attribut-Zähler größer als Null, wird der entsprechende Attribut-Zähler im Eintrag der Wort-Site-Tabelle des Wortes um eins verkleinert.

6.4 Verwalten von Servern und Web-Areas

Neben der Zuordnung zwischen Wort und Dokument können die Wörter eines Dokumentes bei dessen Indizierung auch einem Web-Bereich zugeordnet werden. Dies setzt allerdings voraus, daß dieser zuvor beim System angemeldet wurde und seine Daten

⁶Eine Attributzähler muß größer als Null sein, da es sonst diesen Eintrag nicht gäbe.

in den Suchindex aufgenommen wurden (siehe Abschnitt 6.2.2 und Abbildung 6.7). Unabhängig von den Dokumenten wird eine Beschreibung der Site (Server, Web-Area) indiziert (siehe Abbildung 6.8).

Wird zum Beispiel ein ganzer Server angemeldet und in den Index übernommen, werden folgende Schritte durchlaufen:

1. Der Web-Bereich kann nur aufgenommen werden, wenn er noch nicht im Index existiert. Ist also dessen URL noch nicht in der Site-Tabelle (**SITE-T**) enthalten, wird ein neuer Eintrag eingefügt.
2. Die Site-Beschreibung wird indiziert, indem die Wörter in der Worttabelle (**WORD-T**) gesucht und eventuell neu eingefügt werden und anschließend eine Relation zwischen Site und Wort in die Wort-Site-Info-Tabelle (**WI-T**) eingefügt wird.
3. Anhand der URL des neuen Web-Bereiches werden Dokumente in der Dokumententabelle gesucht, die zu diesem Bereich gehören. Für jedes Wort, das in der Gesamtheit diesen Dokumenten vorkommt, wird eine Wort-Site-Relation in der Wort-Site-Tabelle (**WS-T**) angelegt. Die Anzahl der Dokumente, die dieses Wort enthalten, wird ebenfalls dort vermerkt (**DOCCOUNT**).

Wird ein Web-Bereich abgemeldet, wird sowohl der Eintrag in der Site-Tabelle als auch die entsprechenden Wort-Site-Relationen in der Wort-Site-Tabelle und Wort-Site-Info-Tabelle entfernt.

Eine Besonderheit tritt auf, wenn ein Web-Bereich aufgenommen werden soll, der einen Teilbereich eines bereits angemeldeten Bereiches entspricht. So kann zum Beispiel der Server `http://www.iicm.edu/` im Index enthalten sein, wenn die Web-Area `http://www.iicm.edu/staff/` neu aufgenommen werden soll. In solchen Fällen müssen nach den beiden oben angeführten Schritten alle Wörter der neuen Wort-Site-Relationen (`http://www.iicm.edu/staff/`) in den Wort-Site-Relationen des übergeordneten Bereiches (`http://www.iicm.edu/`) gesucht werden und der Dokumentenzähler um Eins reduziert werden, da ja ein Dokument aus diesem Bereich in einen neuen verschoben wurde. Ist ein Dokumentenzähler gleich Null, wird die Relation gelöscht.

6.5 Suchabfragen

Der Syntax der Suchabfragen des Benutzers bleiben durch die Integration des neuen Suchindex vorerst unberührt. Die Indexschnittstelle (Abschnitt 3.3.1) des Harvest-Broker muß hingegen an den neuen Index angepasst werden. Abschnitt 6.7 zeigt, wie der Suchindex in den Broker integriert wurde. In diesem Abschnitt werden SQL-Abfragen zu speziellen Suchanfragetypen vorgestellt, die auch vom Harvest-Broker mittels der Indexschnittstelle verwendet werden.

6.5.1 Prinzip der Suchabfrage

Mit dem SQL-Befehl **select** lassen sich Zeilen einer Tabelle ausgeben. Welche Zeilen ausgegeben werden sollen, wird mit einer **where**-Klausel bestimmt. Da nun die

Wort- und Dokumentattribute aufgrund der zweiten Normalform (Abschnitt 6.2.1) in unterschiedlichen Tabellen verwaltet werden, muß die Suche nach einem Dokument oder einer Site immer über drei Tabellen erfolgen. SQL bietet deshalb die Möglichkeit der “Join”-Operation (Kreuzprodukt). Hierbei wird eine neue, temporäre Tabelle erzeugt, die alle möglichen Zeilenkombinationen beinhaltet. Abbildung 6.9 zeigt die “Join”-Tabelle aus den drei Tabellen aus Abbildung 6.2

IID	D.Name	W.-Anzahl	WID	Wort	G.-häufigkeit	IID	WID	D.-häufigkeit
...
1	SQL.html	3	1	Daten	24	1	1	2
1	SQL.html	3	1	Daten	24	1	2	1
1	SQL.html	3	2	Bank	7	1	1	2
1	SQL.html	3	2	Bank	7	1	2	1
2	index.html	...	1	Daten	24	1	1	2
2	index.html	...	1	Daten	24	1	2	1
2	index.html	...	2	Bank	7	1	1	2
2	index.html	...	2	Bank	7	1	2	1
...

Abbildung 6.9: Prinzip des “Join”-Operators: “Join” angewandt auf die drei Tabellen aus Abbildung 6.2

Aus diesem Beispiel wird ersichtlich, daß das “Join” noch nicht ausreicht, um nur logisch zusammengehörende Teile in eine Zeilen zu bekommen. So steht zum Beispiel das Wort “Bank” gemeinsam in einer Zeile mit “index.html”, obwohl das Wort in diesem Dokument nicht vorkommt. Daher muß in der **Where**-Klausel beschrieben werden, welche Zeilen der ursprünglichen Tabellen logisch zusammengehören. Im Fall des Index werden Zeilen nur dann zusammengelegt, wenn die IDs gleich sind. Im Beispiel von Abbildung 6.9 müssen also die zwei Wort- und die zwei Dokument-IDs gleich sein.

IID	D.Name	W.-Anzahl	WID	Wort	G.-häufigkeit	IID	WID	D.-häufigkeit
...
1	SQL.html	3	1	Daten	24	1	1	2
1	SQL.html	3	2	Bank	7	1	2	1
...

