Integration von Inhalten aus Content-Managementsystemen in lokalen Suchmaschinen

Diplomarbeit

Universität Rostock, Fachbereich Informatik

vorgelegt von Götz Waschk
geboren am 17.11.1975 in Neustrelitz

Gutachter: Prof. Dr. Andreas Heuer
2. Gutachter: Prof. Dr. Peter Forbrig
Betreuer: Dipl.-Inf. Gunnar Weber
Dipl.-Inf. Mathias Bietz

abgegeben am 31. März 2002

Zusammenfassung

Content-Managementsysteme haben eine wachsende Bedeutung bei der Verwaltung digitaler Dokumente. Sie ermöglichen die zentrale Speicherung und den Zugriff auf mit Metadaten angereicherte Inhalte unterschiedlichsten Typs, wie semistrukturierte Texte, Multimedia- und Web-Daten. Die bestehenden Suchmaschinen haben den Nachteil, dass sie diese oft hochwertigen Daten nicht durchsuchen können. Ein Vorteil eines Content-Managementsystems ist die integrierte Retrievalkomponente, die umfangreiche Funktionen zur Volltextsuche und zur Anfrage an Metadaten bereitstellt. In dieser Diplomarbeit wurde eine Kopplungsarchitektur entwickelt, die bestehende Suchmaschinen und die von Content-Managementsystemen zur Verfügung gestellte Retrievalfunktionalität zu einem verteilten Retrievalsystem vereinigt.

Abstract

Content management systems become increasingly important for the management of digital documents. They allow the central storage and access to content of different types, enriched with meta data like semi-structured text, multimedia and web data. The disadvantage of existing search engines is that they cannot search through this high grade data. One advantage of a content management system is the integrated retrieval component that provides extensive functions for full text search and the query for meta data. This paper describes the development of an integration architecture, that combines existing search engines and the retrieval functionality provided by content management systems into a distributed retrieval system.

CR-Klassifikation

H.2.2 [Database Management]: Languages — Query Language

H.2.4 [Database Management]: Systems — Distributed Systems

H.2.5 [Database Management]: Heterogenous Databases

H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval

Key Words

Content Managementsysteme, Suchmaschinen,
Text Retrieval, verteilte Suche

Inhaltsverzeichnis

Kapitel 1

Einführung

Immer mehr Daten werden in Content-Managementsystemen gespeichert, da diese unschätzbare Vorteile im Umgang mit digitalen Dokumenten bieten. Zwei dieser Vorteile sind die zentrale Verwaltung aller in einem Unternehmen anfallenden Daten und die Verwendung von Metadaten. Natürlich soll dieser Datenbestand durchsucht werden können. Die üblicherweise im Einsatz befindlichen Suchmaschinen können aber nur lokale Dateisysteme bzw. statische Web-Seiten durchsuchen.

Um auch in Content-Managementsystemen gespeicherte Informationen für die lokale Suchmaschine zugänglich zu machen, hat man zwei Möglichkeiten:

1. Einsammeln der Daten über die Web-Repräsentation des Content-Managementsystems (CMS).

2. Ausnutzung der vom Content-Managementsystem zur Verfügung gestellten Retrieval-Funktionalität.

Der erste Ansatz entspricht der Arbeitsweise von Web-Suchmaschinen, falls die Web-Repräsentation aus statischen Seiten besteht. Wenn Webseiten dynamisch als Reaktion auf Nutzereingaben, also als Antwort auf ein ausgefülltes Formular erzeugt werden, versagen etablierte Suchmaschinen. Für diesen Anwendungsfall wurde in [Was01] ein Verfahren implementiert, dass aus Metadaten in Formularen dynamisch Wrapper generiert, die Anfragen an die Web-Server schicken und die dynamisch generierten Seiten auswerten.

Der erste Ansatz hat den Nachteil, dass Daten redundant gespeichert werden. Dies ist beim zweiten Ansatz nicht der Fall, zusätzlich besteht der Vorteil, dass dort die Suche immer auf dem aktuellen Datenbestand stattfindet. Im Rahmen dieser Arbeit wurde der zweite Ansatz verfolgt, um die Suchfunktionalität von lokalen Suchmaschinen und CMSen zu integrieren.

1.1 Gliederung

Diese Arbeit ist folgendermaßen gegliedert: Kapitel 2 stellt die Grundlagen des Information Retrieval (IR) und des Textretrieval bereit, die sowohl in den späteren Abschnitten über Suchmaschinen als auch in denen über Content Management benötigt werden.

Kapitel 3 enthält eine Einführung in die Welt der Suchmaschinen. Es werden die im Rahmen dieser Arbeit benötigten Grundlagen definiert und als Beispiel einige konkrete Systeme vorgestellt.

Kapitel 4 definiert den Begriff des Content-Managementsystem (CMS) und beurteilt mehrere CMS unter dem Aspekt der Retrieval-Funktionalität und der Möglichkeiten, die sie im Rahmen einer Implementierung des hier vorgestellten Konzepts bieten.

In Kapitel 5 werden mehrere mögliche Architekturmodelle vorgestellt, die eine Integration von CMS und Suchmaschinen ermöglichen, Kapitel 6 beschreibt die schließlich implementierte Variante. Die Details der Implementierung findet man in Kapitel 7. Zum Schluss erfolgt in Kapitel 8 die Zusammenfassung der Ergebnisse dieser Arbeit und ein Ausblick auf mögliche Erweiterungen und Verbesserungen der Architektur.

Kapitel 2

Information Retrieval

Dieses Kapitel beschäftigt sich mit Grundlagen des Information Retrieval (IR), da diese in den Kapiteln über Suchmaschinen und Content-Managementsysteme benötigt werden. Zunächst werden Grundbegriffe definiert und anschließend die theoretischen Hintergründe einiger Retrievalmodelle betrachtet.

2.1 Definition

In [SM87] wird IR folgendermaßen definiert:

> Gegenstand des Information Retrieval ist die Repräsentation, Speicherung und Organisation von Informationen und der Zugriff auf Informationen.

Eine andere Definition in [Rij79] definiert IR mit Hilfe einer Gegenüberstellung von Data Retrieval (DR) und IR, die in Tabelle 2.1 wiedergegeben wurde.

Ein IR-System soll dem Nutzer einen einfachen Zugriff auf die Informationen bieten, an denen er interessiert ist. Der Nutzer formuliert eine Anfrage an das IR-System in einer wohldefinierten Anfragesprache, und das IR-System liefert die zur Anfrage passenden Informationen zurück.

Ein wichtiges Merkmal des Information Retrievals, dass auch in Tabelle 2.1 dargestellt wurde, ist die Notwendigkeit unscharfer Anfragen. Der Nutzer weiß in der Regel nicht exakt, welche Begriffe in einem Dokument vorkommen, darum muss ein IR-System die Ähnlichkeit zwischen der Anfrage und Dokumenten ausnutzen. Dies wird durch Techniken wie Thesauri implementiert.

Merkmal	Data Retrieval	Information Retrieval
Trefferermittlung	exakt	teilweise Treffer, beste Treffer
Folgerung	Deduktion	Induktion
Modell	deterministisch	probabilistisch
Klassifikation	monothetisch	polythetisch
Anfragesprache	künstlich	natürlich
Anfragespezifikation	vollständig	unvollständig
geforderte Elemente	zutreffende	relevante
Reaktion auf Fehler	empfindlich	unempfindlich

Tabelle 2.1: Unterschiede zwischen Data und Information Retrieval

2.2 Retrievalmodelle

Definiert wird ein IR-Modell in [BYRN99] als Quadrupel $\langle D, Q, F, R, (q_i, r_j)\rangle$, darin sind:

- D eine Menge logischer Sichten auf Dokumente bzw. Repräsentationen der Dokumente in der Sammlung,
- Q eine Menge logischer Sichten auf die Nutzerbedürfnisse, die Anfragen,
- F ein Rahmen zur Modellierung von Dokumentrepräsentationen, Anfragen sowie deren Beziehungen und
- $R(q_i, r_j)$ eine Ranking-Funktion, die einer Anfrage q_i aus Q und einer Dokumentrepräsentation r_j aus D eine reelle Zahl zuordnet.

Die einzelnen Elemente eines Retrievalmodells müssen noch näher erläutert werden. Anschließend werden einige in der Praxis wichtige Modelle vorgestellt. Die verwendeten Formeln stammen aus [Fuh98]. Es ist noch anzumerken, dass die in den folgenden Abschnitten vorgestellten Retrievalmodelle im Widerspruch zur IR-Definition in Tabelle 2.1 zur Klasse der nicht-probabilistischen Retrievalmodelle gehören.

Dokument Ein Dokument ist eine Sammlung von Wörtern. Repräsentiert werden Dokumente durch Indexterme. Das sind „wichtige“ Wörter, die semantische Bedeutung besitzen, wie z.B. Substantive. Sie werden vereinfacht durch die Entfernung von Stoppwörtern aus der Menge möglicher Indexterme und durch Stammwortreduktion. Indexterme sollen das Dokument von anderen Dokumenten unterscheiden.

Für die folgenden Modelle wird angenommen, dass das Auftreten eines Terms t_1 in einem Dokument unabhängig ist vom Auftreten eines Terms t_2 im selben Dokument. In diesen Fällen kann ein Dokument als Menge von Indextermen modelliert werden. Die Reihenfolge der Indexterme wird beim Retrieval nicht berücksichtigt.

Die Menge aller Indexterme einer Dokumentenmenge wird durch $\{k_1, \ldots, k_t\}$ angegeben, t ist die Anzahl der Dokumente. Ein Dokument d_j wird normalerweise durch einen Vektor repräsentiert:
$d_j = \langle w_{1,j}, \ldots, w_{t,j}\rangle$ mit $w_{t,j}$ als Gewicht des Terms k_i im Dokument d_j.

Anfrage In den einfachen Retrievalmodellen besteht eine Anfrage analog zu den Dokumentrepräsentationen aus einer Menge von gewichteten Termen, und sie wird als Vektor der Termgewichte dargestellt:

$$q_k^Q = \vec{q_k}$$

Welche Gewichte möglich sind, hängt vom jeweiligen Retrievalmodell ab.

2.2.1 Boolesches Retrievalmodell

Im booleschen Modell besteht eine Anfrage aus Termen, die durch die logischen Operatoren AND, OR und NOT verknüpft sind. Eine Anfrage wird als Funktion interpretiert, die eine Mengenzugehörigkeit ermittelt.

Die Anfrage $Q = k_a$ AND (k_b OR NOT k_c) liefert alle Dokumente, die das Wort k_a enthalten und entweder k_b enthalten oder k_c nicht enthalten.

Es gilt für jedes Dokument, dass es entweder die Anfrage erfüllt, dann ist es relevant und es wird 1 zurückgeliefert, oder es erfüllt die Anfrage nicht, ist also

irrelevant, dann wird 0 zurückgeliefert. Die Dokumentbeschreibungen D^D lassen sich als ungewichtete Indexierungen darstellen, also:

$$d_m^D = \vec{d}_m \quad \text{mit} \quad d_{m_i} \in \{0, 1\} \tag{2.1}$$

Im Vektor $\vec{d}_m$ wird jedes im Dokument enthaltene Wort durch eine 1 und jedes nicht enthaltene Wort durch eine 0 repräsentiert.

Die Menge der Anfragebeschreibungen Q^D besteht aus booleschen Ausdrücken, die nach diesen Regeln gebildet werden:

1. $t_i \in T \Rightarrow t_i \in Q^D$
2. $q_1, q_2 \in Q^D \Rightarrow q_1 \wedge q_2 \in Q^D$
3. $q_1, q_2 \in Q^D \Rightarrow q_1 \vee q_2 \in Q^D$
4. $q \in Q^D \Rightarrow \neg q \in Q^D$

Die Anfragen werden also aus Termen mit den logischen Operationen UND, ODER und NICHT rekursiv zusammengesetzt.

Die Retrievalfunktion R kann man rekursiv folgendermaßen definieren:

1. $t_i \in T \Rightarrow R(t_i, \vec{d}_m) = d_{m_i}$
2. $R(q_1 \wedge q_2, \vec{d}_m) = \min(R(q_1, \vec{d}_m), R(q_2, \vec{d}_m))$
3. $R(q_1 \vee q_2, \vec{d}_m) = \max(R(q_1, \vec{d}_m), R(q_2, \vec{d}_m))$
4. $R(\neg q, \vec{d}_m) = 1 - R(q, \vec{d}_m)$

Die Retrievalfunktion berechnet sich also aus dem Vorhanden-Sein des Einzelterms in der Dokumentrepräsentation (Formel 1) und den Berechnungsvorschriften, die den logischen Operatoren in der Anfragerepräsentation zugeordnet werden (Formeln 2-4).

Bewertung Das boolesche Retrievalmodell zählt zu den verbreitetsten Verfahren, ist aber mit einigen schwerwiegenden Problemen behaftet. Vorteilhaft ist zunächst die mächtige Anfragesprache, mittels Mengenalgebra lassen sich beliebige Teilmengen der Dokumentenmenge auswählen.

Ein großer Nachteil ist die nicht vorhandene Abstufung bei der Bewertung der Suchtreffer, ein Dokument ist entweder in der Ergebnismenge oder es ist nicht in der Ergebnismenge. Dadurch ist die Größe der Antwortmenge schwer zu kontrollieren und die Trennung in relevante und nicht relevante Dokumente ist zu streng. Außerdem ist es schwierig, Suchtreffer aus einem booleschen Retrievalsystem beim verteilten Retrieval mit anderen Suchtreffern zu mischen, wie in dieser Arbeit noch dargestellt werden wird.

In theoretischen Untersuchungen wurden Varianten des booleschen Retrievals entwickelt, um eine Rangordnung der Antwortdokumente zu erreichen, die beispielsweise auf die Fuzzy-Logik ausweichen. Dabei besteht die Dokumentrepräsentation $\vec{d}_m$ nicht aus binären Gewichten, sondern es sind beliebige Werte zwischen 0 und 1 möglich, also $d_{m_i} \in [0, 1]$. Verwendet man die oben definierte Retrievalfunktion, dann erhält man Rankingwerte zwischen 0 und 1. Näheres dazu findet man in [Fuh98].

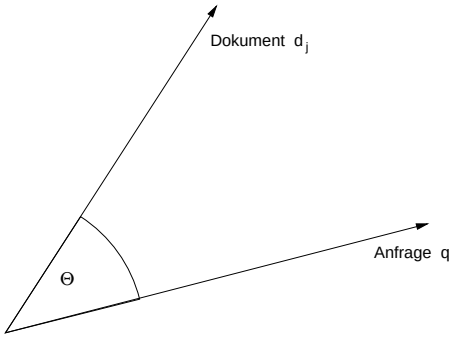

Abbildung 2.1: Veranschaulichung des Abstandsmaßes beim VRM

2.2.2 Vektorraummodell

Im Vektorraummodell (VRM) werden sowohl Anfragen als auch Dokumente durch gewichtete Vektoren repräsentiert. Die Relevanz eines Dokuments lässt sich als Ähnlichkeitsmaß zwischen Anfragevektor und Dokumentvektor berechnen, veranschaulicht in Abbildung 2.1. Die formale Darstellung des Dokumentvektors ist:

$$d_m^D = \vec{d}_m \text{ mit } d_{m_i} \in \mathbb{R} \text{ für } i = 1, \ldots, n$$

Der Anfragevektor ist dann:

$$q_k^Q = \vec{q}_k \text{ mit } q_{k_i} \in \mathbb{R} \text{ für } i = 1, \ldots, n$$

Die Ranking-Funktion gibt den Abstand zwischen Anfrage und Dokument an, ein mögliches Maß zur Berechnung ist das Skalarprodukt von Anfrage- und Dokumentenvektor:

$$R(\vec{q}_k, \vec{d}_m) = \vec{q}_k \cdot \vec{d}_m$$

Die Termgewichte d_{m_i} sollen die Bedeutung der Terme t_i für das Retrieval des Dokuments zum Ausdruck bringen. Die Bedeutung eines Terms in einem Dokument hängt von zwei Faktoren ab:

1. Bedeutung des Terms zur Identifikation des Dokumenteninhaltes (Termfrequenz)
2. Bedeutung zur Unterscheidung des Dokumentes von anderen Dokumenten (Dokumentenfrequenz)

Um daraus eine Gewichtung für die Frage- und Dokumentvektoren berechnen zu können, werden heuristische Formeln angewendet, es sind aber auch andere Berechnungsvorschriften möglich. Zunächst werden die genannten Bewertungsfaktoren formal definiert. In den folgenden Formeln werden diese Symbole verwendet:

d_m^T	Menge der in d_m vorkommenden Terme
tf_{mi}	Termfrequenz von t_i in d_m.
$\max tf_m$	maximale Termfrequenz tf_{mi} aller Terme $t_i \in d_m^T$.
n_i	Anzahl der Dokumente, in denen t_i vorkommt.
$\lvert\underline{D}\rvert$	Anzahl der Dokumente in der Kollektion.

Die Gewichtung wird berechnet aus der inversen Dokumentenfrequenz idf_i und der normalisierten Termfrequenz ntf_i. Die inverse Dokumentenfrequenz idf_i

lässt sich folgendermaßen berechnen:

$$idf_i = \log \frac{|D|}{n_i}$$

Dieser Wert ist um so höher, je seltener ein Begriff in den Dokumenten auftaucht.

