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 Neusl

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

September 1998

© Copyright 1998, Dietmar Neusl

Betreuer: Dipl.-Ing. Christian Glttl
Begutachter: a.Univ.-Prof. Dr. Dr.h.c. Hermann Maurer
Kurzfassung


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 Außereiten 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 Benutzeroberfläche ......................................................... 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 Automatisierte 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
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 Bibliotheken, 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 ARPANET\[1\] eingeführt, das den Informationsaustausch unter den Stützpunkten ermöglichte. Neben diesem entstand das CSNET\[2\], das Forschungseinrichtungen und Industriezentren vernetzte. Daraus entwickelte sich schließlich ein Netzwerk, das sämtliche Universitäten, Forschungs- und Entwicklungsinstanzen miteinander verband. Die allgemeine Nutzung des Internet wurde erst in den Jahren um 1983 mit der Integration des Internet-Protokolls TCP/IP\[3\] in die Berkley UNIX-Distribution eingeleitet.\[HaE96\]

1989 begann Tim Berners-Lee bei CERN\[4\] in Genf mit der Entwicklung eines Informationsystems 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-

---

1. Advanced Research Projects Agency
2. Computer Science Network
3. Transmission Control Protocol / Internet Protocol


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
Die Ergebnisse dieser Untersuchung führen in weiterer Folge - im Gestaltungsbe-
reich - 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 Erzeu-
gung 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 Ähnlichkeitsverfah-
ren⁷ 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 des 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 Verwen-
dung als Hintergrundbibliothek für die *Web-Based-Training*-Umbgebung 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.

---

⁷ siehe 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 gültige Ansätze, Verfahren bzw. Systeme zur Wissensauflistung 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


---

\(^1\)Auf diese Verfahren wird im Abschnitt 2.1.2 näher eingegangen
KAPITEL 2. SUCHDIENSTE

Anmelden von Informationseinheiten

Hier meldet der Autor seine Dokumente bei einem Suchsuchdienst 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


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


boolsche Anfragemethode: 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 einen 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.

<table>
<thead>
<tr>
<th>Gewicht</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>$(A \land B \land C \land D)$</td>
</tr>
<tr>
<td>3</td>
<td>$(A \land B \land C) \lor (A \land B \land D) \lor (A \land C \land D) \lor (B \land C \land D)$</td>
</tr>
<tr>
<td>2</td>
<td>$(A \land B) \lor (A \land C) \lor (A \land D) \lor (B \land C) \lor (B \land D) \lor (C \land D)$</td>
</tr>
<tr>
<td>1</td>
<td>$(A \lor B \lor C \lor D)$</td>
</tr>
</tbody>
</table>

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.

**Nachbearbeitung**\(^2\): 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 auszuscheiden zu können, bedient man sich einer **Stopliste**.
- Verschiedene Wörter 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

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

- 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.


### 2.2 Einteilung von gängigen Suchdiensten


- 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/]

KAPITEL 2. SUCHDIENSTE

- Katalogdienste mit aktivem Anmelden und Redaktionsteams
  - Yahoo  http://www.yahoo.com/
  - Web.de  http://www.web.de/
  - Dinoonline  http://www.dino-online.de/
  - Henkel  http://www.henkel.co.at/henkel/ha_www_1.html

- Metasuchdienst und Kombinationen
  - Metasuchdienst durch Nutzung mehrerer Suchdienste
    - 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 Suchsys-
ystemen aufzeigen. Jedes Verfahren bietet in Teilbereichen Vorteile. Diesen stehen mehr
oder weniger ausgeprägte Schwächen 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 aufge-
bauten Kategorien und bietet so dem Benutzer die Möglichkeit, den Zielbereich seiner
Suche schrittweise einzuschränken. Katalogsuchdienste können aufgrund ihres Konzep-
tes 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 Tricks⁸ erreicht werden, daß Dokumente bei der Ergebnis-
anszeige unangemessen weit vorne plaziert 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 geogra-
phischen Deckungsbereich. So indizieren Magellan und WWW-Worm keinen Volltext,
Web.de und Dino.online nur deutsche Quellen.

⁸z.B. Spanning. Hierbei werden nicht sichtbaren Wörtern im HTML-Dokument zum Zwecke der
Manipulation von Ranking-Verfahren untergebracht.

### 2.3 Schwachpunkte gängiger Suchdienste

#### 2.3.1 Netzwerk- und Serverbelastung


<table>
<thead>
<tr>
<th>Name</th>
<th>Anzahl indizierter Seiten</th>
<th>Seiten pro Tag</th>
<th>Update-Rate⁵</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alta Vista</td>
<td>140 Millionen</td>
<td>10 Millionen</td>
<td>1 Tag bis 1 Monat</td>
</tr>
<tr>
<td>HotBot</td>
<td>110 Millionen</td>
<td>bis 10 Millionen</td>
<td>1 Tag bis 2 Wochen</td>
</tr>
<tr>
<td>Lycos</td>
<td>30 Millionen</td>
<td>6 bis 10 Millionen</td>
<td>2 bis 3 Wochen</td>
</tr>
</tbody>
</table>


⁴http://www.verity.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 diesen Problembereich 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 Änderungsraten am Server ab. Je öfter der Bereich abgesucht wird, desto konsistenter sind die Daten des Suchdienstes. Das Suchprogramm stellt weiter 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 unterer Ebene stehen die lokalen Server, die die vollständige Erfassung ihrer Daten sicherstellen müssen. Darüberliegende Suchdienste fassen Informationen mehrerer Server zu Wissenschustern 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 müssen, 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.


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) vorbehalt-

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


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 Bereich neuerlich durchsucht werden muß.