Abbildung 6.10: Tabelle aus Abbildung 6.9 nach Anwendung der **Where**-Klausel, welche die Gleichheit der IDs verlangt

6.5.2 Sucheanfrage mit einem Wort

Wird nach einem Dokument gesucht, das ein bestimmtes Wort enthalten soll, werden die Dokument-, die Wort-, und die Wort-Dokument-Tabelle mittels “Join”-Operator zusammengelegt. Ähnliches gilt für die Suche in Sites. Hier werden die Site-, die Wort- und die Wort-Site-Tabelle zusammengelegt. In der **Where**-Klausel des **Select**-Befehls werden Einträge mit nicht zusammenpassenden Teileinträgen entfernt. Eine weitere Bedingung in der **Where**-Klausel gibt schließlich an, welches Wort (das Suchwort) in der Zeile stehen muß.

In folgenden Beispiel werden die oben in Abschnitt 6.2.2 beschriebenen Tabelle verwendet. Die Abfrage “jupiter” liefert als Ergebnis “planet.html”

```
select URL from DOC-T, WD-T, WORD-T
where DOC-T.IID=WD-T.IID and WD-T.WID=WORD-T.WID and
WORD='jupiter'
```

Nachdem SQL-Befehl `select` wird angegeben, welches Attribut der Tabellen zurückgeliefert werden soll. In diesem Beispiel ist es die URL des gefundenen Dokumentes. Das SQL-Schlüsselwort `from` wird gefolgt von den Namen der Tabellen, die mittels “Join” temporär verknüpft werden. Hier sind dies `DOC-T`, `WD-T` und `WORD-T`. Die `Where`-Klausel des `Select`-Befehls legt in der zweiten Zeile fest, daß die Dokumenten-ID der Dokumententabelle (`DOC-T.IID`) gleich der der Wort-Dokument-Tabelle (`WD-T.IID`) und die Wort-ID der Wort-Dokument-Tabelle (`WD-T.WID`) gleich der der Worttabelle (`WORD-T.WID`) sein muß. Schließlich wird hier festgelegt, welches Wort gesucht wird (`WORD='jupiter'`).

Die meisten SQL-Datenbanksysteme erlauben die Angabe von regulären Ausdrücken. Mittels Platzhalter (“%”)⁷ kann die Suche auf Wortklassen ausgedehnt werden. Statt dem Gleichheitsoperator (“=”) muß dann der `like`-Operator verwendet werden (...and `WORD like '%it%'`).

6.5.3 Boolesche Operatoren

Mit Hilfe der Mengenoperatoren `INTERSECT` und `UNION` lassen sich die booleschen Operatoren `AND` und `OR` für den Index in SQL realisieren. Beim einer Konjunktion der Suchbegriffe werden für jeden die Dokumente gesucht und anschließend der Durchschnitt `INTERSECT` der einzelnen Mengen ermittelt. Bei der Disjunktion werden die einzelnen Mengen durch `UNION` vereinigt. Das folgende Beispiel liefert als Ergebnis in beiden Fällen “planet.html”

```
select URL from DOC-T, WD-T, WORD-T
where DOC-T.IID=WD-T.IID and WD-T.WID=WORD-T.WID and
WORD='jupiter'
INTERSECT
select URL from DOC-T, WD-T, WORD-T
where DOC-T.IID=WD-T.IID and WD-T.WID=WORD-T.WID and
WORD='mars'
```

```
select URL from DOC-T, WD-T, WORD-T
where DOC-T.IID=WD-T.IID and WD-T.WID=WORD-T.WID and
WORD='jupiter'
UNION
select URL from DOC-T, WD-T, WORD-T
where DOC-T.IID=WD-T.IID and WD-T.WID=WORD-T.WID and
WORD='mars'
```

⁷Postgres-SQL [YC95] unterstützt nur diesen Platzhalter für beliebig viele Zeichen

Der Not-Operator wird mittels “Sub-Select” realisiert. In diesem werden alle Dokumente gesucht, die das “verbotene” Wort enthalten. Dieses “Select” wird also als zusätzliche Bedingung in der **Where**-Klausel festgelegt, daß die Dokument-ID nicht in der Menge der Dokument-IDs von Dokumenten, die das “verbotenen” Wort enthalten, sein darf. Das folgende Beispiel (“jupiter AND NOT mars”) liefert als Ergebnis die leere Menge

```
select URL from DOC-T, WD-T, WORD-T
where DOC-T.IID=WD-T.IID and WD-T.WID=WORD-T.WID
and WORD='mars' and DOC-T.IID not in
(select IID from WD-T, WORD-T
where WD-T.WID=WORD-T.WID and WORD='jupiter')
```

6.5.4 Eingeschränkte Suche

Einschränkung auf Attribute

Durch eine zusätzliche Bedingung in der **Where**-Klausel, läßt sich die Suche individuell für jedes Wort einschränken. Im folgenden Beispiel muß der Suchbegriff “mars” im Titel vorkommen. Der Attribut-Zähler **WD-T.DTITEL** gibt an, wie oft das Suchwort im gefundenen Dokument im Titel vorkommt, und muß in diesem Beispiel größer als Null sein.

```
select URL from DOC-T, WD-T, WORD-T
where DOC-T.IID=WD-T.IID and WD-T.WID=WORD-T.WID
and WORD='mars' and WD-T.DTITEL>0
```

Im nächsten Beispiel darf der Suchbegriff “mars” zusätzlich in keinem anderen Dokument im Titel vorkommen. Der Titel-Zähler der Worttabelle (**WORD.ATITEL**) enthält die Anzahl der Dokumente, in denen das Wort im Titel vorkommt, und muß in diesem Beispiel gleich Eins sein.

```
select URL from DOC-T, WD-T, WORD-T
where DOC-T.IID=WD-T.IID and WD-T.WID=WORD-T.WID
and WORD='mars' and WD-T.DTITEL>0 and WORD.ATITEL=1
```