Die normalisierte Termfrequenz ntf_i berechnet sich aus der Termfrequenz eines Terms und der maximalen Termfrequenz aller Terme in dem betreffenden Dokument:

$$ntf_i = 0.5(1 + \frac{tf_{mi}}{\max tf_m})$$

Das unnormierte Indexierungsgewicht ist das Produkt aus normalisierter Termfrequenz und inverser Dokumentenfrequenz:

$$\alpha_{mi} = ntf_i \cdot idf_i$$

Dann wird der Dokumentenvektor normiert, so dass $|\vec{d_m}| = 1$ gilt. Dazu wird der Betrag w_m berechnet:

$$w_m = \sqrt{\sum_{t_i \in d_m^T} \alpha_{mi}^2}$$

Das normierte Indexierungsgewicht ist dann:

$$d_{mi} = \frac{\alpha_{mi}}{w_m}$$

Bewertung Ein großer Vorteil beim VRM ist die Möglichkeit des Relevance Feedbacks, also der iterativen Bewertung der Ergebnismenge und Anpassung der Anfrage durch den Nutzer. Dabei werden die Fragetermgewichte verändert, was zu einem neu ausgerichteten Anfragevektor führt. Dadurch lässt sich eine besonders gute Trennung von relevanten und nicht relevanten Dokumenten erreichen. Die theoretischen Hintergründe dazu finden sich in [Fuh98].

Weiterhin ist vorteilhaft, dass das VRM eine einfache Frageformulierung und eine gute Retrievalqualität bietet.

Ein theoretischer Nachteil ist die Verwendung der oben genannten Heuristiken zur Gewichteberechnung, es lässt sich nicht vorhersagen, wie gut diese Formeln bei anderen Dokumentkollektionen funktionieren.

Erweiterte Dokumentrepräsentationen lassen sich nur schlecht integrieren, da sich die Gewichtungsformeln nur schwer erweitern lassen, somit ist beispielsweise eine höhere Gewichtung von Wörtern im Dokumententitel problematisch.

2.3 Anfrageverfahren

Es existieren verschiedene Typen von Anfragen in Textretrievalsystemen, die ebenfalls in [BYRN99] vorgestellt werden. Um ein besseres Verständnis des in Kapitel 4 durchgeführten Vergleichs von Retrievalkomponenten verschiedener CMS zu ermöglichen, werden nun die wichtigsten Typen vorgestellt.

Es wird in diesem Abschnitt nicht der theoretische Hintergrund betrachtet, er behandelt lediglich die Nutzersicht auf ein Textretrievalsystem.

2.3.1 Schlüsselwort-basiert

Schlüsselwort-basierte Anfragen bestehen aus einem oder mehreren Wörtern bzw. Wortbestandteilen, die auf unterschiedliche Art miteinander verknüpft werden können.

Suche nach einzelnen Wörtern

Die Suchanfrage besteht aus einer Liste von Wörtern. Das Anfrageergebnis besteht aus den Dokumenten, in denen mindestens eines der Wörter enthalten ist. Es findet ein Ranking nach der Ähnlichkeit zur Anfrage statt, d.h. um so mehr der Suchwörter in einem Dokument vorhanden sind, um so höher sein Ranking-Wert.

Kontext-Anfragen

Bei Kontext-Anfragen werden die Schlüsselwörter der Anfrage nicht isoliert voneinander bearbeitet, sondern es wird die Umgebung der Wörter im Text berücksichtigt. Da die Position eines Wortes im Text entscheidend ist, müssen aufwendigere Index-Typen als bei der Einzelwortsuche verwendet werden. Bei den Kontext-Anfragen kann man folgende unterscheiden:

Phrasensuche Die Anfrage besteht aus einer Aneinanderreihung von Schlüsselwörtern, die in dieser Reihenfolge im Text vorhanden sein müssen. Die Bearbeitung dieser Anfrage ignoriert Unterschiede zwischen Anfrage und Dokument bezüglich Füllwörtern und Leerzeichen und wird in eine Sequenz aus Einzelwortabfragen aufgelöst.

Wortabstandssuche Bei einer Wortabstandssuche (englisch *Proximity Search*) wird in der Anfrage der maximal erlaubte Abstand im Text zwischen den Schlüsselwörtern formuliert. Je nach Implementierung kann dieser Abstand als Anzahl von Zeichen oder Wörtern angegeben werden. Es ist außerdem möglich, dass die Wörter in einer anderen Reihenfolge auftreten, als in der Anfrage angegeben.

Boolesche Anfragen

Es wird ein boolescher Term als Anfrage verwendet. Dieser besteht aus Wörtern, die mittels boolescher Operatoren verknüpft werden. Die Operatoren können auch verschachtelt werden. Die üblicherweise verwendeten Operatoren sind folgende:

AND t_1 AND t_2 ist wahr, wenn sowohl t_1 als auch t_2 wahr sind.

OR t_1 OR t_2 ist wahr, wenn entweder t_1 oder t_2 wahr sind.

BUT t_1 BUT t_2 ist wahr, wenn t_1 wahr und t_2 nicht wahr ist.

Der Operator BUT entspricht einer Operation AND NOT und dient als Ersatz für den reinen NOT-Operator. Er soll die großen Datenmengen verhindern, die ein Negierungsoperator als Ergebnismenge zurückliefern kann und schränkt darum das Universum des Operators auf die Ergebnismenge von t_1 ein.

2.3.2 Pattern Matching

In diese Kategorie fallen die Verfahren, die Muster statt Zeichenketten als Anfrageparameter verwenden. Diese Suchmuster geben einen Algorithmus an, der genutzt wird, um die passenden Dokumente bzw. die passende Zeichenkette zu ermitteln.

Eine übliche Form des Musters sind die Wildcard-Platzhalter, ähnlich denen in der UNIX-Shell, also ein '?' für ein beliebiges Zeichen und ein '*' für eine beliebige, auch leere Zeichenkette.

Andere Verfahren arbeiten mit komplizierteren Mustern, wie regulären Ausdrücken oder Bäumen, wobei letztere in Anfragesprachen für strukturierte Dokumente zur Anwendung kommen.

2.3.3 Strukturierte Anfragen

Die bisher vorgestellten Verfahren haben die Texte als flach strukturiert aufgefasst, es spielte keine Rolle, ob ein Schlüsselwort in einem Absatz oder einer Überschrift enthalten war. Die folgenden Verfahren nutzen die in Dokumenttypen wie XML oder HTML enthalten Strukturinformationen aus.

Man kann die Verfahren für strukturierte Anfragen danach unterscheiden, welche Art der Dokumentstruktur sie unterstützen:

- **feste Struktur** Die Dokumente bestehen aus einer festgelegten Menge oder Liste von Feldern, wie dies z.B. bei Emails mit den einzelnen Header-Feldern anzutreffen ist. Dadurch wird ein Datenmodell erreicht, das an das relationale Datenbankmodell erinnert.

- **Hypertext** Die Dokumente bestehen aus Knoten, die Text enthalten und Hyperlinks, die Kanten dazwischen bilden, was vereinigt einen gerichteten Graphen ergibt. Suchverfahren für Hypertext kommen in Web-Suchmaschinen zum Einsatz.

- **hierarchische Struktur** Die Texte werden rekursiv in einzelne Bereiche zerlegt. Die meisten Verfahren erwarten, dass die Bereiche durch Tags begrenzt werden. Die Beispiele orientieren sich in der Syntax an der Sprache Pat, definiert in [PAT]. Diese erlaubt die Definition von benannten Regionen, die z.B. durch XML-artige Tags begrenzt werden. Durch Anfragen an hierarchisch strukturierte Dokumente können Bereiche nach folgenden Kriterien ausgewählt werden:

  – (nicht) enthalten sein in angegebenen Bereichen

    ```
    region Author not within region Quote
    ```

    Die Anfrage liefert die Regionen „Author“, die nicht in einer Region „Quote“ enthalten sind.

  – (nicht) enthalten angegebener Bereiche

    ```
    region Quote including region Author
    ```

    Die Anfrage liefert die Regionen „Quote“, die eine Region „Author“ enthalten.

  – folgen auf angegebene Bereiche (oder gefolgt werden)

    ```
    region Title fby region Author
    ```

    Die Anfrage liefert die Region „Title“, die von der Region „Author“ gefolgt wird.

  – Bestimmter Abstand zu angegebenen Bereichen

    ```
    region Title near.30 region Author
    ```

    Die Anfrage liefert die Region „Title“ mit einem Abstand von 30 Zeichen zur Region „Author“.

2.4 Ranking

Unter dem Begriff Ranking versteht man laut [EF00] ein Sortieren von Suchtreffern nach ihrer Qualität. Die besten Ergebnisse werden durch einen *Ranking-Faktor* bestimmt. Er entspricht einem Relevanz-Kriterium, sofern es vom Nutzer angegeben wurde.

Die Sortierung der Suchtreffer kann nach einem Grad der Übereinstimmung mit der Suchanfrage erfolgen, z.B. bei Volltextsuchverfahren, die dies ermöglichen, oder nach einem Attribut des Dokuments, beispielsweise dem Erstellungsdatum.

Ob und in welcher Form Ranking in einem Textretrievalsystem vorhanden ist, hängt stark mit dem zugrundeliegenden Retrieval-Modell zusammen. So ist ein Dokument beim Booleschen Retrieval entweder relevant oder nicht relevant, ohne weitere Abstufungen. Die Ergebnismenge einer Anfrage besteht dann aus einer ungeordneten Menge aller relevanten Dokumente. Eine nachträgliche Sortierung nach einem in Bezug auf das Retrievalmodell externen Kriterium, wie das o.g. Erstellungsdatum der Dokumente, ist natürlich trotzdem möglich.

2.5 Verteiltes Retrieval

Da vor allem im Web die Informationsmenge ständig anwächst, nach einigen Abschätzungen mit exponentieller Rate, stoßen traditionelle Retrievalsysteme leicht an ihre Kapazitätsgrenzen. Die Kosten für die Suche und die Indizierung steigen mit der Größe der Dokumentensammlung, erschwerend kommt die Dynamik der Inhalte hinzu.

Verteiltes Retrieval stellt einen Ausweg aus dieser Situation dar, mehrere Retrievalsysteme arbeiten parallel an einer Suchanfrage. In [BYRN99] werden die minimalen Anforderungen an ein Protokoll zur verteilten Suche festgelegt. Folgende Funktionen müssen vorhanden sein:

- Ermitteln von Informationen über einen Such-Server, z.B. eine Liste der Datenquellen, die zum Suchen zur Verfügung stehen und eventuell Statistiken über die Datenquellen.
- Übertragen einer Suchanfrage für eine oder mehrere Datenquellen mit einer wohldefinierten Anfragesprache.
- Empfangen der Suchergebnisse in einem wohldefinierten Format.
- Zugriff auf die Objekte aus den Suchergebnissen.

2.5.1 Anfragebearbeitung

Die Anfragebearbeitung in einem verteilten Retrievalsystem hat den folgenden Ablauf:

1. Auswahl zu durchsuchender Datenbestände
2. Verteilung der Anfrage auf ausgewählte Datenbestände
3. parallele Auswertung der Anfrage durch die einzelnen Retrievalsysteme
4. Kombination der Teilergebnisse zu einem Gesamtergebnis

Ein Problem beim verteilten Retrieval ist die Erzeugung des Gesamtergebnisses durch Mischen der Ergebnisse der einzelnen Teil-Retrievalsysteme. Es wird davon ausgegangen, dass das Gesamtergebnis eine gerankte Liste sein soll.

Der einfachste Ansatz geht von gerankten Ergebnissen der Teilsysteme aus, die nach dem Round-Robin-Prinzip zusammengemischt werden. Diese Methode nimmt keine Rücksicht auf die Gesamtrelevanz eines Dokumentes, und ist darum nicht empfehlenswert.

Bessere Ergebnisse lassen sich erzielen, wenn ein einheitliches Verfahren zur Berechnung der Rankingwerte unter Verwendung globaler Termstatistiken eingesetzt wird. Das ist natürlich nur dann möglich, wenn die Teilsysteme diese Statistiken exportieren.

Wenn die verteilten Dokumentkollektionen semantisch partitioniert sind oder von unabhängigen Stellen administriert werden, müssen die Ergebnisse neu gerankt werden. Die Neubewertung wird über eine Gewichtung der Dokumente basierend auf dem Ähnlichkeitswert der Kollektion vorgenommen. Dieser muss bei der Auswahl der zu durchsuchenden Datenbestände berechnet werden.

Kapitel 3

Suchmaschinen

Dieses Kapitel beschäftigt sich mit dem Aufbau und der Funktionsweise von Suchmaschinen, insbesondere mit den Unterschieden zu generischen Textretrievalsystemen. Zunächst werden die Grundlagen definiert und anschließend konkrete Suchmaschinen-Implementierungen vorgestellt.

3.1 Definition

In diesem Abschnitt werden die Begriffe Suchmaschine und Textretrieval-System über die der gängigen Literatur entnommenen Definitionen erläutert.

3.1.1 Suchmaschinen und Textretrieval

Im Rahmen dieser Arbeit soll unter einer Suchmaschine eine spezielle Art eines Textretrieval-Werkzeugs verstanden werden. Damit wird eine Abgrenzung zu Suchsystemen für andere Datentypen, wie z.B. Multimedia erreicht.

Um die unterschiedlichen Arten von Textretrieval-Systemen zu verstehen, wird die Klassifizierung aus [Nit00] verwendet:

- Indexmaschine
- Suchmaschine
- Volltextdatenbank

Indexmaschinen Indexmaschinen sind die Grundbestandteile eines Textretrieval-Systems. Sie beschleunigen die Suche in Textdaten dadurch, dass sie im Vorfeld der Suchanfrage einen Index über die Inhalte von Texten anlegen. Dafür existieren verschiedene Verfahren, beispielsweise *invertierte Listen*. Eine invertierte Liste besteht aus einer alphabetisch sortierten Liste aller Indexterme, denen jeweils die Dokument-Identifikatoren der Dokumente, in denen der Term enthalten ist, zugeordnet werden. Einen Überblick über die in Indexmaschinen verwendeten Techniken bietet [BYRN99].

Suchmaschinen Eine Suchmaschine stellt Funktionen zur Verwaltung von Dokumenten, eine Nutzerschnittstelle zum Indizieren von Dokumenten und zur Anfragebearbeitung, sowie die Möglichkeit der Koordination von mehreren Indizes und Anfragen bereit.

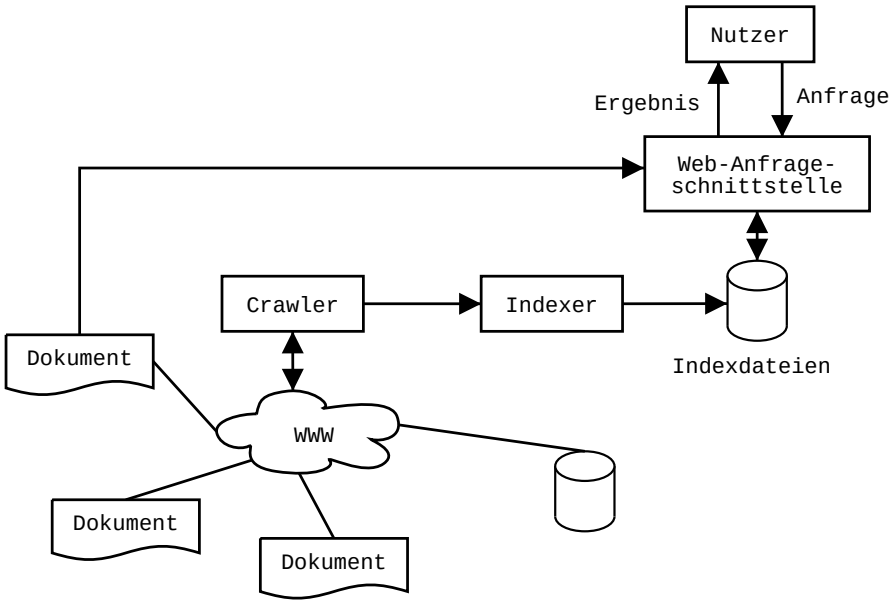

Abbildung 3.1: Funktionsweise einer Web-Suchmaschine

Der Umfang der Funktionen kann stark variieren, ebenso der Aufbau der Nutzerschnittstelle. Die Nutzeroberfläche reicht von einfachen Textkommandos bis zu grafischen Oberflächen. Bei Web-Suchmaschinen sind Web-Formulare üblich.

Häufig bieten Suchmaschinen eine Programmierschnittstelle, mit der Suchanfragen aus Anwendungsprogrammen heraus gestellt werden können.

Volltextdatenbanken Volltextdatenbanken vereinigen die Funktionen eines Datenbank- Management-Systems (DBMS) zur Datenverwaltung mit den Funktionen eines Textretrieval-Systems. Sie unterstützen neben Text noch andere Datentypen, die z.B. Metadaten als Attribute von Texten speichern können.

3.2 Web-Suchmaschinen

Diese verbreitete Kategorie der Suchmaschinen ist auf das Textretrieval im World Wide Web ausgerichtet. Diese Suchdomäne erfordert die Anpassung einiger Annahmen, die für generische Textretrievalsysteme getroffen wurden.