Zusätzlich zur Suche über die Metainformation ist jedes Textdokument über Voll- texte durchsuchung 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

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 den Benutzer zu beliebiger Zeit ausgehört werden (Konzept der vordefinierte Suche). Es wäre naheliegend, die vom System indizierten Daten in geeigneter Form\(^8\) auch anderen Suchsystemen zugänglich zu machen.

### 2.5 Zusammenfassung


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.


\(^8\) Zum 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-System\(^1\) [BDH+94] stellt eine Sammlung von Werkzeugen zur Auffindung, Außbereitung, Verwaltung und Suche von Informationen im Internet dar. Es kann Daten aus den verschiedensten Dokumentformaten (HTML, \TeX, Tar- und Zipfiles, etc.) extrahieren. Zur Indizierung verwendet das Harvest-System externe Werkzeuge wie zum Beispiel Glimpse [WM93].

\begin{center}
\begin{tikzpicture}
  \node (index) {Index};
  \node [below of=index] (index2) {Index};
  \node [right of=index] (index3) {Index};
  \node [below of=index2] (index4) {Index};
  \node [below of=index3] (index5) {Index};
  \node [left of=index, xshift=-2cm] (server1) {InfoServer};
  \node [right of=index, xshift=2cm] (server2) {InfoServer};
  \node [below of=index, yshift=-2cm] (server3) {InfoServer};
  \draw [->] (index) -- (index2);
  \draw [->] (index) -- (index3);
  \draw [->] (index) -- (index4);
  \draw [->] (index) -- (index5);
  \draw [->] (index2) -- (server1);
  \draw [->] (index2) -- (server3);
  \draw [->] (index3) -- (server2);
  \draw [->] (index3) -- (server3);
  \draw [->] (index4) -- (server1);
  \draw [->] (index4) -- (server2);
  \draw [->] (index5) -- (server1);
  \draw [->] (index5) -- (server2);
\end{tikzpicture}
\end{center}

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

\footnote{http://www.harvest.cs.edu}

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


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 Summary Object Interchange Format wird in später in diesem Abschnitt beschrieben.


Das Metadatenformat SOIF

Wie schon eingangs erwähnt, generiert der Gatherer für jedes aufgefundene Informati-onsobjekt eine Inhaltszusammenfassung (Content Summary). Diese enthält Attribut-Wert-Paare und wird im sogenannten Summary Object Interchange Format (SOIF) gespeichert. SOIF basiert auf einer Kombination von IAFA-Templates und BibTeX.

<table>
<thead>
<tr>
<th>SOIFSTROM</th>
<th>→ OBJEKT</th>
<th>OBJEKT SOIFSTROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBJEKT</td>
<td>→ @TYPE</td>
<td>{URL AW-LISTE}</td>
</tr>
<tr>
<td>AW-LISTE</td>
<td>→ AW-LISTE</td>
<td>AW-PAAR AW-LISTE</td>
</tr>
<tr>
<td>AW-PAAR</td>
<td>→ ATTRIBUT {GRÖSSE} BEGRENZER WERT</td>
<td></td>
</tr>
<tr>
<td>ATTRIBUT</td>
<td>→ Zeichenkette</td>
<td></td>
</tr>
<tr>
<td>WERT</td>
<td>→ Zeichenkette</td>
<td></td>
</tr>
<tr>
<td>BEGRENZER</td>
<td>→ :&lt;tab&gt;</td>
<td></td>
</tr>
</tbody>
</table>

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


3Hier wird ein bestimmter Bereich, ein Infocluster, 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.
Abbildung 3.4: Eine HTML-Datei und ein daraus generiertes SOIF-Objekt


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.

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“ 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


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 enormer 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 neu erlich 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


**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 typenbehafeten 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

---

6*Object 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-Systems für den Fall von HTML-Dateien fun-
ktioniert 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 URI, der Objekte individuell festlegen (leave node)
oder einen Startpunkt (root node) mit einer maximalen Verzweigungstiefe angeben.
Zusätzlich können Stoplisten mit URLs von Dokumenten und Sites 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 Admini-
strator kennt beide Bezeichnungen, den lokalen Pfad und die Entsprache für HTTP
oder FTP, In der Konfigurationsdatei kann er für beliebige URLs den lokalen Pfad ver-
merken. 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 ermit-
telt, nachdem der Inhalt behandelt wurde (siehe Abschnitt 3.2.1), die darin enthaltenen
Links. Mit den Zielsendungen 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 Indizie-
rung durch externe Werkzeuge. Dem Benutzer stellt er mit Hilfe eines Web-Clients und
eines CGI-Skripts eine Schnittstelle zur Kommunikation mit dem Suchsystem bereit.
KAPITEL 3. DAS HARVEST-SUCHSYSTEM

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 Brokers 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-Stream, 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 einfliess.


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 Brokers nach immer am Informationsserver, wird seine Lebensdauer neu berechnet. Falls dieser Eintrag abläuft, weil das korrespondierende Dokument am Informationsserver

---

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

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

9 Aspekte 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 Metadaten\(^\text{10}\) 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 Informationquelle 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 Interfaces. Einige Indexer indizieren objektwise, 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.