Das letzte Beispiel zeigt, wie man den Attributzähler in der Dokumententabelle benutzen kann, um relative Wortbedeutungen zu ermitteln. Es sollen jene Wörter des Dokuments mit der URL ‘index.html’ zurückgeliefert werden, deren Anzahl im Volltext größer als ein zehntel der Gesamtwortanzahl im Volltextattribut ist.

```
select URL from DOC-T, WD-T, WORD-T
where DOC-T.IID=WD-T.IID and WD-T.WID=WORD-T.WID
and (WD-T.DBODY/DOC-T.WBODY)>0.1 and URL='index.html'
```

Konfigurierbarer Keyword-Relevanz-Filter

In der Worttabelle wird unter anderem die Anzahl der Dokumente mitgeführt, die das jeweilige Wort als Keyword enthalten. Mit Hilfe dieses Eintrages läßt sich ein Keywordfilter realisieren. So kann festgelegt werden, in welchem Bereich die Anzahl der Dokumente liegen muß, damit ein Keyword als relevant gilt. Im folgenden Beispiel darf dieser Wert nicht über 100 liegen. Es wird nach Dokumenten gesucht, die das Suchwort “mars” als relevantes Keyword enthalten

```
select URL from DOC-T, WD-T, WORD-T
where DOC-T.IID=WD-T.IID and WD-T.WID=WORD-T.WID
and WORD='mars' and WD-T.DKEYWORD>0 and WORD.AKEYWORD<101
```

WD-T.DKEYWORD gibt an, wie oft das Wort im Dokument als Schlüsselwort vorkommt und muß somit laut Aufgabenstellung größer als Null sein. WORD.AKEYWORD zählt die Dokumente, die das dazugehörige Wort als Schlüsselwort beinhalten und muß in diesem Beispiel kleiner als 101 sein.

Im folgenden Beispiel wird gefragt, welche relevanten Keywords im Dokument “planet.html” enthalten sind. Als Schwellwert gilt wieder die Anzahl von 100 Dokumenten

```
select WORD from DOC-T, WD-T, WORD-T
where DOC-T.IID=WD-T.IID and WD-T.WID=WORD-T.WID
and URL='SQL.html' and WORD-T.AKEYWORD<101
```

6.5.5 Gewichtung gefundener Dokumente

In der Wort-Dokument-Tabelle wird festgehalten, wie und wie oft ein Wort in einem Dokument den einzelnen SOIF-Attributen vorkommt (DBODY, DTITLE, DKEYWORD); in der Dokumententabelle steht, wieder getrennt nach Attributen, wieviele Wörter im Dokument enthalten sind (WBODY). Durch festgelegte Bewertungen der einzelnen SOIF-Attribute, läßt sich so ein Wert errechnen, der die relative Bedeutung eines gefundenen Dokumentes widerspiegelt (siehe nächsten Absatz). Das folgende Beispiel gibt die gefundenen Dokumente entsprechend ihrer Bedeutung sortiert aus

```
select URL, (DBODY/WBODY)*0.1+(DTITLE/WTITLE)*0.5+
(DKEYWORD/WKEYWORDS)*0.4 as WEIGHT from DOC-T, WD-T, WORD-T
where DOC-T.IID=WD-T.IID and WD-T.WID=WORD-T.WID and
WORD='jupiter' and WD-T.DTITEL>0 order by WEIGHT desc
```

Der Quotient (DBODY/WBODY) gibt hier zum Beispiel die relative Bedeutung im Volltextattribut an. Zusätzlich werden die einzelnen SOIF-Attribute in der Suchanfrage unterschiedlich bewertet. Der Alias WEIGHT in der ersten Zeile ist notwendig, damit nach diesem Wert sortiert werden kann.

Komplicierter sehen die Abfragen mit mehreren Suchtermen aus. In den folgenden Beispielen werden die in Abschnitt 4.2.2 vorgestellten Ergebnisse verwendet. Es wird nach “mars AND jupiter” gesucht. Aus Gründen der Übersichtlichkeit wird hier auf die Bewertung der einzelnen Wörter der Suchabfrage verzichtet

```
select URL,1-0.5*sqrt(sum(power(1-(DBODY/WBODY)*0.1+
(DTITLE/WTITLE)*0.5+(DKEYWORD/WKEYWORDS)*0.4,2))) as WEIGHT
from DOC-T where WD-T.WID=WORD-T.WID and WORD='mars'
and IID in (select IID from WD-T, WORD-T
where WD-T.WID=WORD-T.WID and WORD='jupiter')
group by URL order by WEIGHT
```

In diesem Beispiel wurde die “UND”-Verknüpfung anders als in Abschnitt 6.5.3 in SQL ausgedrückt. Der Grund liegt in der Notwendigkeit des “Gruppierens” der zurückgelieferten Zeilen. Während INTERSECT und UNION exakt eine Zeile pro gefundenem Dokument zurückgeben, liegt in diesem Beispiel der Sachverhalt etwas anders. Alle Dokumente, die der oben angeführten Query genügen, werden genau zweimal berücksichtigt, einmal für jedes Suchwort. Durch das Gruppieren werden Dokumente mit gleicher URL zu einer Resultatzelle zusammengefasst, wobei die Aggregatfunktion (sum) die Einzelergebnisse aufsummiert.

Die Disjunktion von Suchtermen kommt wieder ohne “Sub-Select” aus. Im folgenden Beispiel wird nach “mars OR Bank” gesucht

```
select URL,0.5*sqrt(sum(power((DBODY/WBODY)*0.1+
(DTITLE/WTITLE)*0.5+(DKEYWORD/WKEYWORDS)*0.4,2))) as WEIGHT
from DOC-T where WD-T.WID=WORD-T.WID and WORD='mars'
or WORD='Bank' group by URL order by WEIGHT
```

6.6 Mehrstufige Suche

Das oben Erwähnte gilt natürlich auch für die Suche in den Sites. Wie und unter welchen Voraussetzungen kann nun automatisch vom System von der Dokument-Suche auf die Site-Suche umgeschaltet werden.