Im Unterschied zu anderen Retrievalsystemen werden die Textdokumente von der Suchmaschine nicht gespeichert, sondern liegen auf den Web-Servern. Darum enthalten Web-Suchmaschinen als Komponente einen sogenannten Crawler[1], der die Dokumente zum Zweck der Indizierung aus dem Web einsammelt. Die Suchmaschine speichert in ihrem Index Hyperlinks auf die ursprünglichen Dokumente in Form von URLs. Die Funktionsweise einer Web-Suchmaschine ist in Abbildung 3.1 dargestellt.

Der Nachteil bei dieser Methode ist die fehlende Kontrolle der Suchmaschine über die Dokumente, Änderungen auf einem durchsuchten Webserver werden erst später nachvollzogen. In der Praxis führt dies häufig zu dem Problem, dass das Dokument unter der im Suchergebnis zurückgelieferten URL nicht mit dem

[1] näheres zur Funktionsweise eines Crawlers steht in [Was01]

Indexeintrag übereinstimmt und somit ein ungültiger Treffer ist, oder die URL sogar ungültig geworden ist.

3.2.1 Metadaten

In den Dokumenttypen, die von Web-Suchmaschinen durchsucht werden können (also hauptsächlich HTML), besteht die Möglichkeit der Einbettung von Meta-Tags, die eine Definition von Metadaten ermöglichen sollen. Für das Format der Meta-Tags existieren verschiedene Formate, wie z.B. Dublin Core[2] oder Learning Object Metadata (LOM). Auch in [Was01] wurde ein spezielles Metadatenformat implementiert.

Die Verwendung von Meta-Tags ist nicht zwingend, in der Praxis bedeutet das, dass die Web-Seiten außerhalb des wissenschaftlichen Bereichs kaum verwertbare eingebettete Metadaten beinhalten.

Außerdem besteht das Problem des Missbrauchs, wie in [SB00] dargestellt. Darum verwenden die besseren Web-Suchmaschinen Heuristiken, die Meta-Tags nach bestimmten Kriterien verwerfen[3].

3.2.2 Meta-Suchmaschinen

Eine Meta-Suchmaschine ist ein verteiltes Retrievalsystem für das Web. Sie leitet die Suchanfragen parallel an andere Suchmaschinen weiter und mischt die Suchergebnisse. In [SBS98] werden folgende Kriterien von einer Meta-Suchmaschine verlangt:

1. Parallele Suche (keine „all-in-one“-Formulare)
2. Mischen der Ergebnisse
3. Eliminierung von Dubletten
4. mindestens AND- und OR-Operatoren
5. Übernahme einer Kurzbeschreibung
6. Verstecken der Spezifika der durchsuchten Suchmaschinen
7. Möglichkeit der vollständigen Suche

Zu 1. ist noch anzumerken, dass bei „all-in-one“-Formularen unter einer einheitlichen Nutzerschnittstelle eine sequenzielle Abfrage mehrerer Suchmaschinen erfolgt.

Diesen Kriterien entsprechend wurde die Meta-Suchmaschine MetaGer[4] implementiert. Den Erfahrungen der Betreiber dieses Suchdienstes zufolge, (ebenfalls in [SBS98]) kann eine Meta-Suchmaschine, die o.g. Eigenschaften besitzt, die Qualität der Suche im Internet wesentlich verbessern.

3.3 Ranking in Suchmaschinen

Da Suchmaschinen eine sehr große Anzahl relevanter Treffer zurückliefern können, hat das Ranking eine hohe Bedeutung. Der Nutzer erwartet, dass die Ergebnisliste absteigend nach der Trefferqualität sortiert ist. Um dieses Ziel zu erreichen, werden in Suchmaschinenimplementierungen oft mehrere Bewertungsfaktoren kombiniert. Das Retrievalmodell, das die Basis der Suchmaschine bildet,

[2] http://dublincore.org/

[3] z.B. zu viele Begriffe in `<meta name="keywords" content="...">`

[4] http://metager.de/

ermöglicht die Berechnung eines grundlegenden Ranking-Wertes, dieser wird mittels mehrerer anderer Faktoren modifiziert, in Web-Suchmaschinen kann z.B. eine höhere Gewichtung von Suchbegriffen im Titel vorgenommen werden.

3.3.1 PageRank

Als Beispiel soll hier ein Ranking-Verfahren, das eine Grundlage für die bekannte Internet-Suchmaschine Google[5] bildet, dargestellt werden. Der PageRank-Algorithmus wurde in [PBMW98] erstmals vorgestellt. Es handelt sich um ein Verfahren, dass die Hyperlink-Struktur des Webs auswertet und die Seiten am höchsten bewertet, die objektiv als wichtig erachtet werden, erkennbar dadurch, dass sie von vielen wichtigen Seiten verlinkt werden.

PageRank wird mit einem idealisierten Web-Surfer verglichen, der zufällig URLs auswählt und zufällig ausgewählten Hyperlinks folgt.

Es seien u eine Webseite, B_u die Menge der Seiten, die Links auf u beinhalten, N_v die Anzahl von Links, die von der Seite v ausgehen, c ein Faktor zur Normalisierung und $E(u)$ ein Vektor über Webseiten, der zu einer Ranking-Quelle korrespondiert. Dann ist der PageRank ein Wert R', der die Gleichung

$$R'(u) = c \sum_{v \in B_u} \frac{R'(v)}{N_v} + cE(u)$$

so erfüllt, dass c maximiert wird und $||R'|| = 1$ gilt. $||R'||$ ist die L_1-Norm von R'.

Es kann also jeder Webseite ein eindeutiger Rankingwert zugeordnet werden. In der Praxis wird dieser Wert noch mit einem Rankingwert aus der Volltextsuche verschmolzen.

3.4 Konkrete Systeme

Dieser Abschnitt stellt zwei im Rahmen dieser Arbeit untersuchte Web-Suchmaschinen vor. Es wurde die Verwendbarkeit im Rahmen einer Integrationsarchitektur untersucht.

3.4.1 Harvest

Bei Harvest handelt es sich um ein frei verfügbares Softwaresystem zum Einsammeln, Extrahieren, Organisieren und Suchen von Informationen im Internet. Es besitzt eine Komponentenstruktur, die einerseits vielfältige Anpassungsmöglichkeiten eröffnet, andererseits aber auch das Verteilen verschiedener Einzelaufgaben auf mehrere Rechner ermöglicht.

Das SWING-Projekt[6] nutzt diese Eigenschaften, um eine verteilte Suchmaschine zu entwickeln. Die Funktionsweise von Harvest wird in [HSWL01] dokumentiert, hier soll kurz die Komponentenstruktur dargestellt werden.

Gatherer

Der Gatherer sammelt Dokumente von verschiedenen Datenquellen, extrahiert Metainformationen, wie z. B. Schlüsselworte, Autorennamen und Titel und speichert diese im SOIF (Summary Object Interchange Format) ab. Dabei handelt es sich um eine Menge aus Attribut-Wert-Paaren. Die SOIF-Dokumente können dann in Form eines Datenstroms von Brokern abgerufen werden.

[5] http://www.google.de/
[6] http://swing.informatik.uni-rostock.de/

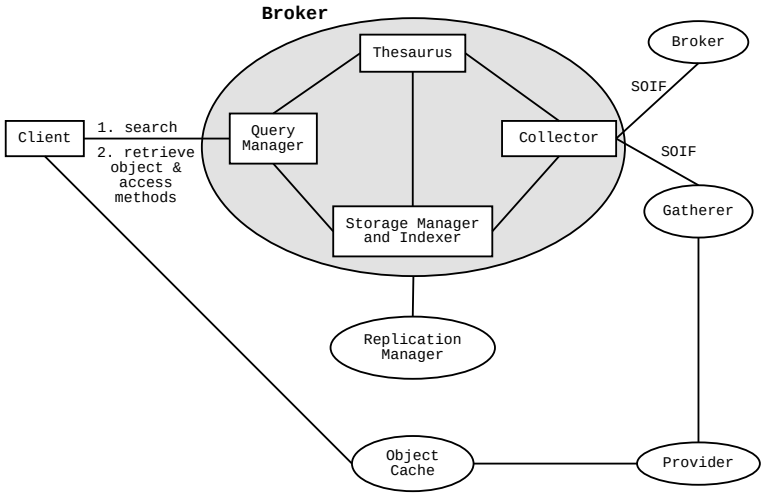

Abbildung 3.2: Architektur von Harvest

Broker

Das Broker-Subsystem liest zu indizierende Informationen von einem oder mehreren Gatherern und entfernt mehrfach vorhandene Daten. Mittels *Index/Search Subsystem* indiziert er die gesammelten Informationen inkrementell und stellt sie über eine WWW-Anfrageschnittstelle zur Verfügung. Bei einer inkrementellen Indizierung werden nur die Dokumente bearbeitet, die seit dem letzten Indizierungslauf geändert wurden.

Index/Search Subsystem

Harvest definiert eine allgemeine Broker-Indexierer-Schnittstelle und ermöglicht somit die Verwendung verschiedener Retrievalsysteme als Indizierungskomponente. Die Standardeinstellung verwendet Glimpse, es wird auch Swish unterstützt. Weiterhin können Retrievalprotokolle wie WAIS über FreeWAIS und Z39.50 über ZQuery unterstützt werden.

Der Funktionsumfang der Anfrageschnittstelle des Brokers hängt von den Fähigkeiten des verwendeten Retrievalsystems ab. Bei Verwendung von Glimpse sind z.B. einfache strukturierte Anfragen innerhalb eines bestimmten SOIF-Attributes möglich.

Replikation

Zur Lastverteilung können die Datenbanken der Broker repliziert werden. Ein Broker kann seine Daten nicht nur von einem oder mehreren Gatherern beziehen, sondern auch einen oder mehrere Broker abfragen. Diese Operation kann auch durch eine Anfrage gefiltert werden. Die Suche findet allerdings immer auf einem Broker statt, eine Anfrage kann nicht verteilt abgearbeitet werden.

Implementierung

Harvest besteht aus einer Menge von Teilkomponenten, die in unterschiedlichen Sprachen programmiert wurden. Es sind C-Programme, Shell- und Perl-Skripte darunter. Die Komponenten werden auf UNIX-spezifische Weise, u.a. durch Pipes miteinander verbunden.

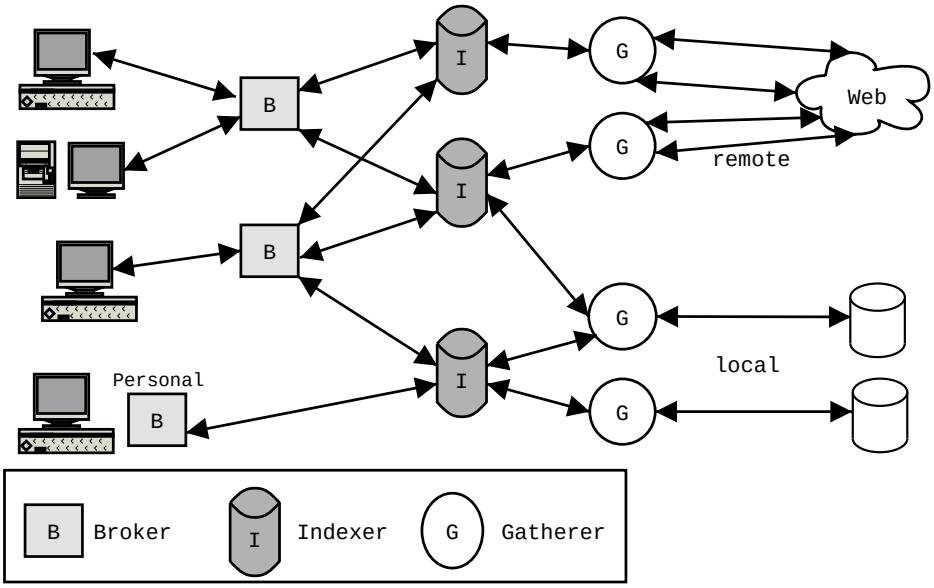

Abbildung 3.3: Aufbau der xFIND-Architektur

Es ist relativ einfach, Harvest zu erweitern. In [Was01] wurde z.B. eine zusätzliche Komponente zur Abfrage von dynamisch generierten Seiten in den Gatherer integriert, ohne dass die ursprünglichen Bestandteile des Systems verändert werden mussten.

3.4.2 xFIND

Bei xFIND (Extended Framework for Information Discovery) handelt es sich um ein skalierbares verteiltes Suchsystem. Es wurde teilweise an Harvest angelehnt, übernommen wurde das Konzept des lokal installierten Gatherers, der die Dokumente einsammelt. In Abbildung 3.3 sind die vielfältigen Anwendungsmöglichkeiten der xFIND-Komponenten dargestellt.

Gatherer

Der Gatherer von xFIND hat die gleiche Funktion wie der Gatherer von Harvest. Er sammelt Dokumente über HTTP oder aus dem lokalen Dateisystem und führt eine Vorverarbeitung durch. Es erfolgt eine Identifizierung von Titel, Schlüsselwörtern, Typ, Sprache, Erstellungs- und Modifikationszeitpunkt. Für jedes Objekt wird eine digitale Signatur berechnet, die u.a. während der Replikation genutzt wird, um ein Objekt eindeutig zu identifizieren.

Die in Dokumenten enthaltenen Metadaten werden ausgewertet. Dabei werden die Formate Dublin Core, LOM[7] und das xFIND-eigene Format unterstützt. Es ist auch eine Konvertierung zwischen diesen Formaten möglich. Da die Metadaten für den Retrieval-Prozess sehr nützlich sind, die Autoren von Dokumenten diese aber selten mit Metadaten anreichern, unterstützt xFIND die Autoren. Es lässt die Definition von Metadaten für eine ganze Dokumentstruktur, ein Verzeichnis oder ein bestimmtes Dokument durch Einfügen zusätzlicher Metadaten-Dateien zu. Speziellere Daten überschreiben dabei allgemeinere.

[7] http://ltsc.ieee.org/wg12/

Der Gatherer ist um zusätzliche Informationsquellen erweiterbar. Aktive Informationssysteme, die sehr dynamisch sind, wie Aktien- oder Nachrichtenticker, können mit xFIND Verbindung aufnehmen und es über neue und veränderte Informationen benachrichtigen. Der Gatherer verarbeitet Metabeschreibungen über Informationsquellen und fasst statistische Daten über die Informationsquellen zusammen.

Indexierer

Der Indexierer erhält die vom Gatherer gesammelten Daten. Ein Indexierer kann auf ein bestimmtes Thema spezialisiert sein oder einer bestimmten Projektgruppe oder Abteilung zugeordnet sein.

Der Indexierer stellt dem Broker statistische Daten wie Term-Frequenzen sowie Beschreibungen der Informationsquellen (z.B. die Anzahl der neuen oder veränderten Dokumente) zur Verfügung.

Es ist auch ein Replikationssystem vorhanden, mit dem die Netzwerklast dadurch reduziert werden kann, dass die Daten oder bestimmte Teilbestände (z.B. abhängig von einem bestimmten Thema oder einer bestimmten Informationsquelle) auf andere Indexierer übertragen werden können.

Broker

Die Broker-Komponente von xFIND trägt die vollständige Bezeichnung *Knowledge Broker*. Er entspricht ebenfalls der verteilten Architektur von xFIND. So ist eine Verteilung von Anfragen auf mehrere Indexierer möglich. Außer den Indexierern können auch weitere Retrievalsysteme abgefragt werden, wenn mittels xFIND API externe Informationsquellen eingebunden werden. Dadurch können dynamisch generierte Informationen eingebunden werden. Existierende Hyperwave-Systeme können direkt durchsucht werden und die Suchergebnisse können mit denen von xFIND gemischt werden.

Aufgrund der verteilten Architektur kann ein Broker individuell für eine Nutzergruppe oder sogar einen einzelnen Benutzer angepasst werden, ohne eine erhöhte Netzlast zu verursachen.

Der Broker ermöglicht es, die Sprache der zu durchsuchenden Dokumente auszuwählen. Er unterscheidet die Suche im Text, im Dokumenttitel und in den Schlüsselwörtern, die vom Indexierer als relevant bewertet wurden. Den drei Kategorien können unterschiedliche Gewichtungen zugeordnet werden.

Bewertungssystem

Eine Spezialität von xFIND ist das Bewertungssystem, das dem Nutzer bei der Erstellung von Qualitäts-Metadaten hilft. Diese Metadaten sind dann ebenfalls durchsuchbar. Die Qualität der Inhalte wird im Format Quality Metadata Scheme (xQMS) gespeichert. Diese Daten können vom Administrator oder von registrierten Experten vergeben werden.

Implementierung

Implementiert wurde xFIND in Java, abgesehen von der Web-Oberfläche, bei der es sich um eine Perl-CGI-Anwendung handelt. Außerdem wird eine relationale Datenbank zur Speicherung von Index-Informationen verwendet. Diese wird über JDBC eingebunden, voreingestellt ist MySQL[8].

[8] http://www.mysql.com/

Dadurch ist xFIND relativ plattformunabhängig. Die einzige Dokumentation (abgesehen vom Quellcode) findet man in [G+00].