\(^\text{10}\)zum Beispiel bei HTML: `<META NAME="expires" CONTENT="DEC 09 1998">`
Zum Beispiel kann ein Indizierwerkzeug, 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 Forms\textsuperscript{11} verwenden. Sie stellen die grafische Benutzerschnittstelle (GUI\textsuperscript{12}) dar.
- einem CGI\textsuperscript{13}-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 diesen 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. Glimpse\textsuperscript{14} [WM93] kann folgende Anfragetypen verarbeiten:

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

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

\textsuperscript{12}Graphical User Interface

\textsuperscript{13}Common Gateway Interface

\textsuperscript{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


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 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 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


Broker können ihre Daten austauschen. So kann ein kooperierendes verteiltes Suchsystem aufgebaut werden. Einzelne Broker können speziellen Themen zugeordnet werden und ihre Daten an übergeordnete 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 Summarizer, der die Konvertierung der HTML-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 SGML-Anwendung [Cov], die mittels einer DTD definiert ist [Gro]. Die Markup-Befehle (Tags) werden von spitzen Klammern 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
KAPITEL 4. VERBESSERUNGSPOTENTIAL AM HARVEST-SYSTEM

eingeschlossen werden, wenn diese Sonderzeichen beinhalten oder bei 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.

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


Anpassung an das Verhalten der Web-Browser

Der HTML-Parser des Gatherer hält sich strickt 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

Konfigurierbarkeit und Fehlertoleranz

Der Parser kann so konfiguriert werden, daß er einzelne Tag-Inhalte ignoriert [HSW96]. \texttt{<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 \texttt{<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.


4.1.2 Modifikation der SOIF-Attribute


- 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-Attributetes modifiziert werden. Zum Beispiel wäre es wünschenswert, nur dann die aus dem Text extrahierten Schlüsselwörter zu verwenden, wenn keine durch das \texttt{<META>-Tag} gewonnen wurden.

\textsuperscript{6}Nachbearbeitung 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.


**Metadatengenerierung**


**Automatische Zusammenfassung**

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.


**Spracherkennung**


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).


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


Der Index unterliegt ständigen Veränderungen und so kann die Worthäufigkeit eines Wortes einmal unterhalb und einmal oberhalb des Grenzwertes liegen. Daher müssen alle Worteinträge mitgeprüft werden und entsprechend der Worthäufigkeit ein- und ausgebildet 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 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

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

7zum Beispiel innerhalb eines Brokerbereiches
Boolesche Systeme erzielen gute Ergebnisse in bezug auf \textit{Recall} \footnote{Recall gibt das Verhältnis zwischen gefundenen relevanten und alle relevanten Dokumenten an.} und \textit{Precision} \footnote{Precision gibt das Verhältnis zwischen relevanten gefundenen und alle gefundenen Dokumenten an.}, wenn die Anfrage exakt formuliert wird.

Folgende Schwächen dieser Systeme werden in [ESM92] erläutert:

\begin{itemize}
  \item Die booleschen Operatoren sind strikt. Zum Beispiel liefern Suchabfragen der Form \texttt{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"\footnote{Durch Umwandlung in die Form \texttt{A and (B and C and D)} oder \texttt{(A and B and C and E) or (A and B and D and E) ...} ließe sich dies auf sehr kompliziertere Weise erreichen.}.
  \item Das boolesche Modell gibt keine Auskunft über die Wichtigkeit der einzelnen gefundenen Dokumente.
  \item Das boolesche Modell bietet keine Möglichkeit, einzelne Suchtermen nach ihrer Bedeutung zu bewerten.
\end{itemize}

Das nachfolgend beschriebene Ranking-Verfahren fällt das Retrieval verzichten auf boolesche Operatoren und scheinen somit benutzerfreundlicher zu sein [Har92].

\textbf{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 widerspiegelt, 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.\footnote{Dieses Verhalten kann von Web-Autoren mit"rsbraucht werden, indem sie Wörter nur zum Zecke des besseren Ranking oft sinnlos aneinander reißen (Spamming.)}

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

\textbf{Ein Beispiel:}

gegeben seien zwei Dokumente \texttt{A} und \texttt{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 \((\text{Zug},0.7) \ (\text{Wagen},0.1))\) ergibt so \((\text{A},0.35)\); \((\text{B},0.22)\). Das Dokument \texttt{A} ist somit relevanter. Verschiedene Ranking-Algorithmen werden in [Har92] beschrieben.
Das bisher beschriebene Konzept läßt sich nun um die Verwendung von booleschen 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 Suchtermen der Anfrage ist. Die Koordinaten entsprechen dem Gewicht der Termen\(^{12}\). Die Koordinaten liegen im Intervall \([0, 1]\). Ein Dokument, das durch den Punkt \((0, 0, \ldots, 0)\) repräsentiert wird, enthält keinen Suchterm. Der Punkt \((1, 1, \ldots, 1)\) steht hingegen für ein Dokument, das alle Suchtermine mit dem maximalen Gewicht enthält. Somit ist es möglich, gefundene Dokumente bezüglich ihrer Distanz zu diesen beiden Punkten zu reihen. Während die boolesche Konjunktion verlangt, daß alle Termen im Dokument auftreten (Optimum \((1, 1, \ldots, 1)\)), werden Dokumente in diesem Modell um so besser bewertet, je näher sie am Optimum liegen. Das Vorkommen aller Termen 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 \(A_1, A_2, \ldots, A_n\) die Suchwörter und \(a_1, a_2, \ldots, a_n\) deren, vom Benutzer vergebenen Gewichte. Durch \(p\) sei angegeben, wie genau sich der boolesche 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 booleschen Operatoren \((\land, \lor)\) bei der Berechnung der Ähnlichkeit und liegt im Intervall \((1, \infty)\). Er wird bei der Konfiguration des Ranking-Systems festgelegt. Suchfragen \((Q)\) mit \(n - 1\) Konjunktionen \((Q_{\land p})\) beziehungsweise \(n - 1\) Disjunktionen \((Q_{\lor p})\) haben in diesem Modell folgende Form:

\[
Q_{\lor p} = (A_1, a_1) \lor_p (A_2, a_2) \lor_p \ldots \lor_p (A_n, a_n)
\]