Bevor die eigentliche Suche beginnt, kann schnell überprüft werden, in wievielen Dokumenten die einzelnen Suchbegriffe vorkommen, indem entweder das Maximum (Disjunktion) oder Minimum (Konjunktion) der Wortgesamthäufigkeiten (ABODY+ATITLE) abgefragt wird. Im folgenden Beispiel wird nach (mars AND jupiter) gesucht

```
select min(ABODY+ATITLE) from WORD-T
where WORD in('mars','jupiter')
```

Der Indexer vergleicht den so erhaltenen Wert mit einem Schwellwert. Wird dieser überschritten, sucht der Indexer in den Sites

```
select URL from SITE-T, WS-T, WORD-T
where SITE-T.IID=WS-T.IID and WS-T.WID=WORD-T.WID and
WORD='mars'
```

6.7 Einbindung in den Harvest-Broker

Der Harvest-Broker verfügt über eine Schnittstelle zur Anbindung beliebiger Suchindexer. In Abschnitt 3.3.1 wurde beschrieben, welche Funktionen vom Broker zum Zwecke der Indizierung und Suche aufgerufen werden.

6.7.1 Die Indizierung

Wenn der Broker von seinen *Collection Points* neue Daten bekommt, wird das Indexinterface davon verständigt. Ist ein Objekt aus dem Index zu löschen, wird `Delete_Object` aufgerufen. Andere Suchmaschinen wie Glimpse reagieren auf diesen Aufruf nicht, da sie am Ende der Übertragung der SOIF-Objekte sämtliche Objekte neu indizieren. Der neue Suchindex löscht hingegen das Objekt wie in Abschnitt 6.3.2 beschrieben. Wurde ein neues Objekt in den Broker aufgenommen, wird der Indexer durch `New_Object` davon in Kenntnis gesetzt. Der Indexer hat drei Möglichkeiten darauf zu reagieren:

- Er indiziert das neue Objekt sofort. Viele Indexer können einzelne Objekte nicht oder nur mit erhöhtem Aufwand indizieren. Der hier beschriebene Indexer indiziert Einzelobjekte in der gleichen Art wie Mengen von Objekten.
- Er merkt sich die Objektnamen in einer Datei vor. Nach Ende der Übertragung können diese SOIF-Objekte indiziert werden. Auch diese inkrementelle Indizierung wird nicht von Glimpse unterstützt. Der neue Suchindex verwendet diese Methode auch nicht, da er eine einfachere Art der inkrementellen Indizierung beherrscht.
- Er ignoriert die Funktionsaufrufe für die einzelnen Objekte und wartet bis die Übertragung beendet ist. Mit `Flush_Index` gibt der Broker dies bekannt. Nun wird einfach das ganze Objektverzeichnis des Broker neu indiziert. Dies ist die einzige Methode, die Glimpse verwenden kann. Der neue Indexer braucht in diesem Fall seinen Index nicht neu aufbauen. Er überprüft lediglich für jedes SOIF-Objekt im Broker, ob dessen aktuelle Version indiziert ist. Dazu müssen URL und die Zeit der letzten Änderung des Objektes mit einem Eintrag in der Dokumententabelle übereinstimmen.

Zwar werden beim Vollindizieren nur neuere SOIF-Objekte tatsächlich neu aufgenommen, dennoch ist diese Methode langsamer, weil die URL und die Zeit der letzten Änderung für jedes Objekt in der Datenbank verglichen werden müssen. Aus diesem Grund ist die objektweise Indizierung vorzuziehen.

Damit der Broker den neuen Indexer verwendet, muß in der Konfigurationsdatei `BROKERHOME/admin/broker.conf` "SQL" als Indexername angegeben werden. Als Index-Typ kann hier wahlweise "PER_OBJ" für die Einzelindizierung und "FULL" für die Vollindizierung eingetragen werden.

Die Server- und Siteberschreibung wird direkt mit dem Broker-Client-Tool (`brkclient`) [HSW96] im Administrator-Modus an den Index übergeben. Angegeben

werden muß der Name des Benutzers, der den Web-Bereich anmeldet, die URL des Bereiches sowie eine kurze und eine längere Beschreibung.

```
brkclient HOST PORT8 "#ADMIN insert_siteinfo Suser=NAME
Surl=URL Brief=KURZ Long=LANG #END"
```

6.7.2 Die Suchabfragen

Die Programmteile des Broker sind durch die Umstellung auf einen anderen Index nicht betroffen. Auch das Benutzerinterface funktioniert weiterhin ohne Änderung. Erst wenn die neue Funktionalität des Suchindex Verwendung finden soll, sind Erweiterungen der Suchanfragesyntax und der Resultatrückgabe notwendig.⁹

Erweiterte Anfragen

Der Broker-Client [HSW96] kann auch Suchanfragen des Benutzers an den Broker schicken. Zusätzlich zum Suchstring kann der Benutzer Parameter mitüberegeben, die das Suchergebnis selbst, dessen Darstellung und Umfang beeinflussen.

```
brkclient HOST PORT "#USER #index Parameterliste #END Anfragestring"
```

Folgende zusätzliche Parameter decken nun die neue Funktionalität des Indexer ab und können der Anfrage hinzugefügt werden:

queryid Durch Angabe dieses Parameters kann das Suchergebnis partitioniert werden. Das Gesamtergebnis wird vom Indexer für geschränkte Zeit zwischengespeichert. Durch Angabe von zwei weiteren Zahlen, werden die gewünschten Suchergebniszeilen festgelegt.

Beispiel: Der Benutzer mit der selbstgewählten ID "123456" sucht nach Dokumenten mit dem Wort "mars" und will die ersten zehn Dokumente erhalten.

```
..."#USER #index queryid 123456-1-10 #END daten"
```

Nun will er Ergebnis 19 ansehen. Der Anfragestring kann wegelassen werden:

```
..."#USER #index queryid 123456-19-19 #END"
```

relkey Hiermit wird festgelegt, daß die relevanten Keywords eines gefundenen Dokumentes mitüberegeben werden sollen. Dieser Parameter wird gefolgt von zwei Zahlen. Die erste gibt die gewünschte maximale Anzahl der zu übergebenden Keywords an. Durch die zweite Zahl wird der Schwellwert zur Relevanzberechnung (siehe Abschnitt 4.2.1) in Prozent übermittelt.