3.4.3 Vergleich der Systeme

Man könnte xFIND als Nachfolger von Harvest betrachten: es ist angelehnt an die Architektur, versucht aber die Schwächen von Harvest zu vermeiden. Ein großer Vorteil ist die Trennung von Indexierer und Broker – erst dadurch kann eine Suchanfrage an mehrere Indexierer verteilt werden. Von Vorteil ist auch die Java-API von xFIND, dadurch ist dieses System leichter zu erweitern.

Kapitel 4

Content-Managementsysteme

Der Begriff Content Management ist in der modernen Informationstechnologie zu einem Modebegriff geworden. Das hat dazu geführt, dass es eine Reihe von Produkten auf dem Markt gibt, die als Content-Managementsysteme angepriesen werden, die aber über sehr unterschiedliche Funktionalität verfügen. Darum muss genau festgelegt werden, was im Rahmen dieser Arbeit unter dem Begriff Content-Managementsystem zu verstehen ist.

In [RR01] wird Content Management als ein Prozess beschrieben, der als Teilschritte die systematische Beschaffung, Erzeugung, Aufbereitung, Verwaltung, Präsentation, Verarbeitung, Publikation und Wiederverwendung von Inhalten umfasst. Diese Sichtweise wird in der ebenfalls aus dieser Veröffentlichung entnommenden Abbildung 4.1 dargestellt.

Diese Darstellung steht in weiten Teilen im Widerspruch zu der in dieser Arbeit verwendeten Definition, da wichtige Aspekte wie die Dokumentenspeicherung und -retrieval nicht berücksichtigt werden. Gerade mit diesen Bestandteilen der Content-Managementsysteme beschäftigt sich aber diese Arbeit, darum sollten sie auch in der Definition vorkommen. Die folgende Definition lehnt sich an [Wer01] an.

4.1 Definition

In [Wer01] wurde eine Definition eines Content-Management-Systems vorgenommen. Anschließend wurden mehrere Systeme mit Hilfe dieser Definition klassifiziert.

Content Das erste Wort im Begriff *Content Management* ist definiert als die Vereinigung aller relevanten Teilinformationen. Content-Management-Systeme ver-

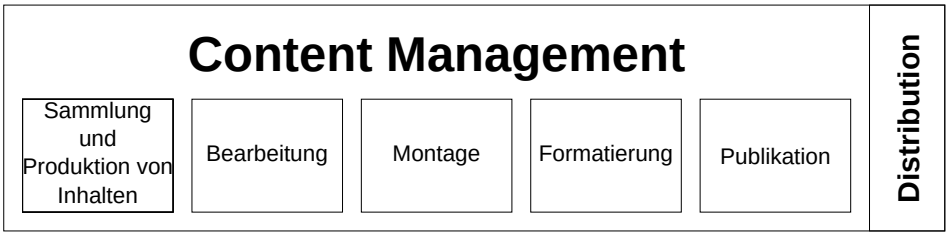

Abbildung 4.1: Content Management als Prozess

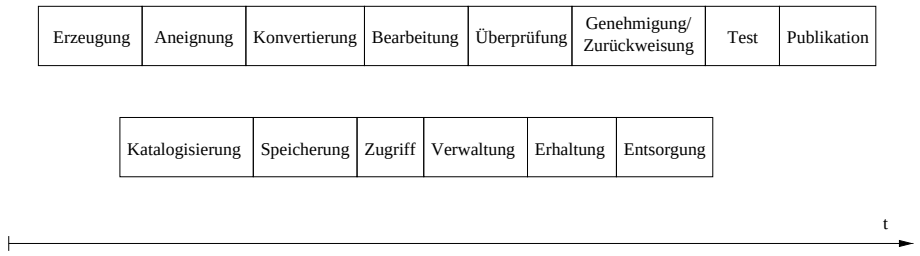

Abbildung 4.2: Bestandteile des Content-Lebenszyklusses

walten unterschiedlichste Arten digitalen Contents. Beispiele dafür sind Notizen, Rechnungen, Web-Seiten, Bilddateien, Multimedia, digitalisierte Buchseiten, sowie Anwendungen und Programmlogik.

Content-Lebenszyklus Alle Inhalte eines CMS durchlaufen eine Reihe von Phasen während ihres Lebenszyklusses. Diese Phasen sind die Erzeugung, Aneignung, Konvertierung, Bearbeitung, Überprüfung, Genehmigung oder Zurückweisung, Test und Publikation der Inhalte. In Abbildung 4.2 wird auch angedeutet, dass parallel zu den Veröffentlichungsprozessen auch die integrierte Speicherung, sowie einige andere damit verknüpfte Prozesse bearbeitet werden.

Web-Content-Managementsysteme Ein Web-CMS besitzt eine Ausrichtung auf WWW-Inhalte. Es werden hauptsächlich Datenformate unterstützt, die im Web Verwendung finden und der Veröffentlichungsprozess ist auf die Erstellung von Web-Sites ausgerichtet. Das bedeutet allerdings nicht, dass ein Web-CMS nicht auch Druckerzeugnisse erstellen kann, der Schwerpunkt liegt dort aber nicht.

Ein Web-CMS lässt sich meistens komplett über den Web-Browser steuern. Es beinhaltet einen Web-Server, der dynamisch generierte Web-Seiten bereitstellt.

4.1.1 Aufbau eines CMS

Man kann ein Content-Managementsystem (CMS) in diese fünf Hauptbestandteile untergliedern, die in Abbildung 4.3 dargestellt wurden. Diesen Teilsystemen können jeweils Teilaufgaben bezüglich des Contents zugeordnet werden.

Kollektionssystem verantwortlich für die Erzeugung, Aneignung und Konvertierung von Inhalten

Publikationssystem Bearbeitung, Überprüfung, Genehmigung bzw. Zurückweisung, Test, Publikation

Speicherungssystem Katalogisierung, Speicherung, Zugriff, Verwaltung, Erhaltung und Entsorgung

Workflow-System Steuerung des Arbeitszyklusses

Administrationssystem Nutzer- und Rechteverwaltung

Das Kollektionssystem ist für das Sammeln der Inhalte verantwortlich, es leitet sie an das Speicherungssystem weiter. Das Publikationssystem bereitet die Inhalte für die Veröffentlichung vor. Eine zentrale Rolle kommt dem Speicherungssystem zu, alle anderen Systeme müssen darauf zugreifen. Zur Implementierung des Speicherungssystems sind sowohl die Funktionalität eines Datenbanksystems als auch IR-Funktionalität notwendig.

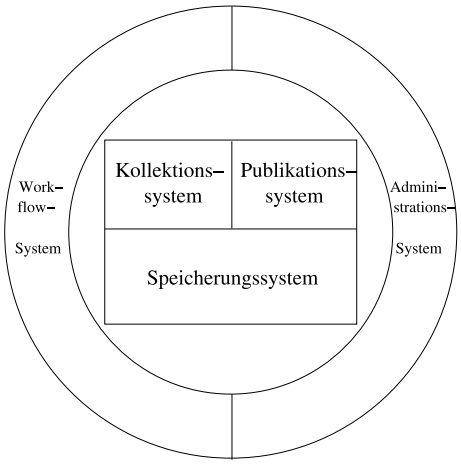

Abbildung 4.3: Bestandteile eines CMS

4.2 Konkrete Systeme

Dieser Abschnitt stellt verschiedene im Rahmen dieser Arbeit untersuchte Content-Managementsysteme vor. Diese Systeme decken unterschiedliche Bereiche des in den vorherigen Abschnitten definierten Spektrums des Content Managements ab.

Es erfolgt eine Klassifikation der Systeme anhand der für diese Arbeit wichtigen Kriterien *Datenmodell*, *Retrievalfunktionalität* und *Programmierschnittstelle*.

4.2.1 IBM Content Manager

Im Rahmen dieser Arbeit wurde die Verbindung aus IBM Content Manager und dem Enterprise Information Portal (EIP) verwendet. In dieser Kombination wird der Content Manager als Datenspeicherungssystem für eine Vielzahl von Inhaltstypen genutzt, wie z.B. Textdokumente, Multimedia-Daten oder Web-Dokumente (HTML, XML). Das EIP stellt Funktionen wie die Application Programming Interface (API) zur schnellen Entwicklung von Portal-Anwendungen und den vereinheitlichten Zugriff auf verschiedene Daten- und Informationsquellen zur Verfügung.

In [Int01] wird der Aufbau des Content Managers beschrieben. Er besitzt eine dreieckige Client-Server-Architektur, die in Abbildung 4.4 dargestellt ist. Sie besteht aus *Library Server*, *Object Server* und *Client*. Jeder dieser Komponenten muss mindestens einmal vorhanden sein, es können aber auch mehrere Object Server und mehrere Clients involviert sein.

Der Library Server ist für die Speicherung der Metadaten zuständig, es werden dort alle Attribute der Indexklassen und auch Daten über die Zugriffsrechte abgelegt. Die Dokumente werden vom Object Server gespeichert. Am Object Server können auch externe Speicherungssysteme wie der Tivoli Storage Manager angeschlossen werden.

Eine Anfrage des Clients nach einem digitalen Datenobjekt läuft folgendermaßen ab: Der Library Server prüft zunächst die Zugriffsrechte, leitet die Anfrage dann an den Object Server weiter und beantwortet die Anfrage des Clients. Der Object Server reagiert auf die Anfrage des Library Servers und liefert das digitale Datenobjekt an den Client.

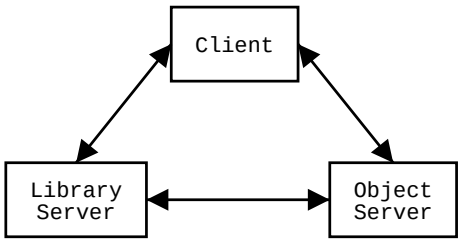

Abbildung 4.4: Dreiecksarchitektur des Content Managers

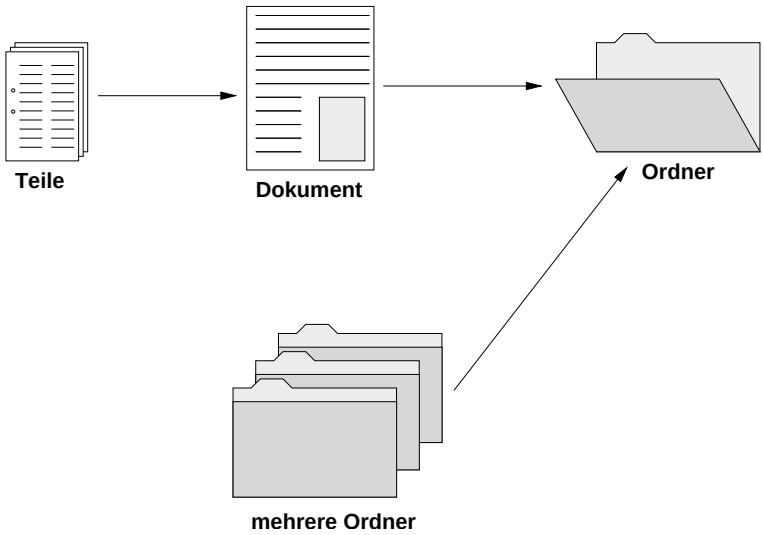

Abbildung 4.5: Datenmodell des EIP: Ordner, Dokumente und Teile

Datenmodell

Das Datenmodell des EIP ist schematisch in Abbildung 4.5 dargestellt. Die vollständige Beschreibung findet man in [Int00b].

Items Items sind die Grundbausteine des EIP. Ein Item kann alles von einem Dokument bis zu einem Ordner repräsentieren. Jedes Item besitzt eine eindeutige und persistente ID-Nummer.

Item-Attribute und Indexklassen Im Content Manager werden Items durch Indexklassen repräsentiert. Eine Indexklasse definiert eine Menge von Attributen, die ein Item beschreiben, das zu dieser Klasse gehört.

Item-Teile Ein Item kann aus einem oder mehreren Teilen bestehen, beispielsweise kann ein Teil ein Multimedia-Objekt oder ein Text sein.

Dokumente und Ordner Ein Dokument ist ein Inhalts-Item, das wiederum aus einzelnen Teilen zusammengesetzt sein kann. Ein Film-Dokument kann beispielsweise aus den Einzelbildern des Films, dem Skript in Textform und einem Foto des Filmplakats bestehen.

Ein Ordner ist ein Container-Item, das Dokumente und andere Ordner enthalten kann.

Retrievalfunktionalität

Der Content Manager unterstützt eine Reihe von Anfragetypen, sowohl die Suche in Attributen als auch im Volltext sowie die Kombination mehrerer Anfragen. Die Kategorie der Anfragen an Bilddaten[1] wird hier außer acht gelassen, da sie für diese Arbeit nicht von Bedeutung ist.

Parametersuche Eine Parameteranfrage startet eine exakte Suche in den Attributen von Indexklassen. Diese Anfrage ist vergleichbar mit einer Datenbank-Anfrage und wird intern auch in eine SQL-Anfrage umgewandelt. Es sind sogar SQL-Operatoren wie *like* und die Platzhalter '%' und '_' in Zeichenketten verwendbar.

Volltextsuche Der IBM Content Manager kann als separat erhältliche Komponente für die Volltextsuche die Text Search Engine verwenden, deren API in [Int00d] beschrieben wird. Es handelt sich dabei um die gleiche Programmkomponente, die auch in anderen Produkten von IBM eingesetzt wird, wie z.B. dem DB2 Text Extender. Im Rahmen dieser Arbeit wurde eine Installation verwendet, in der sie enthalten ist.

Die Text Search Engine bietet einen sehr großen Funktionsumfang, dieser wird in [Int00c] beschrieben. Es werden sowohl strukturierte als auch unstrukturierte Textformate unterstützt. Es werden folgende Anfragetypen unterschieden:

Boolesche Anfrage Eine boolesche Anfrage besteht aus Wörtern oder Wortgruppen, getrennt durch boolesche Operatoren. Wortgruppen werden in Hochkommas eingeschlossen und müssen in genau der gleichen Form im Text auftauchen. Hier ein Beispiel :

```
String cmd ="SEARCH=(COND=(www OR 'web site'));"+
            "OPTION=(SEARCH_INDEX=TMINDEX)";
```

Hinter SEARCH_INDEX steht der zu verwendende Textsuchmaschinenindex.

Freitextanfrage Die Anfrage besteht aus Wörtern, Wortgruppen oder Sätzen in geschweiften Klammern. Die Wörter müssen nicht in der angegebenen Reihenfolge im Text auftreten.

```
String cmd ="SEARCH=(COND=({web site}));"+
            "OPTION=(SEARCH_INDEX=TMINDEX)";
```

Kombinierte Anfrage Sie besteht aus einer booleschen Anfrage, gefolgt von einer Freitextanfrage.

```
String cmd ="SEARCH=(COND=(IBM AND UNIX {web site}));"+
            "OPTION=(SEARCH_INDEX=TMINDEX)";
```

Wortabstandssuche Es werden mindestens zwei Wörter angegeben, die im gleichen Dokument, Absatz oder Satz gesucht werden.

```
String cmd ="SEARCH=(COND=
             ($PARA${'rational numbers' math}));"+
            "OPTION=(SEARCH_INDEX=TMINDEX)";
```

[1] Query By Image Content (QBIC)

Strukturierte Suche Die strukturierte Suche der Textsuchmaschine wird in [Int00a] beschrieben. Sie nutzt die Unterteilung in Abschnitte aus, die bei strukturierten Dokumenttypen vorhanden ist. Es werden HTML, XML oder beliebige andere Textdokumente mit HTML-ähnlichen Tags unterstützt. Vor der Indizierung der Dokumente muss ein Dokumentmodell erstellt werden, das festlegt, welche durch die Tags begrenzten Abschnitte indiziert werden sollen. Das Dokumentmodell ordnet jeweils einem Abschnittsnamen das dazugehörige Tag zu, das diesen Abschnitt begrenzt. Ein Abschnitt kann typisiert werden, außer reinen Volltextabschnitten sind die Typen DATE, TIME, FLOAT und INTEGER erlaubt.

XML-Dokumente müssen eine gültige Document Type Definition (DTD) und ein Wurzelelement besitzen. Das Dokumentmodell für die strukturierte Suche muss eine Untermenge des durch die DTD definierten Dokumenttyps sein. Es besteht aus Pfaden ausgehend von der Wurzel bis zum begrenzenden Tag des Abschnittes. Dadurch sind hierarchische Abschnitte möglich. Ein Beispiel aus einem Dokumentmodell:

```
[MODELS]
modelname = LETTER
; Links steht der Name des Abschnittes, rechts die Tags,
; die den kompletten Pfad vom Wurzelelement zum Tag,
; das den Abschnitt begrenzt, beschreiben.
[LETTER]
LETTER = LETTER
LETTER/date = LETTER/DATE
LETTER/address = LETTER/ADDRESS
LETTER/address/City = LETTER/ADDRESS/CITY
LETTER/Content = LETTER/CONTENT
LETTER/Content/Greetings = LETTER/CONTENT/GREETINGS
```

Ein Beispieldokument zu diesem Dokumentmodell:

```
<?xml version="1.0"?>
<!DOCTYPE LETTER SYSTEM "letter.dtd">
<LETTER>
  <HEADER>Dieses Tag ist nicht in der Modelldefinition
          enthalten, also wird dieser Text zum Abschnitt
          LETTER zugefügt.
  </HEADER>
  <DATE>01.01.2002  03.03.2002</DATE>
  <ADDRESS>Dies ist der Abschnitt namens LETTER/address.
    <CITY>Der Abschnitt LETTER/address/City.</CITY>
  </ADDRESS>
  <CONTENT>Dies gehört zum Abschnitt LETTER/Content.
    <NOSECTION>Auch LETTER/Content, weil NOSECTION nicht
               im Dokumentmodell definiert ist.
    </NOSECTION>
    <GREETINGS>Abschnitt LETTER/Content/Greetings.
    </GREETINGS>
</LETTER>
```

Das obige Modell erlaubt beispielsweise die Suche in einem XML-Dokument, das <LETTER> als Wurzelelement hat, nach einer Adresse, die im Kind-Element <ADDRESS> von LETTER enthalten ist:

```
'MODEL LETTER SECTION (LETTER/address) "Neubrandenburg"'
```

Andere Positionen von <ADDRESS> in der Hierarchie des Dokuments werden nicht berücksichtigt.