\(^{12}\)Suchwörter der Anfrage bzw. Wörter im Index
\[ Q_{NP} = (A_1, a_1) \land_p (A_2, a_2) \land_p \cdots \land_p (A_n, a_n) \quad (4.2) \]

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

\[ Q_{\neg P} = -Q \quad (4.3) \]

Es seien nun weiters \(d_{A_1}, d_{A_2}, \ldots, d_{A_n}\) die Gewichte der Wörter im Index. Die Ähnlichkeitsfunktion \(SIM(Q, D)\) zwischen Anfragen \((Q_{NP}, Q_{NP}, Q_{\neg P})\) und Dokument \((D)\) errechnet sich im P-Modell aus:

\[ SIM(Q_{NP}, D) = \sqrt{a_1^p d_{A_1}^p + a_2^p d_{A_2}^p + \cdots + a_n^p d_{A_n}^p \over a_1 + a_2 + \cdots + a_n} \quad (4.4) \]

\[ SIM(Q_{NP}, D) = 1 - \sqrt{a_1^p (1 - d_{A_1})^p + a_2^p (1 - d_{A_2})^p + \cdots + a_n^p (1 - d_{A_n})^p \over a_1 + a_2 + \cdots + a_n} \quad (4.5) \]

\[ SIM(Q_{\neg P}) = 1 - SIM(Q, D) \quad (4.6) \]

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 Suchwörter \((a_1, a_2, \ldots, a_n)\) in der Anfrage gehen die einzelnen Koordinaten der Dokumente \((d_{A_1}, d_{A_2}, \ldots, d_{A_n})\) 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 gefundener Dokument und Suchanfrage sind, desto wichtiger ist dieses im Gesamtergebnis.

**Beispiel 1:**
Die Suchabfrage \(Q ((Zug, 0.4) \land (Straße, 0.3))\) sei nun durch den Vektor \((0.4, 0.3)\) dargestellt. Es sei weitere \(p = 2\). 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:

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

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: \(((Zug, 0.5) \land (Straße, 0.1))\). \(p = 2\). 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:

\[
SIM(Q, D) = 1 - \sqrt{0.5^2(1 - d_{zug})^2 + 0.1^2(1 - d_{strasse})^2}
\]

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.


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 Gatherers, 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 Gesamthinhaltes 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 Gesamtdindex 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 Benutzenschnittstelle

Multimediaobjekte und Skripts


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

Zweistufige Ergebnisaufbereitung


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 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 ergebenenden neuen Möglichkeiten für die Suche und Ergebnisdarstellung sind Thema des Kapitels 6.
Teil II

Gestaltungsbereich
Gestaltungsbereich


Zwei wesentliche Systemteile sind von Veränderungen betroffen:


Kapitel 5

Der HTML-nach-SOIF-Konverter


Gesucht ist also ein fehler tolerant 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


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-Tag <META NAME="Attributname" CONTENT="Attributwert">
Attributen zu einem vollständigen SOIF-Objekt. Der Konverter besteht aus drei Dateien:


5.1.1 Konfigurierbarkeit


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 nächsthöheren 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 

   <A HREF="../index.html">Start</A>


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:


- 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


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


Durch vortäuschend 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-
fen. Wurden zum Beispiel beim Parsen Schlüsselwörter aus dem HTML-Text und aus dem `<META>`-Tag gewonnen, können abhängig von den Konfigurationseinträgen, bei
de Schlüsselwörter mengen 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 Ver-
halten des Parsers großteils mit jenem der gängigen Webbrowser decken soll. Der hier
vorgestellte Parser des Konverters ist insofern fehler tolerant, 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 anderer-
seits 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 anzeigendes 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
KAPITEL 5. DER HTML-NACH-SOIF-KONVERTER

46
durch seinen Namen “&szlig;” oder seinen dezimalen Code “&amp;#223;” im HTML-
Dokument repräsentiert werden, damit auch Zeichen, die nicht über die Tastatur er-
reichbar sind, in ein Dokument aufgenommen werden können.6 Diese verschiedenartig-
tigen 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 ent-
sprechende 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 Be-
ginn einer Zeile. Nach einem Trennzeichen (Tabulator) folgen die zugeordneten SOIF-
Attribute, die untereinander durch ein Leerzeichen getrennt sind. Die Tabelle hat somit
two 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 angege-
benen Tag-Attribute-Kombination “A HREF”

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

<NOTAG>: Mit diesem Schlüsselwort wird eine fehlende Definition bezeichnet. Somit läßt sich bestimmen, was mit Textteilen vor einem eventuellen <BODY>-Tag passiert. Beispiel:

Text ist außerhalb jeder HTML-Definition. <BODY>Hallo...

Der Inhalt der zweiten Spalte der Tabelle gibt an, welche SOIF-Attribute den in der ersten Spalte beschriebenen Textteilen und HTML-Attributinhalten zugeordnet werden sollen.

Die folgende Muster bezeichnen SOIF-Attribute:

soifattribut steht für den Namen des SOIF-Attributes, das beschrieben werden soll.


PARENT : Mit diesem Schlüsselwort wird bestimmt, daß der HTML-Inhalt, der durch die Tabelle zugeordnet wurde, auch den SOIF-Attributen der nächstäußerren HTML-Definition übergeben werden muß.

IGNORE : Dieses Schlüsselwort bestimmt, daß der Inhalt der HTML-Definition ignoriert wird. Es darf nur allein in der Ziepspalte stehen.


Erläuternde Beispiele

Die folgenden Beispiele sollen die Funktionsweise der HTML-SOIF-Tabelle näher veranschaulichen. Berücksichtigung findet hierbei die Tabelle aus Abbildung 5.1.