Beispiel: Maximal fünf relevante Schlüsselwörter sollen pro Dokument mitgeschickt werden. Dabei gilt ein Keyword als relevant, wenn es in nicht mehr als 10% der Dokumente als Schlüsselwort vorkommt.

```
..."#USER #index relkey 5-10 #END"
```

⁸Host und Portnummer des Broker, der abgefragt werden soll.

⁹Auf die nötigen Adaptionen der CGI-Skripts und HTML-Formulare wird nicht weiter eingegangen.

body title keyword Durch die Verwendung dieser Parameter können die jeweiligen SOIF-Attribute verschieden bewertet werden. Dies hat dann Einfluß auf die Gewichtung der Dokumente (Ranking).

Beispiel:

```
..."#USER #index body 1 title 6 keyword 4 #END"
```

siteinfo Im neuen Suchindex kann nicht nur nach Dokumenten sondern auch nach Server- und Sitebeschreibungen gesucht werden, was durch Angabe dieses Parameters erreicht wird.

Beispiel:

```
..."#USER #index siteinfo #END"
```

Auch die Syntax des Suchanfragestrings wurde erweitert. Durch Angabe des Anfangsbuchstabens des SOIF-Attributnames vor dem jeweiligen Suchwort wird die Suche auf das entsprechende Attribut eingeschränkt:

Beispiel: Hier soll "mars" im Titel und "jupiter" als Keyword in den Dokumenten enthalten sein:

```
brkclient localhost 9552 "#USER #END t:mars AND k:jupiter"
```

Erweiterungen bei der Ergebnisrückgabe

Bei einer Suchanfrage wird nun in der Funktion `Resolve_Query` des Indexinterface aus der internen Query-Repräsentation ein SQL-Befehl generiert und mittels Datenbank-API an den Datenbankserver weitergeleitet. Die zurückgelieferten Dateinamen sind gleichzeitig die Objekt-IDs für den Broker. Diese werden an den *Registry Manager* (siehe Abschnitt 3.3) weitergeleitet. Er überprüft nun, ob die Objekte noch gültig sind.¹⁰ Ist dies der Fall, werden URL und gewünschte¹¹ SOIF-Attribut-Inhalte an den Broker-Client geschickt.

Wie können nun aber Ergebnisse einer Suche in der Severzusatzinformation gesendet werden. Es existiert ja weder ein SOIF-Objekt, dessen Inhalte vom Broker gesendet werden könnten noch ein entsprechender Eintrag in der *Registry*? Wie können Zusatzinformationen wie das Dokumentgewicht und die relevanten Schlüsselwörter übertragen werden? Im Indexinterface sind solche zusätzlichen Übertragungen nicht vorgesehen. Daher werden diese Daten direkt über die bestehende Verbindung zwischen Broker und Client geschickt. Folgende Identifier kennzeichnen dabei den Inhalt einer Antwort an den Broker-Client:

siteinfo Hier wird die URL und der Besitzer einer Site zurückgeliefert.

```
siteinfo Surl=URL Suser=BENUTZERNAME
```

¹⁰Diese Überprüfung ist beim neuen Index überflüssig, da abgelaufene Dokumente in der Datenbank tatsächlich sofort gelöscht werden.

¹¹Die Anfrageparameter, welche im ursprünglichen Harvest-Broker definiert sind und z.B. den Umfang des Suchergebnisses festlegen werden in [HSW96] dokumentiert.

weight Das errechnete Gewicht eines gefundenen Dokumentes.

weight GEWICHT

relkey Die relevanten Schlüsselwörter werden übergeben. Hinter jedem Wort steht durch einen Doppelpunkt getrennt dessen Häufigkeit im Dokument.

relkey KEY1:ANZAHL1 KEY2:ANZAHL2 ...

Der Broker-Client gibt diese Zeilen in der gleiche Art auf die Standardausgabe. CGI-Skripts können von dieser lesen und das Gesamtergebnis am Web-Client darstellen.

6.8 Implementierung

Der Indexer wurde auf das Postgres-Datenbanksystem [YC95] (Version 6.3.2) aufgesetzt und in C unter Verwendung des Postgres-API implementiert. Parallel dazu wurden die Datenbankroutinen in *Embedded SQL* entworfen, um eine Portierung auf andere SQL-Datenbanksysteme zu ermöglichen.

6.9 Zusammenfassung

In diesem Kapitel wurde ein neuer, auf einem relationalen Datenbanksystem basierender Suchindex für den Harvest-Broker vorgestellt. Dieser Index kann statt anderer externer Suchwerkzeuge verwendet werden und bietet eine Reihe neuer Möglichkeiten.

Die einzelnen Objekte (Wörter, Dokumente und Server beziehungsweise Sites) werden ebenso wie die Beziehungen dieser Objekte untereinander in jeweils einer Tabelle verwaltet. Zusätzlich werden Eigenschaften der Objekte und der Beziehungen in den entsprechenden Tabellen mitgeführt. So wird zum Beispiel in der Worttabelle festgehalten, wie oft jedes Wort im untersuchten Bereich vorkommt. Die Beziehung zwischen Wort und Dokument enthält zusätzliche Attribute, die Auskunft geben, wie oft das Wort im Dokument in welchem SOIF-Attribut vorkommt.

Der neue Index ist aufgrund der in den einzelnen Tabellen mitgeführten Daten in der Lage, relevante Schlüsselwörter zu filtern und Dokumente entsprechend ihrer Wichtigkeit im Suchergebnis zu reihen. Da die Wörter nicht nur den Dokumenten, sondern auch den Servern, in den sie vorkommen, zugeordnet werden, ist eine zweistufige Suche machbar. Damit wird die Ergebnisliste kompakter und die Suche ist schneller, da im Allgemeinen wesentlich weniger Sites als Dokumente vorhanden sind. Es kann auch nur in den Sitebeschreibungen gesucht werden.

