

GETESS: Zum Einsatz einer Abstract-Datenbank als Index einer Internet-Suchmaschine

Ilvio Bruder Antje Düsterhöft Andreas Heuer Meike Klettke
Denny Priebe

Lehrstuhl Datenbank- und Informationssysteme
Fachbereich Informatik, Universität Rostock
D-18051 Rostock
E-Mail: getess@informatik.uni-rostock.de

Zusammenfassung

Ziel des Projektes GETESS ist die Realisierung eines Suchdienstes, der sich von existierenden Suchmaschinen durch die inhaltliche Analyse der zu untersuchenden deutschen WWW-Dokumente, d.h. durch den Einsatz von Informationskondensaten (*Abstracts*) dieser Dokumente auf Basis von linguistischem und Ontologie-Wissen sowie die Verwendung einer natürlichsprachlichen Dialogschnittstelle abgrenzt. Kernpunkt des GETESS-Suchdienstes ist die Erzeugung der Informationskondensate durch einen Sammelagenten, deren Speicherung in objektrelationalen Datenbanken sowie deren Abfrage aus den Datenbanken mittels einer speziellen Anfragesprache (IRQL).

Die genannten Aspekte zur Kopplung von Suchmaschinen- und Datenbank-Techniken werden in diesem Artikel überblicksartig beschrieben.

1 Einleitung

Suchmaschinen sind heutzutage ein wichtiges Hilfsmittel zur kostengünstigen Informationsrecherche im Internet. Die von den Suchmaschinen gelieferten Ergebnisse entsprechen jedoch nicht immer den Vorstellungen der Benutzer; sie sind häufig zu umfangreich, nicht zutreffend oder unvollständig ([SEW, Coma, Comb]). Die Ursache dafür liegt u.a. darin, daß sich die für die Recherche eingesetzten Werkzeuge größtenteils auf Schlüsselworte und sogenannte syntaktische Attribute von bereitgestellten Dokumenten (wie z.B. META-Tags) konzentrieren, ohne die eigentliche Bedeutung der Informationen zu beachten.

GETESS-Idee. Die Idee der GETESS-Suchmaschine besteht nun darin [SBB⁺99, SBB⁺99, DHK⁺99], die für die Suche verwendeten Dokumente bezüglich ihres Inhaltes detaillierter zu analysieren. Im Gegensatz zu den üblicherweise bei den Suchmaschinen verwendeten Schlüsselwort-Indexe wird bei GETESS ein sogenannter Abstract-Index benutzt. Zu den einzelnen Internet-Dokumenten werden Abstracts gebildet, die Informationskondensate auf Basis von linguistischem und Ontologie-Wissen¹ darstellen. Desweiteren wird eine natürlichsprachliche Dialogschnittstelle bereitgestellt, die es dem Benutzer erlaubt, seine Suchanfrage in natürlichsprachlichen Phrasen zu formulieren und im Dialog zu konkretisieren. Für die Beantwortung der Suchanfrage werden die natürlichsprachlichen Phrasen auf die in Datenbanken gespeicherten Abstracts abgebildet. Als Ergebnis wird eine Menge von Internet-Dokumenten geliefert, deren Inhalt dem Benutzer natürlichsprachlich auf Basis der Abstracts als Zusammenfassung präsentiert wird.

¹Unter einer Ontologie verstehen wir eine Menge von Begriffen, die über ihre inhaltliche Beziehung zueinander geordnet sind. Beispielsweise ist "Unterkunft" der Oberbegriff zu Hotel, Jugendherberge, Pension usw.

GETESS-Partner. GETESS ist ein vom BMBF gefördertes Projekt, das von den Partnern DFKI Saarbrücken (automatische Sprachverarbeitung), Universität Karlsruhe (Konstruktion von Ontologien), GECKO mbH Rostock (Dialoggestaltung) und der Universität Rostock (Suchmaschinen mit Datenbanktechnik) getragen wird. Ziel des Projektes ist es, das Know-how der verschiedenen Partner in einem Prototyp zu integrieren und als Synergieeffekt einen Suchdienst zu entwickeln, der als Basis für zukünftige Suchmaschinen dienen kann.

Fokus des Artikels. In diesem Artikel werden wir speziell die Kopplung der Suchmaschinen- mit der Datenbank-Technik beschreiben. Diese Kopplung betrifft

- das periodische Einsammeln und Analysieren von Internet-Dokumenten und die Erstellung eines Suchindex; d.h. das Füllen der Datenbanken mit Abstracts und
- die Nutzung des Suchindex durch die Suchmaschine - also die Abfrage der Abstracts aus den Datenbanken mittels der Anfragesprache IRQL.

Im Zentrum steht dabei eine Datenbank, die die Informationskondensate der Internet-Dokumente so speichert, daß ein schneller Zugriff auf die Analyse-Ergebnisse gewährt wird und die Vorteile der Datenbanktechnologie bzgl. der Anfragemöglichkeiten ausgenutzt werden können.