- Das <TAG>-Muster

  <title>Der Wagen</title>  \[\rightarrow\] title[9]: Der Wagen

- Das <TAG:ATTRIBUTE>-Muster

  <a href="welcome.html">Willkommen</a>  \[\rightarrow\] url-references[12]:

  welcome.html

  keywords[10]: Willkommen

  body[10]: Willkommen
<table>
<thead>
<tr>
<th>SOIF</th>
</tr>
</thead>
<tbody>
<tr>
<td># Text innerhalb der Definitionen werden Attribute zugeordnet.</td>
</tr>
<tr>
<td>&lt;BODY&gt; body</td>
</tr>
<tr>
<td>&lt;NOTAG&gt; body</td>
</tr>
<tr>
<td>&lt;TITLE&gt; title</td>
</tr>
</tbody>
</table>

# Text innerhalb der Definitionen und der nächstäußer Definition werden Attributen zugeordnet.

| <H1> | headings PARENT |
| <A> | keywords PARENT |
| <B> | keywords PARENT |

# Der Inhalt des HTML-Attribute wird einem SOIF-Attribut zugewiesen.

| <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... | url-references{12}: |
  | <a href="welcome.html"> Willkommen</a> | welcome.html |
  | </code>Hallo</body> | body{5}: Hallo |

- **Das Schlüsselwort <NOTAG>**
  
  | Finster <body>war es... | body{...}: Finster war es... |

- **Fixe Zuweisung**
  
  | <script>...</script> | script{10}: JavaScript |
5.2.2 Module und Filter

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


7Die Datei standard.pl enthält einige Filterfunktionen.
Listenoperationen).\footnote{Da Perl keine Variablendeklaration braucht, ist bei der Parameterübergabe an eine Funktion eine Kontextangabe notwendig. Sie gibt Auskunft über den Typ des Parameters.}

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

Abbildung 5.3: Konfigurationsdatei: Separatoreinträge

Werden Namen von SOIF-Attributen mittels Referenz zur Laufzeit generiert (zum Beispiel mit der \texttt{<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 \texttt{keywords} und \texttt{METAkeywords}. Der Inhalt von \texttt{METAkeywords} wurde aus einem \texttt{<META>-Tag} gewonnen, dessen \texttt{NAME}-Attribut den Inhalt "keywords" hat. Hätte der Konverter nicht zwischen den 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 \texttt{<META>-Tags} dem SOIF-Attribut \texttt{keywords} hinzugefügt (\texttt{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 \texttt{keywords} nicht verloren geht, wurde in Abbildung 5.2 \texttt{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-Attributes\footnote{In der Regel wird es als "body" bezeichnet.} Trennzeichen erzwingen. In dieser Sektion werden, nach der Bezeichnung \"[SEPARATOR]\" und einer Zeile mit

Mit diesem Mechanismus läßt sich der Inhalt genau eines SOIF-Attributes gliedern. So können zum Beispiel Absatzmarken ("\texttt{<P>}"; "\texttt{</P>}"), Zeilenumbrüche ("\texttt{<BR>}"), Überschriften ("\texttt{<H1>}"; "\texttt{<H6>}"; "\texttt{</H1>}"; "\texttt{</H6>}") durch eine unterschiedliche Anzahl von Trennzeichnen im Volltext-Attribut repräsentiert werden. Diese Gliederung machen sich dann Filtermodule zu Nutze.

5.3 Filtermodule


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 Eigenschaf

Erläuternde Beispiele

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


capital: 5. der html-nach-soif-konverter
Attribute im SOIF-Objekt nicht mehr erkennbar ist, werden die Inhalte einzelner Listen zur Veranschaulichung umrandet dargestellt.

- Satzstruktur bleibt erhalten
  \[
  \text{\textless H1\textgreater Dies ist ein Satz\textless/\textgreater} \quad \rightarrow \quad \text{headings\{\ldots\}: Dies ist ein Satz Dies ist ein weiterer Satz}
  \]

- Das "PARENT"-Schlüsselwort
  \[
  \text{\textless BODY\textgreater Dies ist \textless B\textgreater ein Satz\textless/\textgreater BODY} \quad \rightarrow \quad \text{body\{\ldots\}: Dies ist ein Satz keywords\{\ldots\}: ein Satz}
  \]

- \textless NUTAG \& \textless BODY\textgreater
  Die SOIF-Tabelle aus Abbildung 5.1 ordnet sowohl dem \textless NUTAG\textgreater- als auch dem \textless BODY\textgreater-Tag das SOIF-Attribut "body" zu. Daher stehen Textteile vor dem \textless BODY\textgreater-Tag in einem anderen Listenelement als diejenigen danach. Im SOIF-Attribut sind diese Teil schließlich durch einen Zeilenumbruch getrennt.

  Der \textless body\textgreater Wagen\textless/\textgreater body\{\ldots\}: 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 \textless body\textgreater Wagen\textless/\textgreater body\{\ldots\}: Der Wagen

- HTML-Attribute
  \[
  \text{<a href="1.htm">I</a> \quad \rightarrow \quad \text{keywords\{\ldots\}: I url-references\{\ldots\}: 1.html 2.html}}
  \]

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 \texttt{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}"  


    url-references "enc1738 \@{X1}"

5.3.2 Automatische Zusammenfassung

Neben der Möglichkeit, aus den Metadaten\footnote{z.B.: \texttt{<META NAME="description" CONTENT="In diesem Dokument wird ..."}>} 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 einem 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 einem 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 einem Absatz, dessen Reihung im Übergabeparameter angegeben ist.

    body abstract "abs.numx(\@{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.
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.