Die Einbettung des neuen Index in den Harvest-Broker verlangte nicht nur die Implementierung der einzelnen Funktionen des Indexinterface, sondern auch Erweiterungen in anderen Bereichen, da die zusätzliche Funktionalität des neuen Index keine Entsprechung beim Broker hat.

Im nächsten und letzten Kapitel werden nach einer Zusammenfassung dieser Arbeit weitere Erweiterungen des vorgestellten Suchsystems vorgeschlagen und mögliche zukünftige Anwendung erörtert.

Kapitel 7

Zusammenfassung und Ausblick

Gängige Suchdienste wie *AltaVista* oder *HotBot* arbeiten sich unabhängig von einander durchs Netz. Sie verfolgen Hyperlinks, laden Dokumente und extrahieren deren Inhalt. Schätzungen gehen davon aus, daß das gesamte Web dreimal täglich durchsucht wird [Bek96]. Die Belastung für das Netz und die Informationsserver schränkt andere Dienste in ihrer Leistung ein. Das Web wächst, so daß dieses Suchkonzept immer problematischer wird. Nicht nur die Auffindung, auch die Suche nach relevanter und adäquater Information wird aufgrund der Größe und Unstrukturiertheit des Informationsangebotes zunehmend schwieriger.

Mit dem in dieser Arbeit vorgestellten Harvest-System wird ein anderer Ansatz verfolgt. Der Harvest-Gatherer durchsucht einen Informationsserver lokal auf dessen Hostrechner. Er extrahiert Dokumenteninformation und gibt komprimierte Inhaltsbeschreibungen an einen Suchindex (Harvest-Broker) über das Netz weiter. Diese Strategie reduziert somit Netz- und Serverlast. Das Harvest-System geht nun noch einen Schritt weiter und ermöglicht Brokern, ihre Daten untereinander auszutauschen. So können Suchdienste einem geographischen oder thematischen Bereich zugeordnet werden und ihre indizierten Daten an übergeordnete, umfassenderer Dienste weiterreichen. Es lassen sich so hierarchische Suchsysteme aufbauen. Diese Kooperation einzelner Broker verhindert, daß ein Web-Bereich von mehr als einem Gatherer durchsucht wird. Dies macht das Gesamtsystem effizienter.

Eine beschriebene Schwachstelle des Systems ist der HTML-Parser, der sich strikt an die DTD-Definition von HTML hält. Web-Browser verhalten sich toleranter, so daß es zu Unterschieden zwischen angezeigten und indizierten Dokumenten kommt. Dokumentenzusatzinformationen können in Ermangelung einer geeigneten Schnittstelle nicht in die Inhaltsbeschreibung einfließen. Der Harvest-Broker verwaltet seinen Suchindex mittels externer Werkzeuge. Diese bieten keine besonderen *Retrieval*-Möglichkeiten. Relevante Schlüsselwörter können nicht erkannt, ein Ranking gefundener Dokumente nicht durchgeführt werden.

Im Rahmen dieser Diplomarbeit wurde ein HTML-Konverter und ein Suchindex auf der Basis einer relationalen Datenbank entwickelt. Diese ersetzen die entsprechenden Module des Harvest-Systems. Der neue HTML-Konverter wurde dem Verhalten der Web-Browser angepasst und ist somit fehlertoleranter. Er verfügt über Schnittstellen

zu externen Modulen, die zusätzliche Dokumenteninformation (Sprache, Kurzzusammenfassung) in die Inhaltsbeschreibung einbringen können. Der Suchindex ermöglicht eine Reihe von *Retrieval*-Verfahren, wie das Ranking von gefundenen Dokumenten, die Keyword-Relevanz-Filterung und die zweistufige Suche. In den neuen Index können auch Server- und Sitezusatzinformationen, die der Informationsanbieter bei der Anmeldung seines Bereiches angibt, aufgenommen werden.

In einer weiteren Ausbaustufe könnten durch Anpassungen und Erweiterungen der Benutzerschnittstelle die Möglichkeiten des neuen Indexers weiter ausgeschöpft werden. So wären auf Server- und Sites beschränkte Suchanfragen und die Gewichtung einzelner Suchterme denkbar. Auch die Ergebnispräsentation müßte die zusätzlichen Dokument- und Serverinformationen berücksichtigen. Wünschenswert ist eine zweistufige Ergebnisdarstellung, bei der zuerst lediglich die Informationsserver samt Beschreibung aufgelistet werden. Erst bei entsprechender Anwahl werden dann die der Suchanfrage genügenden Dokumente angezeigt.

Stemming-Verfahren und ein Thesaurus könnten das System ergänzen. Bei der hyperrelationale Suche, also die Suche nach Begriffgruppen über die Dokumentgrenzen hinweg, müßte in einer weiteren Tabelle der Index-Datenbank die zu einem Dokument gehörenden Hyper-Links verwaltet werden. Erfüllt ein Dokument im Index nur einen Teil der (konjunktiven) Suchanfrage, könnte optional in den über Links verbundenen Dokumenten nach dem noch nicht gefundenen Wörtern gesucht werden und schließlich auch die so einer Anfrage genügenden Dokumente dem Suchergebnis hinzugefügt werden. Dadurch wird versucht, die teilweise willkürliche physikalische Trennung von Informationsteilen zu überwinden.

Viele Verwaltungs- und Kontrollaufgaben des Harvest-Broker sind durch die Anbindung an den neuen Index überflüssig geworden. Daher könnte der Broker gänzlich durch ein Teilsystem ersetzt werden, das die Schnittstelle zwischen Datenbankindex und Gatherer beziehungsweise Benutzer auf einfachere und effizientere Weise realisiert.

Module für die Spracherkennung könnten in den HTML-Konverter unter Verwendung der in den Abschnitten 5.3 und 5.1.2 beschriebenen Methoden integriert werden.

Administrations-Tools für die Verwaltung von Serveranmeldungen sollten in das Harvest-System aufgenommen werden. Angemeldete Bereiche würden zum Beispiel auf einem kleinen Testsystem indiziert und nach einer Begutachtungsphase in das eigentliche Suchsystem übernommen.