GTR Der Vollständigkeit halber muss noch der Anfragetyp GTR (Global Text Retrieval) erwähnt werden, dessen Zweck die Unterstützung von Sprachen ist, die zwei Bytes pro Zeichen zur Kodierung verwenden, wie Japanisch oder Chinesisch. Es werden aber auch Sprachen mit einem Byte pro Zeichen unterstützt, es muss allerdings die richtige Sprachkodierung eingestellt werden.

Programmierschnittstelle

Die API des IBM Content Managers bzw. des EIP besteht aus einer C++- und einer Java-Variante. Damit ist ein vollständiger Zugriff auf die Retrievalkomponente möglich.

Ein Vorteil der Java-Sprachanbindung ist die Verwendbarkeit der Kommunikation über Remote Method Invocation (RMI). Dadurch kann die Client-Anwendung vollständig in Java implementiert werden und ist unabhängig von den Bibliotheken des Content Managers. Ein RMI-Server kann mit mehreren Datenspeichern verbunden sein, die vom EIP unterstützt werden, das sind neben dem Content Manager noch andere Systeme wie Domino.Doc, OnDemand oder über JDBC eingebundene Datenbanksysteme.

4.2.2 Hyperwave

Als kommerzielle Weiterentwicklung des Hyper-G-Systems [M+96] entstand die Firma Hyperwave[2] und das gleichnamige Produkt. Die folgenden Ausführungen beschreiben den Hyperwave Information Server (HWIS) 5.1. Dabei handelt es sich um ein Web-Content-Managementsystem.

Die Architektur des HWIS wird in [Hyp99a] beschrieben. Der Server besteht aus mehreren Modulen die individuell ausgetauscht werden können. Man kann drei Schichten unterscheiden, aus denen HWIS zusammengesetzt ist: die *Protokoll-Konvertierungsschicht*, die *Sitzungsschicht* und die *Datenbankschicht*, siehe Abbildung 10.

Die Protokoll-Konvertierungsschicht wandelt alle von Hyperwave unterstützten Übertragungsprotokolle (u.a. HTTP, FTP, Gopher) in das eigene Protokoll Hyper-G Client-Server-Protokoll (HG-CSP) um, als Gateway für HTTP kommt das Modul WaveMaster zum Einsatz.

Die Sitzungsschicht kommuniziert mit der Datenbankschicht. Für jede Verbindung zu einem Client wird eine Instanz des Sitzungs-Servers *hgserver* gestartet, dadurch werden Zustandsinformationen behalten und es können Anfragen der Clients parallel bearbeitet werden.

In der Datenbankschicht werden Dokumente, Hyperlinks und Metainformationen gespeichert. Die Schicht besteht aus drei Modulen, dem Objektserver, dem Volltext-Server und dem Dokument-Cache-Server. Der Objektserver baut auf einem Datenbanksystem auf, als Alternative zum integrierten Programm wird Oracle unterstützt. Der Volltext-Server verwaltet die Suchindizes über alle Textdokumente. Der Dokument-Cache-Server ist für die Speicherung von lokalen Dokumenten und die Zwischenspeicherung von Dokumenten, die von entfernten Servern geladen wurden, verantwortlich.

[2] http://www.hyperwave.com

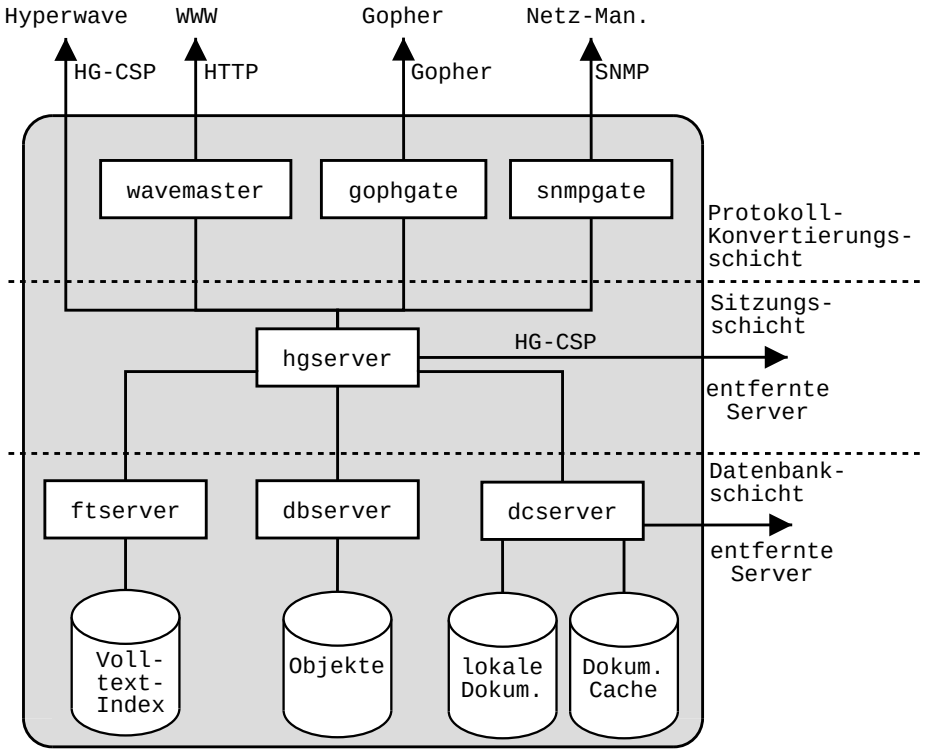

Abbildung 4.6: Architektur von Hyperwave

Datenmodell

Das Datenmodell von Hyperwave ist ausgerichtet auf den praktischen Einsatz im Web, in Präsentationen und Multimediaanwendungen. Die Grundbausteine sind Kollektionen und Dokumente. Kollektionen entsprechen Ordnern, sie können Dokumente und weitere Kollektionen enthalten. Ein Dokument kann gleichzeitig in mehreren Kollektionen referenziert werden, siehe Abbildung 4.7.

Es existieren mehrere Kollektionstypen für verschiedene Anwendungszwecke. Dies sind Kollektion, Sequenz, Sprach-Cluster, Multi-Cluster und Alternativ-Cluster. Die Spezialtypen müssen noch näher erläutert werden:

Sequenz Die Dokumente in einer Sequenz besitzen eine bestimmte festgelegte Reihenfolge, in der sie betrachtet werden sollen. Das System erzeugt automatisch die Hyperlinks zum nächsten und vorherigen Element.

Sprach-Cluster Ein Sprach-Cluster repräsentiert ein mehrsprachiges Dokument. Enthalten sind mehrere Dokumente in unterschiedlichen Sprachen. Es wird beim Anzeigen durch den Nutzer das Dokument ausgewählt, das seiner Spracheinstellung entspricht.

Multi-Cluster Die in einem Multi-Cluster enthaltenen Dokumente werden als zusammengesetztes Dokument angezeigt. Dadurch lassen sich einzelne Bausteine von unterschiedlichen Autoren bearbeiten und in mehreren Dokumenten wiederverwenden.

Alternativ-Cluster Ein Alternativ-Cluster liefert im Gegensatz zum Multi-Cluster nicht alle Dokumente zurück, sondern es wird ein Dokument in Abhängigkeit von den Nutzereinstellungen ausgewählt. Dadurch lässt sich beispiels-

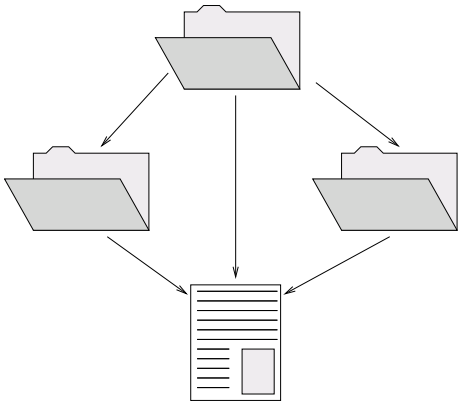

Abbildung 4.7: Kollektionshierarchie in Hyperwave

weise eine Anpassung an die zur Verfügung stehende Bandbreite des Nutzers bewerkstelligen.

Die Dokumente werden in Hyperwave als Objekte abgespeichert, die Metadaten in Form von Attribut-Wert-Paaren beinhalten, wie beispielsweise `Author=waschk`. Die Attribute lassen sich noch folgendermaßen klassifizieren:

- Nutzerdefinierte Attribute werden vom Autor des Dokuments festgelegt. Sie werden noch folgendermaßen unterschieden:
  - Indizierte Attribute. Diese können durchsucht werden, sie müssen vom Administrator entsprechend gekennzeichnet werden.
  - Nicht-Indizierte Attribute. Nach diesen Attributen kann nicht direkt gesucht werden, siehe nächster Abschnitt über die Retrievalfunktionalität.
- Systemdefinierte Attribute werden vom System selbst vergeben, das System legt auch fest, welche indiziert werden müssen.
  - Nur lesbare Attribute werden vom System geschrieben und können vom Nutzer nicht verändert werden. In diese Kategorie fallen u.a. `TimeCreated` und `Size`.
  - Les- und schreibbare Attribute haben eine Bedeutung für das System, können aber vom Nutzer verändert werden. Beispiele sind `Name` und `Rights`.

Retrievalfunktionalität

In [Hyp99c] werden die Retrievalfähigkeiten von Hyperwave beschrieben. Es ist sowohl eine Suche in den Metadaten der Dokumente als auch eine Volltextsuche im Dokumentinhalt möglich. Als Retrievalkomponente für die Volltextsuche kommt in Hyperwave 5.1 entweder die eigene Volltextsuchmaschine oder das Produkt Verity[3] Search97 zum Einsatz. In dieser Arbeit wurde die Variante mit Verity untersucht.

[3] http://www.verity.com

Attributsuche In den Metadaten kann mittels booleschen Anfragen gesucht werden. Es können eingeschränkte Platzhalter verwendet werden: ein '*' am Wortende sucht nach Wörtern mit dem angegebenen Prefix, ein '^' am Wortende steht für Wörter, deren Wert größer oder gleich dem Suchbegriff ist. Hier sind einige Beispiele:

- `Author=waschk || Author=weber`
- `TimeModified>96/03/01<97/01/15`
- `Keyword=Datenbank^ &! Author=waschk`

In der Anfrage können nur indizierte Attribute verwendet werden. Da dieser Anfragetyp dem booleschen Retrievalmodell entspricht, findet kein Ranking der Ergebnismenge statt. In Hyperwave wird dieser Anfragetyp als *key query* bezeichnet.

Volltextsuche Die Volltextsuche in Hyperwave unterstützt laut [Hyp99c] folgende Funktionen:

Thesaurus Es wird eine Synonymsuche mit Hilfe eines Thesaurus durchgeführt.

Stammwortreduktion Es ist eine Suche in einem Index möglich, bei dessen Erzeugung eine Stammwortreduktion durchgeführt wurde.

Phonetische Suche Die phonetische Suche ermöglicht es, nach ähnlich klingenden Begriffen zu suchen.

Groß/Kleinschreibung Es besteht die Auswahlmöglichkeit, ob die Groß- und Kleinschreibung berücksichtigt werden soll oder nicht.

Wortabstandssuche Die Wortabstandssuche ermöglicht verschiedene Angaben für das Abstandsmaß. Man kann angeben, dass alle Wörter im selben Absatz, im selben Satz, direkt aufeinander folgend oder in einem in Wörtern bemessenen Abstand stehen.

strukturierte Suche Die strukturierte Suche soll es ermöglichen, innerhalb eines angegebenen XML-Tags zu suchen. Dies hat in praktischen Versuchen nicht funktioniert, es wurde immer eine Volltextsuche durchgeführt, auch wenn eine Einschränkung auf den Bereich eines bestimmten XML-Tags vorgegeben war.

Kombinierte Anfrage Bei der kombinierten Anfrage werden nacheinander eine Attribut- und eine Volltextsuche durchgeführt, die Trefferlisten werden mit ODER verknüpft. Dabei werden die Treffer der Attributanfrage, die dem booleschen Retrievalmodell entsprechen, mit dem Wert „100%“ rankiert und dementsprechend an den Anfang der Ergebnisliste positioniert.

Objektanfrage Eine Objektanfrage, beschrieben in [Hyp99a], filtert die Ergebnisse einer Attribut- oder Volltextanfrage nach beliebigen Attributen. Im Gegensatz zur Attributanfrage können auch Attribute verwendet werden, die nicht indiziert sind. Ein Beispiel:

```
-keyquery Title=News -query TimeExpire<97/01/01
```

Diese Anfrageform hat den Nachteil, dass die Ergebnisliste der gefilterten Anfrage in jedem Fall komplett durchsucht werden muss. Das führt bei größeren Datenmengen zu langen Wartezeiten.

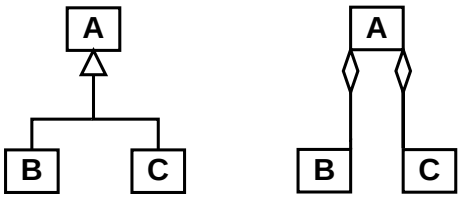

Abbildung 4.8: Vererbung + Aggregation = Akquisition

Beschränkung des Suchbereichs Für alle Anfragetypen ist die Beschränkung des Suchbereichs auf einen Teilbaum der Kollektionshierarchie von Bedeutung. Man kann als Suchbereich eine Kollektion, eine Menge von Kollektionen, den gesamten Server oder eine Menge von Kollektionen auf einem entfernten Hyperwave-Server verwenden.

Programmierschnittstelle

Hyperwave unterstützt mehrere Schnittstellen zur Anpassung des Systems. Zur Personalisierung der Web-Oberfläche kann mittels HTML, JavaScript und CSS erfolgen, es steht die Template-Sprache PLACE [Hyp99b] zur Verfügung. Diese APIs sind hauptsächlich dazu gedacht, Web-Anwendungen mit Hyperwave zu erstellen. Allerdings ist auch eine vom Web unabhängige JavaScript-Implementierung vorhanden, mit der auf die Daten eines Hyperwave-Servers zugegriffen werden kann.

4.2.3 Zope

Z Object Publishing Environment (Zope) ist eine Entwicklungsumgebung für Web-Anwendungen, aufgebaut auf einem Anwendungsserver. Eine der möglichen Anwendungen, die mit Zope realisiert werden können, ist die eines Web-Content-Managementsystems. Der Aufbau und die Funktionsweise von Zope werden in [LP00] beschrieben.

Datenmodell

Zope ist vollständig objektorientiert, auch die Inhalte werden als Objekte mit Attributen und auf sie anwendbaren Methoden implementiert. Dabei kommt ein Objektmodell mit einer vereinfachten Vererbungstechnik namens Akquisition, dokumentiert in [Mau] zum Einsatz. Dabei werden Vererbung und Aggregation verknüpft, wie in Abbildung 4.8 schematisch dargestellt. Anschaulich kann man sich das Datenmodell als hierarchische Ordnerstruktur vorstellen, die Attribute und Methoden werden in der Hierarchie nach unten vererbt. In der Praxis wird dies hauptsächlich eingesetzt, um in allen dynamischen Seiten einheitliche Elemente wie z.B. Menüs einzubinden.

Retrievalfunktionalität

Als Komponente ist in Zope die Suchmaschine ZCatalog integriert. Alle Dokumente, die später durchsucht werden sollen, müssen zuvor katalogisiert werden. Wenn sich ein Dokument geändert hat, muss der Indizierungsprozess manuell gestartet werden, wenn nicht ausschließlich Objekte verwendet werden, die automatisches Katalogisieren unterstützen.

In einem ZCatalog lassen sich verschiedene Indextypen definieren, die unterschiedliche Anfragen unterstützen.

Textindex Ein Textindex für die Volltextsuche ermöglicht boolesche Anfragen und unterstützt bei Verwendung eines speziellen Vokabular-Objekts (*Globbing Vocabulary*) auch '*' als Platzhalter.

Feldindex Mit diesem Index lassen sich Objekte nach spezifischen Werten durchsuchen. Damit können Datumswerte, Zahlen oder Zeichenketten indiziert werden. Als Anfrageformen können exakte Anfragen und Bereichsanfragen verwendet werden.