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. Perlscript 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 $ *
KAPITEL 5. DER HTML-NACH-SOIF-KONVERTER

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.


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 Wortschlägen auszuscheiden sind. Dieser Filter läßt sich für die Suche und für die erweiterte Dokumentenbeschreibung verwenden.

**Ranking der Suchergebnisse:** Je nach zu erreichendem Gewicht sollen die gefundenen Dokumente gereiht werden. Dieses Gewicht hängt von der Formatierung und Häufigkeit des Suchwörtes im jeweiligen 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 auszuscheiden (siehe auch Abschnitt 4.2.1). Statisch kann dies mittels Stoplisten geschehen. Ein Relevanzfilter hingegen zählt die Dokumente, in denen das jeweilige Schlüsselwort auftritt und entscheidet dann, ob dieses auch tatsächlich als solches verwendet wird.

- Ein Wort, das oft auftritt, aber gut über die einzelnen Server verteilt ist, würde aufgenommen; ein geballter 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 ausgeschieden 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 Wahrhafftigkeiten 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 Zugriffsmechanismen\(^1\), die es erlauben, effizient nach Textteilen zu suchen. SQL-Datenbanken sind weit verbreitet und bieten sich somit als Basis für den Suchindex an.

Bevor auf die Details der Index-Datenbank eingegangen wird, werden im folgenden 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]


---
\(^1\) Relativ in Verbindung mit dem LIKE-Vergleichsoperator [FB92]
\(^2\) Zum 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 Inkonstanz der Daten. [Dat86]

<table>
<thead>
<tr>
<th>Dokumentnamen</th>
<th>Wortanzahl</th>
<th>Wort</th>
<th>Gesamthäufigkeit</th>
<th>Häufigkeit im Dokument</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>SQL.html</td>
<td>3</td>
<td>Daten</td>
<td>24</td>
<td>2</td>
</tr>
<tr>
<td>SQL.html</td>
<td>3</td>
<td>Bank</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>index.html</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Dokumenttabelle</th>
<th>Worttabelle</th>
<th>Wort-Dokument-Tabelle</th>
</tr>
</thead>
<tbody>
<tr>
<td>DID</td>
<td>Dname</td>
<td>Wanzahl</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>1</td>
<td>SQL.html</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>index.html</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

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.


Damit auch die in Abschnitt 4.2.4 beschriebene zweistufige Suche möglich wird, muß weitem 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 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 gleich\(^3\) 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 Dokumenttabelle, 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.

<table>
<thead>
<tr>
<th>WID</th>
<th>WORD</th>
<th>ABODY</th>
<th>AKEYWORD</th>
<th>ATITLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>405</td>
<td>jupiter</td>
<td>23</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>406</td>
<td>mars</td>
<td>27</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>407</td>
<td>sonne</td>
<td>29</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

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

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

Dokumenttabelle


<table>
<thead>
<tr>
<th>IID</th>
<th>URL</th>
<th>FILENAME</th>
<th>LTIME</th>
<th>WBODY</th>
<th>WTITLE</th>
<th>WKEYWORD</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>5</td>
<td>planet.html</td>
<td>OBJ1907865</td>
<td>9400023</td>
<td>20</td>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>index.html</td>
<td>OBJ6898324</td>
<td>9400059</td>
<td>124</td>
<td>27</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>stars.html</td>
<td>OBJ8989896</td>
<td>9400099</td>
<td>6755</td>
<td>152</td>
<td>12</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Abbildung 6.4: Dokumenttabelle - DOC-T

Wort-Dokument-Tabelle


<table>
<thead>
<tr>
<th>WID</th>
<th>IID</th>
<th>DBODY</th>
<th>DKEYWORD</th>
<th>DTITLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>405</td>
<td>5</td>
<td>9</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>405</td>
<td>68</td>
<td>6</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>406</td>
<td>5</td>
<td>55</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Abbildung 6.5: Wort-Dokumenttabelle - WD-T

---

4 Sekunden 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.

<table>
<thead>
<tr>
<th>IID</th>
<th>URL</th>
<th>FILENAME</th>
<th>DOKCOUNT</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>17</td>
<td><a href="http://www.iicm.edu">www.iicm.edu</a></td>
<td>SITE089</td>
<td>988</td>
</tr>
<tr>
<td>18</td>
<td><a href="http://www.ml.org">www.ml.org</a></td>
<td>SITE164</td>
<td>2045</td>
</tr>
<tr>
<td>19</td>
<td>styria:9552/TouchTheCity/</td>
<td>SITE091</td>
<td>109</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Abbildung 6.6: Site-Tabelle - SITE-T

Wort-Site-Tabelle


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.

<table>
<thead>
<tr>
<th>IID</th>
<th>WID</th>
<th>SBODY</th>
<th>SKEYWORD</th>
<th>STITLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>5</td>
<td>19</td>
<td>19</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>18</td>
<td>60</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>18</td>
<td>55</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

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

Den Sites (Server, Webbereiche) werden nicht nur die Wörter der enthaltenen Dokumente zugeordnet, sondern auch Worte, die in einer Site-Beschreibung extrahiert werden. Diese wird indiziert, wenn der Systemadministratoren einen zuvor angemeldeten Webbereich 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 Attributzähler verwendet.

<table>
<thead>
<tr>
<th>IID</th>
<th>WID</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>5</td>
<td>4149</td>
</tr>
<tr>
<td>6</td>
<td>34148</td>
</tr>
<tr>
<td>5</td>
<td>138444</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

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. Trifft 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 Reihe 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 überschreitet.

- 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 Dokumenten-Tabelle 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 Dokumenttabelle:

1. Es wird überprüft, ob das zu indizierende Objekt ein gültiges SOIF-Objekt ist. Nur im positiven Fall wird mit dem Einfügen begonnen.