Mögliche zukünftige Anwendungen: Das in dieser Arbeit vorgestellte und erweiterte Suchsystem verbindet Dokumenteninhalte mit Serverinformation, die vom Anbieter bei der Anmeldung angegeben wird. Expertengruppen könnten nun die angemeldeten Bereiche hinsichtlich Art und Qualität des gebotenen Inhaltes bewerten und diese Web-Bereiche unterschiedlichen thematischen und qualitativen Kategorien zuordnen. Diese ließen sich jeweils von einem Broker verwalten. Übergeordnete Broker könnten darauf aufbauend verschiedene Stufen der Qualität und der thematischen Kategorie der angebotenen Information unterscheiden. Benutzerprofile könnten in weiter Folge dazu dienen, dem Suchenden relevante und adäquate Inhalte zu liefern. Bei umfangreichen

oder länger dauernden Suchen wäre es denkbar, die Ergebnisse mittels E-mail zu einem späteren Zeitpunkt zuzustellen. Durch Einbettung eines *Pricing System* könnten Leistungen des Suchdienstes aufwandbezogen verrechnet werden. Ausgewählte Benutzer (Experten), die Teile des Informationsangebotes bewerten und kategorisieren, bekämen ihren Aufwand gutgeschrieben.

Ein solches System kann nun Information auffinden, kategorisieren und für die Wiederauffindung bereitstellen. Dokumente werden dadurch wiederverwendbar. Sie ließen sich zu bestimmten Themen mit bestimmter Qualität bestellt beziehungsweise abonnieren. Es wäre denkbar, daß Veränderungen von Inhalten den Abonnenten bekanntgemacht werden. Dem Autor würde mitgeteilt werden, wer seine Dokumente wiederverwendet. Durch Einbeziehung von Versionsverwaltungsverfahren könnten ältere Versionen weiter zugänglich gemacht werden, solange Referenzen darauf bestehen.

Die Vertrautheit des Suchsystems mit dem Angebot des von ihm durchsuchten Bereiches, die Bewertung und Kategorisierung der Inhalte stellen eine wesentliche Voraussetzung für dessen Verwendung als Teil von Wissenssystemen dar. So könnte die *Web-Based-Training*-Umgebung von Hyperwave (GENTLE¹) [DM98] ein solches Suchsystem als dynamische Hintergrundbibliothek verwenden. Hier könnte zusätzlich durch Aufnahme von Diskussionsbeiträgen und Bemerkungen eine dynamische Wissensdatenbank aufgebaut werden. [DGM⁺98]

Die Anbindung von *Information Agents* könnte die Verwendung von plattformunabhängig implementierten Verfahren zum Auffinden, Filtern und Verwalten von Information erlauben. Das Harvest-Suchsystem wären um eine Anbindung an *Mobile Agents*, die sich individuell an die jeweiligen Gegebenheiten am Hostrechner des Informationsserver anpassen würden, zu erweitern. Auch die Benutzerschnittstelle könnte um eine *Agent*-Anbindung ergänzt werden. Die Benutzer würde bei der Angabe von Suchanfragen von *Interface Agents*, welche vom Verhalten und Feedback des Benutzers lernen und dessen Instruktionen ausführen, unterstützt. [NN98]

¹GEneral Networked TRaining and LEarning Environment

Abbildungsverzeichnis

2.1	Quorum-Level: Beispiel einer Query-Hierarchie	7
3.1	Ineffizienz gewöhnlicher Suchsysteme	14
3.2	Lokal und über das Netz arbeitende Gatherer	15
3.3	Das SOIF-Format: Formale Beschreibung in BNF	16
3.4	Eine HTML-Datei und ein daraus generiertes SOIF-Objekt	17
4.1	Zwei Dokumente: Wörter und ihre Bedeutung für das Dokument	34
5.1	Konfigurationsdatei: Ausschnitt SOIF-Tabelle	48
5.2	Konfigurationsdatei: Filter und Module	49
5.3	Konfigurationsdatei: Separatoreinträge	50
5.4	Beispiel einer kompletten Konfigurationsdatei	55
6.1	Beispiel für die Verletzung der zweiten Normalform	59
6.2	Beispiel für die Erlangung der zweiten Normalform	59
6.3	Worttabelle - WORT-T	60
6.4	Dokumenttabelle - DOC-T	61
6.5	Wort-Dokumenttabelle - WD-T	61
6.6	Site-Tabelle - SITE-T	62
6.7	Wort-Site-Tabelle - WS-T	62
6.8	Wort-Site-Info-Tabelle - WS-T	63
6.9	Prinzip des "Join"-Operators: "Join" angewandt auf die drei Tabellen aus Abbildung 6.2	68
6.10	Tabelle aus Abbildung 6.9 nach Anwendung der Where -Klausel, welche die Gleichheit der IDs verlangt	68

Tabellenverzeichnis

2.1	Suchdienste in Zahlen	10
-----	---------------------------------	----

Größenverzeichnis

Größe	Einheit	Beschreibung	Formel
Q_{\vee}	–	Disjunktives Query	4.2
Q_{\wedge}	–	Konjunktives Query	4.2
d_{Ax}	–	Gewicht des Wortes x im Dokument A	4.6 4.5 4.6
a_1	–	Gewicht des Suchwortes x	4.6 4.5 4.6
$SIM(Q, A)$	–	Ähnlichkeit zwischen Query Q und Dokument A	4.6 4.5 4.6