Schlüsselwort-Index Ein Zope-Objekt kann mittels Schlüsselwörtern kategorisiert werden, ein Objekt kann gleichzeitig in mehreren Kategorien eingeordnet sein. Eine Anfrage an einen Schlüsselwortindex besteht aus einer Liste von Wörtern und liefert alle Objekte, die mindestens eines enthalten. Die Wörter sind also implizit mit oder verknüpft.

Pfadindex Die Ordnerhierarchie der Zope-Objekte korrespondiert mit deren URLs, und eine Pfadanfrage ermittelt die Objekte, deren Pfad den gesuchten Teilstring enthält.

Programmierschnittstelle

Zope ist als Open-Source-System im Prinzip unbegrenzt erweiterbar. Schnittstellen zur Skriptprogrammierung unterstützen Perl und die Implementierungssprache von Zope, Python.

Es existieren sehr viele Erweiterungsmodule für Zope, die sogenannten *Products*. Sie können von der Zope-Seite[4] heruntergeladen werden und stellen Funktionalität wie z.B. XML-Unterstützung zur Verfügung.

Der entfernte Zugriff auf einen Zope-Server ist über die XML-RPC-Schnittstelle[5] möglich. Damit können Zope-Objekte gesteuert werden, es sind aber aus Sicherheitsgründen Einschränkungen vorhanden, die den Aufruf einiger Methoden unterbinden.

4.3 Vergleich der Retrievalfunktionalitäten

Es ist sehr schwer, die drei vorgestellten CMS zu vergleichen, da sie unterschiedlichste Anwendungsbereiche haben. Hyperwave und Zope sind auf Web-Contentmanagement ausgerichtet, während das IBM EIP auf die unternehmensweite Verwaltung großer Datenmengen spezialisiert ist.

Entsprechend unterschiedlich ist auch die Retrievalfunktionalität der Systeme, die in Tabelle 4.3 zusammengefasst wird.

Alle untersuchten Systeme unterstützen eine exakte Suche in den Metadaten, die mit einfachen Anfragen an relationale Datenbanksysteme vergleichbar sind. Unterschiede gibt es bei der Mächtigkeit der Metadatensuche, am meisten Abstriche müssen die Anwender bei Zope machen, dort sind in Anfragen an Schlüsselwortindizes keine Platzhalter möglich, es werden explizit alle Suchbegriffe mit ODER verknüpft.

[4] http://www.zope.org/
[5] http://www.xmlrpc.com

Merkmal	Content Manager	Hyperwave	Zope
Metadatensuche	ja	ja	ja
Boolesche Operatoren	ja	ja	nur ODER
Platzhalter	'*' und '?'	'*' und '^'	keine
Einschränkung des Suchbereichs	ja	ja	ja
Volltextsuche	ja	ja	ja
Boolesche Anfragen	ja	ja	ja
Pattern Matching	ja	ja	mit spez. Index
Hierarchisch strukturierte Anfragen	ja	nein	nein
Stoppworteliminierung	ja	ja	nur englisch
Stammwortreduktion	ja	ja	englisch

Tabelle 4.1: Retrievalfunktionalitäten der CMS im Vergleich

Alle Systeme ermöglichen es, den Suchbereich auf einen Teil der Ordner- oder Kollektionshierarchie einzuschränken.

Die Fähigkeiten bezüglich Volltextsuche sind ebenfalls unterschiedlich, nur der Content Manager besitzt durch die Textsuchmaschine eine funktionierende Suchmöglichkeit in strukturierten Texten.

Kapitel 5

Integrationsarchitekturen

Dieses Kapitel setzt sich mit den unterschiedlichen möglichen Ansätzen zur Integration der in den vorangegangenen Kapiteln vorgestellten Softwaresysteme zur Ermöglichung einer verteilten Suche auseinander. Es werden die Vor- und Nachteile der möglichen Integrationsarchitekturen dargestellt.

5.1 CMS mit integrierter Suchmaschine

Der erste Ansatz zur Kopplung von Suchmaschinen und Content Management Systemen erweitert ein bestehendes Content-Managementsystem um eine Suchmaschinenfunktionalität. Eine Möglichkeit ist die Erweiterung der Retrievalkomponente des CMS um einen speziellen Suchindex, Voraussetzung dafür ist eine Erweiterungsschnittstelle der Retrievalkomponente, siehe Abbildung 5.1.

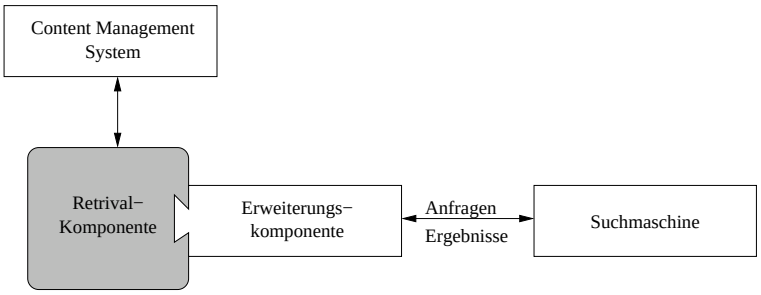

Abbildung 5.1: Erweiterung eines CMS

Der Nutzer stellt Anfragen an das CMS, die Retrievalkomponente übergibt sie an die Erweiterungskomponente. Diese wandelt sie in ein Format um, das von der Suchmaschine verarbeitet werden kann, und leitet sie an die Suchmaschine weiter. Die Ergebnisliste wird von der Erweiterungskomponente entgegengenommen und in das Format des Retrievalsystems konvertiert.

Die von der Suchmaschine indexierte Dokumentkollektion könnte im CMS als spezielle externe Kollektion repräsentiert werden. Hyperwave unterstützt beispielsweise die Einbindung externer Dokumente.

5.1.1 Bewertung

Ein offensichtlicher Nachteil ist die Festlegung auf ein bestimmtes CMS. Dieses System muss eine Erweiterungsmöglichkeit der Retrievalkomponente bereitstel-

len, was nicht selbstverständlich ist.

Bedenkt man die Unterschiede der verschiedenen CMS, gerade auch bezüglich der Erweiterungsschnittstellen, dann ist eine Portierung zu anderen Systemen sehr aufwendig.

5.2 Erweiterung einer Suchmaschine um Anfragen an CMS

Eine weitere Architekturvariante stellt die Integration von Content Management Systemen in eine Suchmaschine dar. Dazu wird eine bestehende Suchmaschine um eine Funktion erweitert, an die zusätzlich zu den bestehenden Indexen Anfragen gestellt werden, wie in Abbildung 5.2 dargestellt.

Diese funktionelle Erweiterung leitet die Suchanfrage an die Retrievalkomponente des Content-Managementsystems weiter. Vorher muss die Anfrage natürlich noch in das von der Retrievalkomponente verwendete Format umgewandelt werden.

Anschließend werden die Suchergebnisse vom Content-Managementsystem gelesen, ebenfalls geeignet konvertiert und mit den anderen Teilergebnissen aus den Indexen der Suchmaschine gemischt. Dabei muss eine geeignete Anpassung des Rankings vorgenommen werden.

5.2.1 Bewertung

Vorteile dieser Architektur sind der Zugriff über die einheitliche Suchschnittstelle, der bei der Verwendung der Oberfläche der Suchmaschine gewährleistet ist. Die Suchtreffer aus dem Suchmaschinenindex und dem CMS müssen nur noch gemischt werden.

Der Nachteil ist die Festlegung auf eine bestimmte Suchmaschine, die dann entsprechend erweitert wird. Damit verbunden ist eine Beschränkung der Retrievalfunktionalität des neuen Systems auf die Fähigkeiten der Suchmaschine, da deren Anfragesprache verwendet wird.

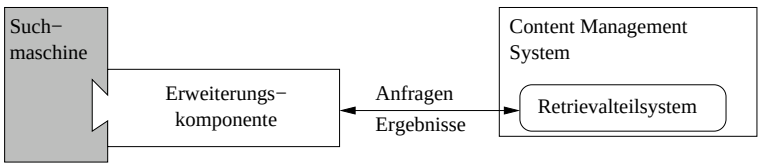

Abbildung 5.2: Erweiterung einer Suchmaschine

5.2.2 xFIND und Hyperwave

Laut [G+00] ist es mit xFIND bereits möglich, in einem Hyperwave-Server zu suchen, der Broker von xFIND leitet seine Anfragen an die xFIND-Indexierer und an die Suchkomponente von Hyperwave weiter. Es handelt sich also um eine Implementierung der hier vorgestellten Architektur.

5.3 Übergeordnete Kopplungskomponente

Der letzte hier vorgestellte Ansatz verbindet Suchmaschinen und die Retrievalkomponenten von Content-Managementsystemen durch eine übergeordnete

Kopplungskomponente, eine sogenannte *Middleware*, siehe Abbildung 5.3. Diese hat eine ähnliche Funktionsweise wie eine Metasuchmaschine. Die Anfrage wird an die eingebundenen Suchmaschinen und Content-Managementsysteme aufgeteilt. Die Suche findet parallel statt, die Ergebnisse der Teilanfragen müssen anschließend gemischt werden. Diese Architektur erfordert keine Festlegung auf ein bestimmtes Content-Managementsystem oder eine bestimmte Suchmaschine. Stattdessen wird für jede Suchmaschine bzw. jedes CMS ein Wrapper benötigt, der Umwandlungen zwischen dem globalen Anfrageformat und dem Teilsystem-spezifischen Format vornimmt.

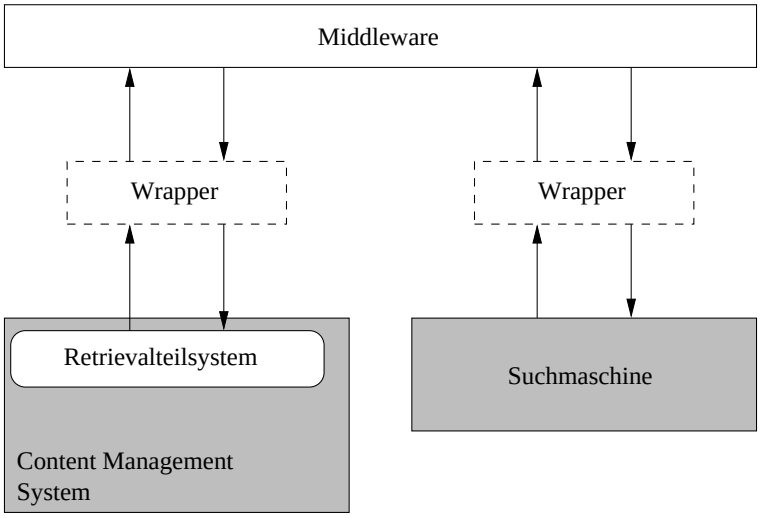

Abbildung 5.3: Middleware zur Integration von Suchmaschinen und CMS

5.3.1 Bewertung

Der Vorteil dieses Modells ist die Gewährleistung maximaler Flexibilität, es ist unabhängig von bestimmten Suchmaschinen und CMS. Es ist ebenfalls unabhängig von den spezifischen Retrievalfähigkeiten der zu integrierenden Teilsysteme.

Der Nachteil ist ein höherer Implementierungsaufwand, die Wrapper müssen für jedes Teilsystem eine Umwandlung der Anfragesyntax und des Ergebnisformates durchführen.

Kapitel 6

Architekturvorschlag

Für die Implementierung wurde die in Abschnitt 5.3 beschriebene Architektur ausgewählt. Eine übergeordnete Kopplungskomponente ermöglicht maximale Flexibilität, benötigt aber für jedes einzubindende Teilsystem einen Wrapper, dargestellt in Abbildung 6.1. In diesem Kapitel werden die Probleme, die diese Architektur aufwirft und deren Lösungsansätze vorgestellt.

Zunächst werden die Anfrageoperationen festgelegt, die der Nutzer an die Anfragekomponente stellen kann, anschließend werden die Integration von Systemen mit unterschiedlichem Funktionsumfang und die Erzeugung einer vereinigten Ergebnisliste erläutert.

6.1 Unterstützte Anfragetypen

Aus dem Vergleich der Content-Managementsysteme in Abschnitt 4.3 sind folgende Anfragetypen ersichtlich, die von den Systemen unterstützt werden:

- **Attributsuche** Die Attributsuche ist eine Suche in indizierten Metadaten, vergleichbar mit einer Anfrage an ein RDBMS. Es sind exakte Anfragen möglich, teilweise werden aber auch Platzhalter unterstützt. Mehrere Attribute können durch boolesche Operatoren verknüpft werden.

  Web-Suchmaschinen besitzen normalerweise keine oder nur eine sehr eingeschränkte Attributsuche, bei xFIND und Harvest ist aber zumindest eine Suche im <title>-Tag der HTML-Dokumente möglich.

- **Volltextsuche** Die Retrievalkomponenten aller untersuchten CMS und alle Suchmaschinen unterstützen die Volltextsuche. Die konkreten Systeme unterscheiden sich hinsichtlich der verfügbaren Funktionen der Volltextsuche, also z.B. Pattern Matching oder die Unterstützung der strukturierten Suche in Texten, die in Abschnitte unterteilt sind.

- **Kombinierte Suche** Es wird die Suche in Attributen und die Volltextsuche kombiniert, beide Suchverfahren werden mit booleschen Operatoren verknüpft.

Alle Anfragetypen haben wiederum die Gemeinsamkeit, dass der Gültigkeitsbereich angegeben werden kann, also eine Einschränkung auf einen Teilbereich der Ordner- oder Kollektionshierarchie vorgenommen wird.

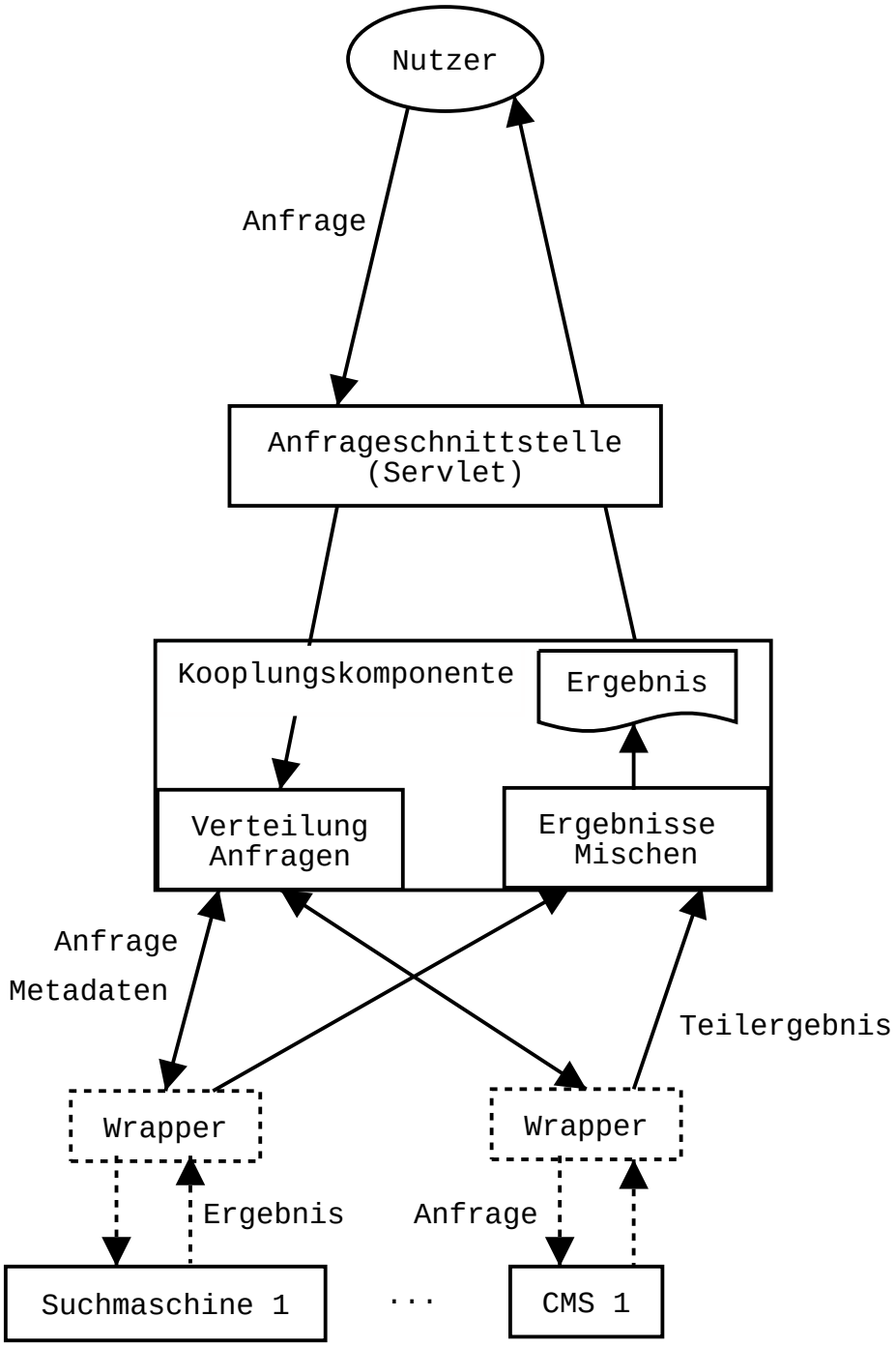

Abbildung 6.1: Kopplungsarchitektur

6.2 Unterschiedliche Retrieval-Fähigkeiten

Die zu integrierenden Retrievalsysteme besitzen einen sehr unterschiedlichen Funktionsumfang. Bei der Integration unter einer gemeinsamen Oberfläche muss entschieden werden, welche Funktionen angeboten werden sollen. Dabei gibt es zwei Möglichkeiten:

1. die Schnittmenge der in allen Systemen verfügbaren Funktionalität verwenden,
2. einen möglichst großen Anteil der Funktionalität jedes Teilsystems einbringen.