Ermittlung der dazugehörnden Site: Mit Hilfe der URL des einzufügenden Dokumentes wird in der Site-Tabelle nach einer dazugehörnden Site gesucht. Falls eine solche vorhanden ist, wird der Dokumentzähler 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.


---

\(^5\)Solche 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 Tabellen-
    attribut 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 vor-
    kommt. Kommt zum Beispiel das Wort fünfmal im Volltext und drei-
    mal 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 in betreffendem 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 Do-
      kument dieser Site das Wort. Es brauchen nur mehr die jeweiligen Attri-
      but-Zähler des Eintrages entsprechend um eins vergrößert wer-
      den, da diese Tabellenattribute die Dokumente zählen, in denen das
      Wort vorkommt.

    • Andernfalls wird ein neuer Eintrag mit Wort-ID und Site-ID an-
      gelegt und die jeweiligen Attribut-Zähler des Eintrages werden auf
      eins gesetzt, da das einzufüllende Dokument das erste ist, das auf
      dieser Site dieses Wort enthält.

Dokumente im Index sind voneinander unabhängig, das heißt, ein Neueintrag be-
einflußt die anderen Dokumenteinträge nicht. Anders ist dies bei Site-Einträge. Wird
eine Site in den Index aufgenommen, muß überprüft werden, ob nicht bereits eine über-
geordnete 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 umge-
wandelt 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 Erstellldatum eingefügt werden soll (siehe Abschnitt 6.3.1). Folgende Schritte werden hierbei durchlaufen:

Behandlung des Eintrages in der Dokumenttabelle:

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 fundene Wort wird nun folgende Prozedur durchlaufen:

1. Der Eintrag in der Wort-Dokument-Tabelle wird gelöscht.
3. In der Wort-Site-Tabelle wird, falls die Site im Index existiert, nach dem Eintrag des zu behandelnden Wortes gesucht.

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

---

6 Eine 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.


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.


6.5 Suchabfragen


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

<table>
<thead>
<tr>
<th>HD</th>
<th>D.Name</th>
<th>W-Anzahl</th>
<th>WID</th>
<th>Wort</th>
<th>G-Blitufigkeit</th>
<th>HD</th>
<th>WID</th>
<th>D-Blitufigkeit</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>2</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>1</td>
<td>SQL.html</td>
<td>3</td>
<td>1</td>
<td>Daten</td>
<td>24</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>SQL.html</td>
<td>3</td>
<td>1</td>
<td>Daten</td>
<td>24</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>SQL.html</td>
<td>3</td>
<td>2</td>
<td>Bank</td>
<td>7</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>index.html</td>
<td>...</td>
<td>1</td>
<td>Daten</td>
<td>24</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>index.html</td>
<td>...</td>
<td>2</td>
<td>Bank</td>
<td>7</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>index.html</td>
<td>...</td>
<td>2</td>
<td>Bank</td>
<td>7</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

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 Zeile 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 zusammengelagert, wenn die IDs gleich sind. Im Beispiel von Abbildung 6.9 müssen also die zwei Wort- und die zwei Dokument-IDs gleich sein.

<table>
<thead>
<tr>
<th>HD</th>
<th>D.Name</th>
<th>W-Anzahl</th>
<th>WID</th>
<th>Wort</th>
<th>G-Blitufigkeit</th>
<th>HD</th>
<th>WID</th>
<th>D-Blitufigkeit</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>1</td>
<td>SQL.html</td>
<td>3</td>
<td>1</td>
<td>Daten</td>
<td>24</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>SQL.html</td>
<td>3</td>
<td>2</td>
<td>Bank</td>
<td>7</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

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

### 6.5.2 Suche anfrage mit einem Wort

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'
```


Die meisten SQL-Datenbanksysteme erlauben die Angabe von regulären Ausdrücken. Mittels Platzhalter (“%”) 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'
```

Postgres-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 zusätzlich Bedingung in der WHERE-Klausel festgelegt, daß die Dokument-ID nicht in der Menge der Dokument-IDs von Dokumenten, die das "verbotene" Wort enthalten, sein darf. Das folgende Beispiel ("jupiter AND NOT mars") liefert als Ergebnis die leere Menge

\[
\begin{align*}
&\text{select URL from DOC-T, WD-T, WORD-T} \\
&\text{where DOC-T.IID=WD-T.IID and WD-T.WID=WORD-T.WID} \\
&\text{and WORD=’mars’ and DOC-T.IID not in} \\
&(\text{select IID from WD-T, WORD-T} \\
&\text{where WD-T.WID=WORD-T.WID and WORD=’jupiter’})
\end{align*}
\]

### 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.

\[
\begin{align*}
&\text{select URL from DOC-T, WD-T, WORD-T} \\
&\text{where DOC-T.IID=WD-T.IID and WD-T.WID=WORD-T.WID} \\
&\text{and WORD=’mars’ and WD-T.DTITEL>0}
\end{align*}
\]

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.

\[
\begin{align*}
&\text{select URL from DOC-T, WD-T, WORD-T} \\
&\text{where DOC-T.IID=WD-T.IID and WD-T.WID=WORD-T.WID} \\
&\text{and WORD=’mars’ and WD-T.DTITEL>0 and WORD.ATITEL=1}
\end{align*}
\]

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.

\[
\begin{align*}
&\text{select URL from DOC-T, WD-T, WORD-T} \\
&\text{where DOC-T.IID=WD-T.IID and WD-T.WID=WORD-T.WID} \\
&\text{and (WD-T.DBODY/DOC-T.WBODY)>0.1 and URL=’index.html’}
\end{align*}
\]
Konfigurierbarer Keyword-Relevanz-Filter