Literaturverzeichnis

- [BCL⁺94] Tim Berners-Lee, Robert Cailliau, Ari Luotonen, Henrik Frystyk Nielsen und Arthur Secret. The World Wide Web. *Communications of the ACM*, Volume 37, (August 1994), S. 76–82.
- [BDH⁺94] C. Mic Bowman, Peter B. Danzig, Darren R. Hardy, Udi Manber, Michael F. Schwartz und Duane P. Wessels. Harvest: A Scalable, Customizable Discovery and Access System. Technical Report CU-CS-732-94, Department of Computer Science, University of Colorado, Boulder, (1994).
- [Bek96] Bernhard Bekavac. Der findet. WWW-Suchmaschinen und Katalog. *iX*, Nummer 7, (1996), S. 102–110.
- [BMM94] T. Berners-Lee, L. Masinter und M. McCahill. Uniform Resource Locators. <ftp://ds.inernic.net/rfc/rfc1738.txt>, (Dezember 1994).
- [Bra97] D. Brake. Lost in Cyperspace. *New Scientists, IPC Magazines Limited*, <http://www.newscientist.com/keysites/networld/lost.html>, (Juni 1997).
- [Buc96] Lydia Buchmüller. Information Superhighway und Baukunst. *Overload*. Verlag HdA, (1996).
- [Cam94] William G. Camargo. The Harvest Broker. Master’s thesis, Department of Computer Science, Pennsylvania State University, (Dezember 1994).
- [Cov] Robin Cover. SGML Web Page. <http://www.sil.org/sgml/sgml.html>.
- [Dat86] C. J. Date. *An Introduction to Database Systems*. Addison-Wesley, (1986).
- [DGM⁺98] Thomas Dietinger, Christian Gütl, Herman Maurer, Maja Pivec und Klaus Schmaranz. Intelligent Knowledge Gathering and Management as New Ways of an Improved Learning Process. Erscheint in *Proceedings of Web-Net98*, (1998).
- [DM98] Thomas Dietinger und Herman Maurer. GENTLE - General Network Training and Learning Environment. *Proceedings of ED-MEDIA98/ED-TELECOM98*, (1998), S. 274–280.
- [ESM92] E.Fox, S.Betrabet und M.Koushik. Extended Boolean Models. *Information Retrieval: Data Structures and Algorithms*, Prentice-Hall, (1992), S. 393–418.

- [FB92] William B. Frakes and Ricardo Baeza-Yates, Hrsg. *Information Retrieval: Data Structures and Algorithms*. Prentice-Hall, (1992).
- [GAM98] Christian Gütl, Keith Andrews und Hermann Maurer. Future Information Harvesting and Processing on the Web. Erscheint in *European Telematics: Advancing the Information Society*, (Februar 1998).
- [GDN⁺98] Christian Gütl, Thomas Dietinger, Dietmar Neussl, Bernhard Knögler und Klaus Schmaranz. Dynamic Background Libraries as an improved Way for Web-Based Learning using HKS (Hierarchical Interactive Knowledge System). Erscheint in *Proceedings of ICCE98*, (Oktober 1998).
- [Gra97] Graphics, Visualisation and Usability Center. GUV's 7th WWW User Survey. Technical Report, College of Computing, Georgia Institute of Technology, Atlanta, (Juni 1997).
- [Gro] Web Design Group. ISO 8859-1 Character Set Specification. <http://www.iso.org/>.
- [GS98] Christian Gütl und Klaus Schmaranz. DigLib 2000 - A Working Prototype for Next Generation of Digital Libraries. Technical Report, Institute for Information Processing and Computer Supported New Media (IICM), University of Technology Graz, (Februar 1998).
- [HaE96] Die Geschichte des Internet. *Internet Magazin*, S. 100–103, (1996).
- [Har92] Donna Harman. Ranking Algorithms. *Information Retrieval: Data Structures and Algorithms*. Prentice-Hall, (1992) S. 363–392.
- [HS95] Darren R. Hardy und Michael F. Schwartz. Customized Information Extraction as a Basis for Resource Discovery. Technical Report, Department of Computer Science, University of Colorado, Boulder, (1995).
- [HSW96] Darren R. Hardy, Michael F. Schwartz und Duane P. Wessels. *Harvest User's Manual*, (1996).
- [Hyp98] *Hyperwave Installation Guide Version 2.6*, (1998).
- [JJ97] M. Jeusfeld and M. Jarke. Suchhilfe für das World Wide Web - Funktionsweise und Metadatenstruktur. *Wirtschaftsinformatik*, Nummer 39, (1997).
- [Koc96] Traugott Koch. Suchmaschinen im Internet, Weiter auf dem Weg zur virtuellen Bibliothek! <http://www.ub2.lu.se/tk/demos/D09603-manus.html>, (1996).
- [Mau96] Hermann Maurer, Hrsg. *HyperWave The Next Generation Web Solution*. Addison-Wesley, (1996).
- [Net96] Network Wizards: Internet Domain Survey. <http://www.nw.com/zone/WWW/report.html>, (1996).

- [NK94] Christian Neuss und Robert E. Kent. Conceptual Analysis of Resource Meta-Information. <http://www.fhg.de/www/ww95/proceedings/papers/94/ww3.html>, (1994).
- [NMW98] Bonnie A. Nardi, James R. Miller und David J. Wright. Coolaborative, Programmable Intelligent Agents. *Communications of the ACM*, Volume 41, (März 1998), S. 96–104.
- [NN98] H. S. Nwana and D. T. Ndumu. A Brief Introduction to Software Agent Technology. *Agent Technology*. Springer, (1998), S. 29–47.
- [Rag] Dave Raggett. HTML 3.0 Draft Specification. <http://www.w3.org/hypertext/WWW/Markup/html3/CoverPage.html>.
- [SFW83] G. Salton, E. A. Fox und H. Wu. Extended Boolean Information Retrieval Systems. *Communications of the ACM*, Volume 26, (Dezember 1983), S. 1022–36.
- [Sul98a] Danny Sullivan. How Big Are The Search Engines ? <http://www.searchenginewatch.com/size.html>, (1998).
- [Sul98b] Danny Sullivan. Search Engine Feature Chart. <http://www.searchenginewatch.com/features.html>, (1998).
- [Wes95] D. Wessels. Thoughts and Proposals on the Second-Generation SOIF Syntax. <http://blaur.net/~wessels/Harvest/soif2.html>, (Februar 1995).
- [WM93] Sun Wu und Udi Manber. Glimpse: A Tool to Search Through Entire File Systems. Technical Report TR 93-34, Department of Computer Science, University of Arizona, Tucson, (1993).
- [WMB94] Ian H. Witten, Alistair Moffat und Timothy C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Pictures*. Van Nostrand Reinhold, (1994).
- [YC95] Andrew Yu and Holly Chen. *The Postgres95 User Manual*, (1995).