Natürlich ist es wünschenswert, auf möglichst wenige Fähigkeiten der Teilsysteme verzichten zu müssen. In diesem Fall muss eine geeignete Anpassung von Teilsystemen mit fehlenden Funktionen vorgenommen werden.

Klassifikation nicht vorhandener Funktionen

Zunächst muss festgestellt werden, welche Retrievalfunktionen fehlen könnten.

- **eingeschränkte Volltextsuche** Die verschiedenen Retrievalsysteme verwenden unterschiedliche Volltextsuchmaschinen mit variierendem Funktionsumfang. Es ist wahrscheinlich, dass nicht alle Systeme über einen Thesaurus oder über Stammwortreduktion verfügen.
- **fehlende Attribute** Bei der Suche in den Metadaten kann man in einer heterogenen Umgebung nicht davon ausgehen, dass überall die gleichen Attribute verwendet werden.
- **fehlende boolesche Operatoren** Nicht alle untersuchten Retrievalsysteme unterstützen alle booleschen Operatoren, z.B. verwendet Zope implizit eine ODER-Verknüpfung der Suchbegriffe bei der Schlüsselwortsuche.

Die folgenden Unterabschnitte beschreiben, wie die Kopplungsarchitektur mit dem Fehlen dieser Funktionen umgeht.

Eingeschränkte Volltextsuche

Die unterschiedlichen Fähigkeiten der Volltextsuchmaschinen haben entscheidenden Einfluss auf die Retrievalqualität einer Teilanfrage. Wenn ein Teilsystem eine Option der Volltextsuche, wie z.B. Stammwortreduktion nicht unterstützt und die Operation trotzdem ausgeführt wird, dann wird die Retrievalqualität dadurch gesenkt.

Die Umwandlung einer Volltextanfrage, die eine bestimmte Option verwendet, in eine Anfrage ohne diese Option funktioniert natürlich nur bei Anfragetypen, deren Syntax sich auf eine einfachere Form abbilden lässt. Dies ist bei den in Abschnitt 2.3.1 beschriebenen Schlüsselwort-basierten Anfragen möglich, eine Pattern-Matching-Anfrage lässt sich aber nicht in eine Folge von Schlüsselwörtern übersetzen.

Fehlendes Attribut in der Anfrage

Die Metadaten der zu integrierenden Retrievalsysteme verwenden unterschiedliche Attribute, in diesem Abschnitt soll der Fall betrachtet werden, dass keine Abbildung des gesuchten Attributs auf ein lokales Attribut, z.B. durch Umbenennen

möglich ist. Es wird in Abhängigkeit von der Bedeutung des Attributs in der Anfrage entschieden, wie diese Situation behandelt wird. Dabei werden zwei Fälle unterschieden:

1. Attribut ist erforderlich

   Das gesuchte Attribut ist unverzichtbarer Teil der Anfrage, in diesem Fall kann die Anfrage auf dem Teilsystem, dem dieses Attribut fehlt, nicht ausgeführt werden.

2. Attribut ist optional

   Das Attribut dient in der Anfrage nur zur Verkleinerung der Ergebnismenge. Es kann weggelassen werden, in diesem Fall besitzt die Anfrage an das Teilsystem aber eine geringere Präzision, was in der Behandlung der Suchtreffer berücksichtigt werden muss.

Fehlende Boolesche Operatoren

Wenn ein Retrievalsystem nicht alle booleschen Operatoren in der Anfrage unterstützt, dann muss die Anfrage in Teilanfragen zerlegt werden. Die Ergebnismengen können dann mittels boolescher Mengenoperationen zum Gesamtergebnis kombiniert werden. Da dies dem booleschen Retrievalmodell entspricht, gehen vorher vorhandene Ranking-Informationen verloren.

Gültigkeitsbereich der Anfrage

Alle untersuchten CMS ermöglichen die Organisation der Dokumente in einer Kollektionshierarchie und unterstützen dies auch in ihren Retrievalkomponenten. Die Daten, die von Web-Suchmaschinen eingesammelt werden, besitzen ebenfalls eine hierarchische Struktur.

Es ist von der Suchmaschinen-Implementierung abhängig, ob die Anfrage an eine Teilhierarchie möglich ist. Bei Google kann man beispielsweise in den erweiterten Suchoptionen die Domain angeben, auf die die Anfrage beschränkt wird.

Im Rahmen dieser Arbeit wird die Angabe eines Gültigkeitsbereichs als optional eingestuft, dadurch kann die Anfrage auch ausgeführt werden, wenn die verwendete Suchmaschine dies nicht unterstützt.

6.2.1 Umwandlung der Anfrage

Eine der Aufgaben der Wrapper ist die Umwandlung der Anfrage aus dem globalen Anfrageformat in das Format des jeweiligen Retrievalsystems. Dabei wird eine Anpassung an die Fähigkeiten des Systems vorgenommen.

In der Konfiguration des Wrappers werden die unterstützten Optionen des Retrievalsystems gespeichert, zusammen mit Umformungsregeln in die lokale Anfragesprache.

Einige globale Anfragen werden in einfachere lokale umgewandelt, wenn die notwendige Option im Retrievalsystem nicht bereitgestellt wird. Die Umformung findet dabei von spezielleren zu allgemeiner formulierten Anfragen statt. Dadurch sinkt die Präzision, was beim Mischen der Teilergebnisse zum Gesamtergebnis berücksichtigt wird.

Beispiel

Die Phrasensuche nach

```
'Information Retrieval'
```

kann in die folgende boolesche Anfrage umgewandelt werden:

```
'Information' AND 'Retrieval'
```

Damit werden dann auch Textdokumente gefunden, in denen *Information* und *Retrieval* zwar im selben Dokument, aber nicht in einer Wortgruppe stehen.

6.3 Mischen der Ergebnisse

Eine wesentliche Aufgabe der Kopplungskomponente ist, wie bei allen verteilten Retrievalsystemen, auf geeignete Weise eine Ergebnisliste aus den Suchergebnissen der Teilretrievalsysteme zu erzeugen.

Wünschenswert ist dabei die Verwendung eines globalen Rankings. In einem idealen verteilten Retrievalsystem liegt allen Teilsystemen das gleiche Retrievalmodell mit einer einheitlichen Ranking-Funktion zugrunde, siehe dazu Abschnitt 2.5.1. Es werden dann ebenfalls globale Statistiken zur Berechnung der Ranking-Werte verwendet. Das führt zu einer vollständigen Transparenz des verteilten Retrievals in Bezug auf das Ranking.

Dieser Idealfall ist bei der Integration verschiedener konkreter CMS und Suchmaschinen nicht gegeben. Folgende Schwierigkeiten treten auf:

1. Boolesches Retrievalmodell ohne Ranking
2. nicht offengelegte Rankingfunktion
3. kein Zugriff auf die Termstatistiken

Wenn kein Retrievalmodell verwendet wird, das Ranking ermöglicht, dann ergeben sich wieder zwei Möglichkeiten: Entweder werden alle Dokumente, die in der Trefferliste auftauchen, zur Kopplungskomponente übertragen und dort mittels noch festzulegender Ranking-Funktion bewertet, oder allen Dokumenten wird ein einheitlicher Wert zugeordnet.

Der erste Ansatz verursacht einen sehr hohen Kommunikationsaufwand und ist deshalb nicht empfehlenswert. Der zweite Fall ist nicht günstig, was das Mischen der Treffer betrifft. Wenn alle Teilsysteme bei einer Anfrage kein Ranking unterstützen, bleibt als Misch-Algorithmus nur das Round-Robin-Verfahren, also die Treffer der Ergebnisliste werden reihum aus den Ergebnismengen der Teilsysteme entnommen. Verwendet hingegen nur ein Teil der Retrievalsysteme kein Ranking, dann kommt es zu einer Anhäufung der einheitlich bewerteten Treffer dieser Systeme. Wenn man eine aufwendige Neuberechnung des Ranking-Werts vermeiden will, kann man diesen Effekt nur Abmildern, indem man die Anzahl von Treffern ohne Rankingwert beschränkt.

6.3.1 Berechnung der Ranking-Werte

Die Ausgangsbasis ist folgende: Die Teilsysteme liefern entweder nicht gerankte Ergebnismengen oder bewertete Ergebnislisten an die Kopplungskomponente zurück. Wenn die Systeme Ranking-Werte zurückliefern, dann wird deren Berechnungsgrundlage nicht offenbart. Trotzdem müssen daraus globale Ranking-Werte berechnet werden. Man kann allerdings die Teilsysteme mit Hilfe einer Abschätzung der von ihnen gelieferten Retrievalqualität bewerten. Man berechnet einen Faktor, mit dem die normalisierten lokalen Ranking-Werte multipliziert werden, um näherungsweise einen globalen Ranking-Wert zu erhalten.

Der Bewertungsfaktor eines Retrievalsystems wird für jede Anfrage neu berechnet. Die Grundidee zur Herleitung des Berechnungsfaktors ist die Annahme, dass die Retrievalqualität niedriger ist, wenn eine in der Anfrage enthaltene

Anfrageoption	System A	System B	System C
Wortabstand	nein	nein	ja
Groß/Kleinschreibung	ja	ja	ja
Stammwortreduktion	nein	ja	ja
Bewertung	33%	66%	100%

Tabelle 6.1: Beispiel: Bewertung der Anfragen

Retrieval-Funktion nicht unterstützt wird und eine Umformung der Anfrage, wie in Abschnitt 6.2 beschrieben, durchgeführt werden muss. Es wird die vereinfachende Annahme getroffen, dass die grundlegende Retrievalqualität aller Teilsysteme gleich ist.

Die von der Anfrage verwendeten Funktionen werden mit denen vom System unterstützten verglichen. Wenn eine Funktion nicht vorhanden ist, sinkt die Retrievalqualität und der Bewertungsfaktor des Teilsystems für die Anfrage wird verkleinert.

Beispiel

Tabelle 6.1 illustriert das Verfahren zur Berechnung der Bewertungsfaktoren. Es wurde eine Anfrage mit den Optionen Wortabstandssuche, Unterscheidung von Groß- und Kleinschreibung und der Benutzung eines Indexes mit Stammwortreduktion verwendet. Die Retrievalsysteme A und B unterstützen nicht die vollständige Anfrage, sie erhalten einen geringeren Bewertungsfaktor. In diesem Fall wurden alle Optionen der Volltextanfrage gleich behandelt, in der Implementierung sollten Heuristiken entwickelt werden, um sie realistischer zu bewerten.

Berechnung der Wichtungsfaktoren

Die endgültigen Wichtungsfaktoren berechnen sich aus den oben erläuterten Bewertungsfaktoren nach folgender, an [BYRN99] angelehnte Formel:

$$w = 1 + \frac{s - \bar{s}}{|C|\bar{s}} \tag{6.1}$$

Darin sind w der Wichtungsfaktor, $|C|$ die Anzahl der Retrievalsysteme, s der Bewertungsfaktor des Teilsystems und $\bar{s}$ der durchschnittliche Bewertungsfaktor.

Der globale Ranking-Wert $r_g(d_j)$ eines Dokumentes d_j berechnet sich als Produkt des normalisierten lokalen Ranking-Werts mit dem Wichtungsfaktor w_i des Teilretrievalsystems i:

$$r_g(d_j) = w_i \frac{r_i(d_j)}{\max r_i}$$

$r_i(d_j)$ ist der Ranking-Wert des Dokumentes d_j im Retrievalsystem i. $\max r_i$ ist der maximale Ranking-Wert des Retrievalsystems i.

Kapitel 7

Implementierung

Dieses Kapitel dokumentiert die im Rahmen dieser Arbeit entstandene prototypische Implementierung. Es wurde das in Kapitel 6 eingeführte Architekturmodell für jeweils eine ausgewählte Suchmaschine und ein CMS implementiert.

Da sowohl xFIND als auch das Enterprise Information Portal (EIP) über eine Java-API verfügen, was die Integration erleichtert, wurden diese beiden Systeme ausgewählt.

7.1 Aufbau

Der grobe Aufbau der Implementierung wurde bereits in Abbildung 6.1 auf Seite 40 dargestellt. Jetzt werden die einzelnen Bestandteile näher erläutert.

7.1.1 Nutzerschnittstelle

Die Bedienoberfläche wurde als Servlet in Java implementiert. Der Nutzer stellt die Anfragen mit Hilfe eines Web-Browsers an dynamisch generierte Webseiten. Die Anfragekomponente erzeugt aus den Nutzereingaben den Anfragetext, der dann an die Wrapper verteilt wird.

7.1.2 Wrapper

Die Wrapper bilden die Hauptbestandteile der Kopplungsarchitektur. Sie beinhalten die Komponente zur Konvertierung der Anfragen, die systemspezifische Definition der Anfrageumwandlung und den IR-System-Treiber.

Der Aufbau und die Funktionsweise dieser zentralen Komponente werden in Abbildung 7.1 dargestellt. Ein Wrapper besteht aus einem XSLT-Parser, der aus der globalen Anfrage und der für das Teilsystem spezifischen Konfigurationsdatei in Form eines Stylesheets den lokalen Anfragetext erzeugt. Ein Nebenprodukt dieses Vorgangs sind die Daten über die verwendeten Optionen, die zusammen mit der Ergebnisliste an die Ranking-Komponente übertragen werden.

IR-System-Treiber

Für jedes einzubindende CMS und jede Suchmaschine muss ein Treiber implementiert werden, der für die Kommunikation mit dem Retrievalsystem zuständig ist. Er verwendet die API des Retrievalsystems, ruft die Anfragefunktion mit dem Anfragetext als Parameter auf und nimmt die Ergebnisliste entgegen. Diese wird in ein einheitliches Format konvertiert und an die Ranking-Komponente übertragen.

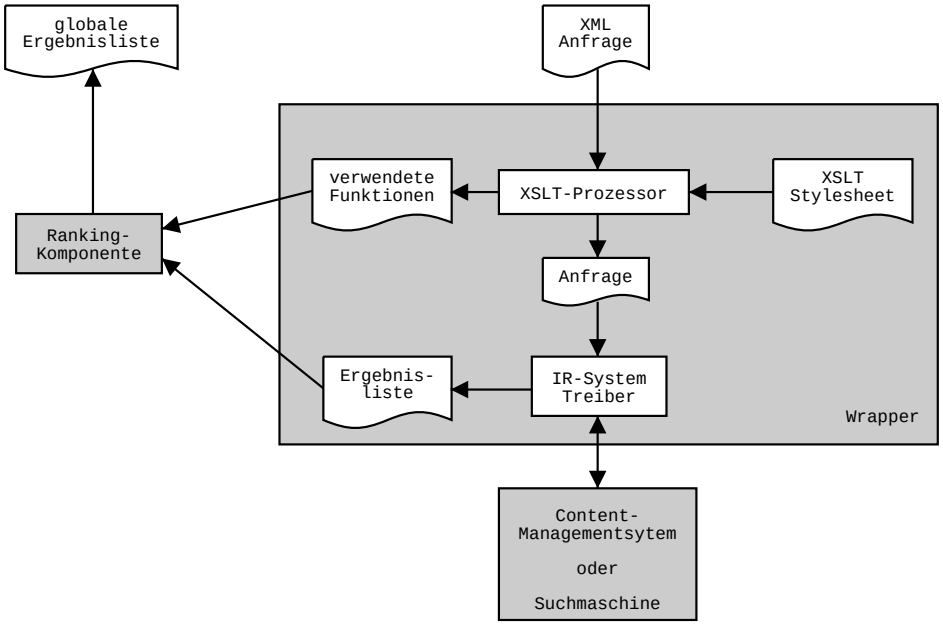

Abbildung 7.1: Aufbau des Wrappers

7.1.3 Ranking-Komponente

Die Ranking-Komponente ist für das Erzeugen der global gerankten Ergebnisliste zuständig. Sie empfängt die Ergebnislisten der Wrapper und berechnet aus den Daten über die Anfragefunktionalität des Retrievalsystems die Wichtungsfaktoren.

7.2 Verarbeitung der Anfragen

Die Anfragekomponente erzeugt die globalen Anfragen in einem speziellen XML-Format. Dies ermöglicht die Formulierung aller in Abschnitt 6.1 vorgestellten Anfragetypen, einschließlich der möglichen Volltextsuchoptionen. Die allgemeinen Anfragen können regelbasiert in die Formate der konkreten Retrievalsysteme umgewandelt werden.