In der Worttabelle wird unter anderem die Anzahl der Dokumente mitgefahren, 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, wiedergetrennt 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.DTITLE>0 order by WEIGHT desc
```


Komplizierter 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+(DKWORD/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 Resultateinzel zusammengefasst, wobei die Aggregatfunktion (sum) die Einzelwerte summiert.

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+(DKWORD/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 wie vielen 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
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 DeleteObject 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 NewObject 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.


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 "PEROBJ" für die Einzelindizierung und "FULL" für die Vollindizierung eingetragen werden.

Die Server- und Siteberschreibung wird direkt mit dem Broker-Client-Tool (brkcclient) [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.

\texttt{brkclient HOST PORT "#ADMIN insert . siteinfo Suser=NAME
Surl=URL Brief=KURZ Long=LANG #END"}

\section*{6.7.2 Die Suchabfragen}

Die Programmteile des Brokers 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.\footnote{Host und Portnummer des Broker, der abgefragt werden soll.}

\subsection*{Erweiterte Anfragen}

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

\texttt{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:

\begin{itemize}
  \item \texttt{queryid} Durch Angabe dieses Parameters kann das Suchergebnis partitioniert werden.
    Das Gesamtsresultat wird vom Indexer für geschätzte 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 "mensch" und will die ersten zehn Dokumente erhalten.
    \hspace{1cm} \ldots"#USER #index queryid 123456-1-10 #END Daten"
    Nun will er Ergebnis 19 ansehen. Der Anfragestring kann weggelassen werden:
    \hspace{1cm} \ldots"#USER #index queryid 123456-19-19 #END"
  \item \texttt{relkey} Hiermit wird festgelegt, daß die relevanten Keywords eines gefundenen Dokumentes mitübergeben 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.
    \hspace{1cm} \ldots"#USER #index relkey 5-10 #END"
\end{itemize}

\footnote{Auf 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 Ergebnisrückgab**

Bes einer Suchanfrage wird nun in der Funktion ResolveQuery 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 gewünschte\(^{11}\) 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
```

\(^{10}\) Diese Überprüfung ist beim neuen Index überflüssig, da abgelaufene Dokumente in der Datenbank tatsächlich sofort gelöscht werden.

\(^{11}\) Die 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.

**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 Gesammtresultat 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.


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 denen 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.

In nächsten und letzten Kapitel werden nach einer Zusammenfassung dieser Arbeit weiter Erweiterungen des vorgestellten Suchsystems vorgeschlagen und mögliche zukünftige Anwendung erörtert.
Kapitel 7
Zusammenfassung und Ausblick


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 fehler tolerant. 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 Benutzeroberfläche die Möglichkeiten des neuen Indexers weiter ausgeschöpft werden. So wären auf Server- und Sites beschränkte Suchanfragen und die Gewichtung einzelner Suchtermine 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.


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 Schmittstelle 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.

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.


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 (GENTLE\textsuperscript{1}) [DM98] ein solches Suchsystem als dynamische Hintergrundbibliothek verwenden. Hier könnte zusätzlich durch Aufnahme von Diskussionsbeiträgen und Bemerkungen eine dynamische Wissensdatenbank aufgebaut werden. [DGM\textsuperscript{+}98]


\textsuperscript{1}GEneral 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 Document ....... 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

<table>
<thead>
<tr>
<th>Größe</th>
<th>Einheit</th>
<th>Beschreibung</th>
<th>Formel</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q_v$</td>
<td>–</td>
<td>Disjunktives Query</td>
<td>4.2</td>
</tr>
<tr>
<td>$Q_\land$</td>
<td>–</td>
<td>Konjunktives Query</td>
<td>4.2</td>
</tr>
<tr>
<td>$d_{Ax}$</td>
<td>–</td>
<td>Gewicht des Wortes x im Dokument A</td>
<td>4.6 4.5 4.6</td>
</tr>
<tr>
<td>$a_1$</td>
<td>–</td>
<td>Gewicht des Suchwortes x</td>
<td>4.6 4.5 4.6</td>
</tr>
<tr>
<td>$SIM(Q, A)$</td>
<td>–</td>
<td>Ähnlichkeit zwischen Query Q und Dokument A</td>
<td>4.6 4.5 4.6</td>
</tr>
</tbody>
</table>
Literaturverzeichnis


Harvesting and Processing on the Web. Erscheint in European Telematics: 
Advancing the Information Society, (Februar 1998).

[GDN+98] Christian Güttl, Thomas Dietinger, Dietmar Neussl, Bernhard Knögl und 
Klaus Schmaranz. Dynamic Background Libraries as an improved Way 
for Web-Based Learning using HIKS (Hierarchical Interactive Knowledge 

[Gra97] Graphics, Visualisation and Usability Center, GUV's 7th WWW User Sur-
vey. Technical Report, College of Computing, Georgia Institute of Techno-
logy, Atlanta, (Juni 1997).

iso.org/.

for Information Processing and Computer Supported New Media (IICM), 
University of Technology Graz, (Februar 1998).


[Har92] Donna Harman. Ranking Algorithms. Information Retrieval: Data Struc-

[HS95] Darren R. Hardy und Michael F. Schwartz. Customized Information Ex-
traction as a Basis for Resource Discovery. Technical Report, Department 

[HSW96] Darren R. Hardy, Michael F. Schwartz und Duane P. Wessels. Harvest 


[JJ97] M. Jensfeld and M. Jarke. Suchhilfe für das World Wide Web - Funktions-

[Koc96] Traugott Koch. Suchmaschinen im Internet, Weiter auf dem Weg zur 