Hier ist eine globale Beispielanfrage, die verwendeten Optionen sind Phrasensuche und die Wortabstandssuche:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="local.xsl" ?>
<query>
  <para>
    <phrase>
      <term>rational</term>
      <term>numbers</term>
    </phrase>
    <term>math</term>
  </para>
</query>
```

Die Anfrage sucht nach der Wortgruppe „rational numbers“ im gleichen Absatz wie „math“.

7.2.1 Erzeugung der lokalen Anfragen

Die globalen Anfragedokumente können mittels XSL Transformationen (XSLT), beschrieben in [Cla99], in das lokale Format umgewandelt werden. Eigentlich wurde XSLT entwickelt, um XML-Dokumente in andere XML-Dokumente umzuwandeln. Das Prinzip basiert auf der Umwandlung eines Ausgangsbaums in einen Zielbaum mit Hilfe von Schablonen, die in Form von XSLT-Stylesheets angegeben werden. Der Mechanismus ist flexibel genug, um andere Zielformate als XML zu erzeugen und es existieren gute Werkzeuge, die die XLST-Spezifikation erfüllen. Aus diesen Gründen wurde das Verfahren zur Transformation der Anfragen ausgewählt.

Der folgende Stylesheet konvertiert die obige Beispielanfrage in das Format des Content Managers:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output cdata-section-elements="term" method="text"/>
  <xsl:template match="query">
  SEARCH=(COND=(<xsl:apply-templates />));
  </xsl:template>
  <xsl:template match="para">
  $PARA${<xsl:for-each select="*[position() != last()]">
           <xsl:apply-templates select="."/>
         </xsl:for-each>
         <xsl:apply-templates select="*[last()]"/>}
  </xsl:template>
  <xsl:template match="phrase">
  '<xsl:for-each select="*[position() != last()]">
     <xsl:apply-templates select="."/>
     <xsl:text> </xsl:text>
   </xsl:for-each>
   <xsl:apply-templates select="*[last()]"/>'
  </xsl:template>
</xsl:stylesheet>
```

Daraus wird diese Anfrage erzeugt, nach Entfernung überflüssiger Zeilenumbrüche:

```
SEARCH=(COND=($PARA${'rational numbers' math}));
```

Der folgende Stylesheet demonstriert die Umwandlung der Anfrage in eine allgemeinere, in diesem Fall für die Verwendung in xFIND:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output cdata-section-elements="term" method="text"/>
  <xsl:template match="query">
    <xsl:apply-templates />
  </xsl:template>
  <xsl:template match="para">
    <xsl:for-each select="*[position() != last()]">
      <xsl:apply-templates select="." /> AND </xsl:for-each>
    <xsl:apply-templates select="*[last()]"/>
  </xsl:template>
```

```
  <xsl:template match="phrase">
    <xsl:for-each select="*[position() != last()]">
      <xsl:apply-templates select="."/> AND
    </xsl:for-each>
    <xsl:apply-templates select="*[last()]"/>
  </xsl:template>
</xsl:stylesheet>
```

Die Anfrage in xFIND lautet dann:

```
rational AND numbers AND math
```

Die ursprüngliche Semantik konnte nicht abgebildet werden, es werden jetzt die drei Begriffe „rational“, „numbers“ und „math“ im selben Dokument gesucht.

Nicht immer existiert eine Abbildung für alle Anfragetypen. Ein Beispiel dafür ist ein Retrievalsystem, das Pattern Matching nicht unterstützt. In dem Fall schlägt die Umwandlung der Anfrage fehl und der Ranking-Komponente wird mitgeteilt, dass das Teilsystem von dieser Anfrage ausgeschlossen wird.

7.2.2 Statistik der Anfrageumwandlung

Bei der Umwandlung der XML-Anfrage in das lokale Anfrageformat wird eine Statistik über die Abbildung der enthaltenen Operatoren erzeugt. Es handelt sich dabei um eine Liste von Tupeln, die aus dem Operatornamen und einem Wahrheitswert bestehen. Dieser Wert ist 1, wenn der Operator in die lokale Syntax übertragen werden konnte und 0, wenn eine Umwandlung in eine allgemeinere Form durchgeführt werden musste.

Für das laufende Beispiel ergibt sich für den Content Manager:

$$\{(\text{para}, 1), (\text{phrase}, 1)\}$$

und für xFIND:

$$\{(\text{para}, 0), (\text{phrase}, 0)\}$$

Diese Liste wird an die Ranking-Komponente übertragen, um dann den Bewertungsfaktor des Teilsystems für die Anfrage und schließlich den Wichtungsfaktor zu berechnen.

7.3 Berechnung des globalen Rankings

Die Ranking-Komponente hat als Grundlage für die Erzeugung der globalen Ergebnisliste die normalisierten lokalen Ranking-Werte der Teilsysteme und die in Abschnitt 7.2.2 beschriebene Statistik über die Umwandlung der Anfrage.

Die Bewertungsfaktoren s werden nach dieser Formel berechnet:

$$s = \sum_{i=1}^{n} f_i o_i \text{ mit } o_i \in \{0, 1\}$$

Darin sind o_i die den Operatoren der Anfrage zugeordneten Wahrheitswerte und f_i die Gewichte der Operatoren.

Nach Formel 6.1 werden aus den Bewertungsfaktoren die Wichtungsfaktoren berechnet. Die Wichtungsfaktoren werden dann mit den lokalen Ranking-Werten multipliziert.

Im laufenden Beispiel ergeben sich bei 2 Teilsystemen und allen Operatorengewichten $f_i = 1$ die Wichtungsfaktoren 1.5 für den Content Manager und 0.5 für xFIND.

7.3.1 Teilergebnisse ohne Ranking

Die Suchergebnisse der Teilanfragen ohne Ranking, also z.B. entsprechend dem booleschen Retrievalmodell, werden alle mit einem einheitlichen Ranking-Wert versehen. Es wurde eine Möglichkeit implementiert, die Trefferanzahl dieser Teilanfragen auf einen festen Maximalwert zu begrenzen. Damit wird der Annahme gefolgt, dass der Nutzer hauptsächlich an den gerankten Suchtreffern interessiert ist.

Kapitel 8

Zusammenfassung und Ausblick

In dieser Arbeit wurden die Fähigkeiten von zwei unterschiedlichen Systemen zur Suche in Textdokumenten kombiniert. Suchmaschinen und Content-Managementsysteme haben jeweils einen anderen Schwerpunkt.

Bei den Suchmaschinen liegt dieser im Textretrieval. Sie ermöglichen die Suche in Dokumenten, die außerhalb des Systems, z.B. auf Web-Servern oder in lokalen Dateisystemen gespeichert sind.

Content-Managementsysteme sind spezialisiert auf die Integration von Inhalten unterschiedlichster Art, wie semistrukturierte Texte, Web- und Multimediadaten. Sie stellen vielfältige Funktionen für den Umgang mit diesen Daten bereit, unter anderem auch die eines Retrievalsystems.

Die untersuchten Suchmaschinen sind hauptsächlich zur Volltextsuche geeignet. Content-Managementsysteme haben die volle Kontrolle über die von ihnen verwalteten Inhalte, sie speichern sie hierarchisch strukturiert und mit Metadaten angereichert. Dadurch ermöglichen sie andere, komplexere Anfrageoperationen.

Es existieren eine Reihe von verteilten Retrievalsystemen auf der Basis von Web-Suchmaschinen. Diese kombinieren die Fähigkeiten der parallel durchsuchten Teilsysteme zu einer Meta-Suchmaschine. Bislang existierte kein vergleichbares System, das die speziellen Fähigkeiten von Content-Managementsystemen ausnutzt und unter einer gemeinsamen Oberfläche vereinigt.

In der Arbeit wurden mehrere verfügbare Content-Managementsysteme und Suchmaschinen bezüglich ihrer Retrievalfunktionalität analysiert. Anschließend wurden verschiedene Kopplungsarchitekturen diskutiert und schließlich eine in Form eines Prototyps realisiert.

8.1 Ergebnisse

Als wichtigstes Ergebnis hat diese Arbeit gezeigt, dass die Integration der Retrievalfunktionalität von Content-Managementsystemen und Suchmaschinen möglich ist. Es konnten Systeme mit einem sehr unterschiedlichen Funktionsumfang verwendet werden, um dieselbe Anfrage zu bearbeiten.

Traditionelle Ansätze, die eine Extraktion der Daten aus dem Content-Managementsystem und eine Indexierung durch eine Suchmaschine verlangen, verursachen einige Probleme. So werden die Inhalte mehrfach gespeichert und die Suche arbeitet somit nicht mit den aktuellen Daten, sondern verwendet den Stand der letzten Indizierung.

Der in dieser Arbeit vorgestellte Architekturentwurf vermeidet diese Probleme. Keine Daten werden redundant gespeichert, die Dokumente verbleiben unter der Kontrolle des Content-Managementsystems. Das Speicherungs- und Retrievalsystem des CMS trägt die Verantwortung für die Aktualität der Suchergebnisse.

Durch die Flexibilität des Wrapper-Konzepts und die Verwendung von Standardkomponenten bei der Umwandlung der Anfragen ist die Erweiterung der Architektur um zusätzliche Systeme ohne großen Aufwand möglich.

Probleme bereiteten die Programmierschnittstellen der Retrievalsysteme. Oft sind das Retrievalmodell und damit das Verfahren zur Berechnung der Ranking-Werte nicht offengelegt. Dadurch wird die Erzeugung globaler Ergebnislisten erschwert. Wenn keine globalen Termstatistiken verfügbar sind, können die Verfahren aus bestehenden verteilten Retrievalsystemen, die z.B. in [BYRN99] vorgestellt wurden, nicht angewendet werden.

Die hier implementierten Bewertungsalgorithmen bieten nur eine grobe Näherung für das globale Ranking. Die lokal in den Teilsystemen erzeugten Ranking-Werte müssen verwendet werden, ohne ihre Bedeutung zu kennen.

8.2 Ausblick

Der hier vorgestellte Prototyp kann noch in einigen Punkten verbessert werden. Er demonstriert die verteilte Anfragebearbeitung, hat aber einige Probleme beim globalen Ranking. Es wäre lohnenswert zu untersuchen, wie eine Neuberechnung der Ranking-Werte durch die Kopplungsarchitektur bewerkstelligt werden könnte, ohne dass die kompletten Dokumente übertragen werden müssen. Dazu könnten die im CMS zur Verfügung stehenden Metadaten genutzt werden. Durch eine Neubewertung der Suchtreffer ließen sich auch die nicht gerankten Ergebnismengen besser integrieren.

Im Architekturentwurf wurde stillschweigend angenommen, dass sich die Dokumentkollektionen der Teilsysteme nicht überlappen. Darum wurde eine Dublettenliminierung nicht eingeplant. Dies mag bei mehreren eingebundenen Content-Managementsystemen zutreffend sein, bei mehreren Web-Suchmaschinen kann es aber zu Problemen führen. Eine Erweiterung zu einer vollständigen Meta-Suchmaschine nach den Kriterien in Abschnitt 3.2.2 sollte aber nicht schwer sein.

Abkürzungsverzeichnis

API Application Programming Interface

CGI Common Gateway Interface

CMS Content-Managementsystem

DBMS Datenbank-Management-System

DR Data Retrieval

DTD Document Type Definition

EIP Enterprise Information Portal

HWIS Hyperwave Information Server

IR Information Retrieval

LOM Learning Object Metadata

RDBMS Relationales Datenbank-Management-System

RMI Remote Method Invocation

SWING Suchdienst für WWW-basierte Informationssysteme der nächsten Generation

URL Uniform Resource Locator

VRM Vektorraummodell

xFIND Extended Framework for Information Discovery

XML Extensible Markup Language

XSL Extensible Stylesheet Language

XSLT XSL Transformations

Zope Z Object Publishing Environment

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[BYRN99] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, 1999.

[Cla99] James Clark. XSL Transformations (XSLT). W3C recommendation, W3C, November 1999. http://www.w3.org/TR/xslt.

[EF00] Albert Endres and Dieter W. Fellner. *Digitale Bibliotheken: Informatik-Lösungen für globale Wissensmärkte*. Datenbanken und Informationssysteme. dpunkt-Verlag, 2000.

[Fuh98] Norbert Fuhr. *Information Retrieval Skriptum zur Vorlesung*. Universität Dortmund, 1998. http://ls6-www.cs.uni-dortmund.de/ir/teaching/courses/ir/script/irskall.ps.gz.

[G+00] C. Gütl et al. *Extended Framework for Information Discovery xFIND V0.95 Programmers Documentation*. IICM, TU-Graz, Austria, 2000. ftp://xfind.iicm.edu/pub/Archive/xFIND-0.95/Documentation/xFIND_ProDoc_V095.pdf.

[HSWL01] Darren R. Hardy, Michael F. Schwartz, Duane Wessels, and Kang-Jin Lee. *Harvest User's Manual*, May 2001. http://www.arco.de/~kj/harvest/manual/.

[Hyp99a] Hyperwave Information Management, Inc. *Hyperwave Administrator's Guide*, September 1999.

[Hyp99b] Hyperwave Information Management, Inc. *Hyperwave PLACE Workshop*, June 1999.

[Hyp99c] Hyperwave Information Management, Inc. *Hyperwave User's Guide*, August 1999.

[Int00a] International Business Machines Corporation. *DB2 Text Extender Administration and Programming*, June 2000.

[Int00b] International Business Machines Corporation. *IBM Enterprise Information Portal for Multiplatforms Application Programming Guide*, August 2000. http://www-3.ibm.com/software/data/eip/pubs/v7/apgoo.pdf.

[Int00c] International Business Machines Corporation. *IBM Enterprise Information Portal Managing Enterprise Information Portal*, first edition, August 2000. http://www-3.ibm.com/software/data/eip/pubs/v7/eipmnge.pdf.

[Int00d] International Business Machines Corporation. *Text Search Engine Application Programming Reference*, first edition, July 2000. http://www-3.ibm.com/software/data/eip/pubs/v7/txtse.pdf.

[Int01] International Business Machines Corporation. *IBM Content Manager for Multiplatforms System Administration Guide*, January 2001. http://www-4.ibm.com/software/data/cm/pubs/cm71/sysadm/sysadm.pdf.

[LP00] Amos Latteier and Michel Pelletier. *The Zope Book*. New Riders Publishing, 2000. http://www.zope.org/Members/michel/ZB/.

[M+96] Hermann Maurer et al. *Hyper-G now Hyperwave™ The Next Generation Web Solution*. Addison-Wesley, 1996.

[Mau] Dieter Maurer. Building Dynamic Web Sites with Zope. http://www.dieter.handshake.de/pyprojects/zope/book/chap1.html.

[Nit00] Raiko Nitzsche. Kopplung von objektrelationalen Datenbanksystemen mit externen Textretrieval-Werkzeugen. Diplomarbeit, Universität Rostock, Fachbereich Informatik, 2000.

[PAT] University of Michigan. *Opentext Pat 5.0 Manual*. http://www.hti.umich.edu/sgml/pat/pat50manual.html.

[PBMW98] Larry Page, Sergey Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford University, January 1998. http://www-db.stanford.edu/~backrub/pageranksub.ps.

[Rij79] C. J. van Rijsbergen. *Information Retrieval*. Department of Computing Science University of Glasgow, 1979. http://www.dcs.gla.ac.uk/Keith/Preface.html.

[RR01] Gunther Rothfuss and Christian Ried. *Content Management mit XML*. Springer, 2001.

[SB00] Wolfgang Sander-Beuermann. Neues und Megatrends bei Suchmaschinen. In *InetBib2000*, March 2000. http://metager.de/inetbib2000/.

[SBS98] Wolfgang Sander-Beuermann and Mario Schomburg. Internet information retrieval - the further development of meta-searchengine technology. In *Proceedings of the Internet Summit*, Genf, July 1998. Internet Society. http://www.uni-hannover.de/inet98/paper.html.

[SM87] Gerard Salton and Michael McGill. *Information Retrieval: Grundlegendes für Informationswissenschaftler*. McGraw Hill, 1987.

[Was01] Götz Waschk. Integration von dynamischen Inhalten in Suchmaschinen mit Hilfe von Metadaten. Studienarbeit, Universität Rostock, Fachbereich Informatik, September 2001. http://e-lib.informatik.uni-rostock.de/fulltext/2001/pre-diploma/WaschkGoetz-2001.ps.gz.

[Wer01] Daniela Wersin. Evaluation and comparison of content management systems. Bachelor thesis, University of Rostock, August 2001. http://e-lib.informatik.uni-rostock.de/fulltext/2001/pre-diploma/WersinDaniela-2001.pdf.

Thesen

1. Content-Managementsysteme stellen umfangreiche Retrievalfunktionalität zur Volltextsuche und zur Suche in den Metadaten bereit.

2. Durch die flexible Kopplungsarchitektur können Systeme mit sehr unterschiedlichen Fähigkeiten unter einer gemeinsamen Oberfläche integriert werden.

3. Eine Anpassung an den eingeschränkten Funktionsumfang eines Teilsystems ist mittels regelbasierter Umformung der Anfrage möglich.

4. Das Mischen von gerankten Suchergebnissen mit ungerankten Ergebnismengen zu einer globalen Ergebnisliste ist problematisch, da ein gemeinsames Vergleichskriterium zur Einsortierung der Suchtreffer fehlt.

5. XML ist als Austauschformat für die Suchanfragen geeignet, weil sich damit beliebig verschachtelte Syntaxbäume darstellen lassen, die mittels XSLT in andere Formate umgewandelt werden können.

6. Durch die Verwendung von Standardkomponenten und festen Schnittstellen sinkt der Aufwand bei der Integration der Teilsysteme.