Die für GETESS realisierte Suchmaschinen-Datenbank-Kopplung ist im hohen Maße unabhängig von der konkreten Art des Informationskondensates und dem konkreten Dialogsystem. Die für GETESS generell geltenden Einschränkungen (Anwendungsbereich Tourismus und die Restriktionen bzgl. der natürlichen Sprache) sind im Kontext der Suchmaschinen-Datenbank-Kopplung nicht mehr relevant. Das bedeutet, daß die Möglichkeit besteht, eigene Informationskondensate sowie ein eigenes Dialogsystem zu definieren und die für GETESS gewählte Lösung zu adaptieren.

Aufbau des Artikels. Zunächst werden wir in Abschnitt 2 illustrieren, welchen Inhalt und welche Form die Informationskondensate (Abstracts) besitzen und welche Art der Suchanfragen wir auf Grundlage der Abstracts mit der Suchmaschine GETESS abdecken wollen. Anschließend werden wir die Erstellung von Abstracts in Abschnitt 3 betrachten² sowie die Abstractspeicherung (Abschnitt 4). In Abschnitt 5 werden wir die Anfragemöglichkeiten der Abstract-Datenbanken detaillierter vorstellen. Abschließend werden wir in Abschnitt 6 unsere Erläuterungen noch einmal zusammenfassen.

2 Abstracts - Informationskondensate als Index einer Internet-Suchmaschine

Informationen im Internet werden zumeist als Texte, Bilder, Videos u.ä. präsentiert. Die Extraktion von Inhalten aus diesen Präsentationsformen ist gegenwärtig Forschungsgegenstand [Kir98]. Ein Schwerpunkt im Zusammenhang mit Internet-Suchmaschinen liegt in der Analyse von natürlichsprachlichen Texten [TRE]. Im Gegensatz zur Stichwort-Extraktion wird dabei versucht,

- natürlichsprachliche Texte zu klassifizieren,
- wichtige, komplexe Informationen zu erkennen und
- die Klassifikation von Texten bzw. die erkannten Informationen innerhalb der Texte zu beschreiben und für die Beantwortung von Suchanfragen bereitzustellen.

Bei der Analyse von natürlichsprachlichen Texten sind dann sprachspezifische Probleme wie Synonymie oder Mehrdeutigkeit, aber auch Schreibfehler oder Abkürzungen zu beachten. Gleichzeitig sind Struktur-Spezifika, die im Zusammenhang mit dem Internet auftreten, einzuarbeiten. So kann im Internet

- eine unter gewissen Aspekten zusammengehörige Information auf mehreren Internet-Dokumenten verteilt sein (z.B. die Beschreibung eines Hotels mit den einzelnen Zimmertypen) oder
- ein und dieselbe Information mehrfach vorhanden sein (z.B. die Beschreibung eines Hotels in Deutsch und Englisch³),

²Details finden sich in [BDB⁺00] und [DG00]

³Natürlich ist diese Information nur ein und dieselbe unter z.B. dem Aspekt der Kosten für ein Doppelzimmer. In anderen Zusammenhängen muß diese Information nicht als 'gleich' betrachtet werden.

- eine Webpage Füllinformationen enthalten, die z.B. Werbe- oder Marketingcharakter haben und unter bestimmten Aspekten keinen Beitrag zur eigentlichen Information leisten (z.B. Werbebanner auf einer Homepage einer Suchmaschine).

Die genannten sprach-, domänen- und strukturspezifischen Aspekte sind bei der Extraktion von Inhalten aus Texten zu beachten.

2.1 Informationskondensat - Abstract

Zunächst werden wir den Begriff des Abstracts genauer definieren.

Ein **Abstract** ist eine Menge von instanziierten Templates bzw. domänenspezifischer Konzeptbeschreibungen.

Im Kontext des Projektes GETESS bedeutet das, daß aus einem gegebenen Dokument eine Zusammenstellung von domänenspezifischen Informationen des jeweiligen Inhalts des Dokumentes generiert werden.

Ein Template beschreibt in diesem Zusammenhang ein Konzept, welches, wenn es im Dokument vorkommt, mit den Werten aus dem Inhalt des Dokumentes instanziiert wird und in die Menge der Abstract-Beschreibungen aufgenommen wird.

Betrachten wir das folgende Beispiel eines HTML-Dokumentes (Abbildung 1). Das Beispiel stellt eine Hotel-Beschreibung dar und ist zu finden unter <http://www.all-in-all.com/1293.htm>.

Hotel Mecklenburger Hof
D-19288 Ludwigslust
Lindenstrasse 40 - 44
Tel. 03874/ 410-0
Fax: 03874/ 410-100

Das Hotel Mecklenburger Hof liegt verkehrsgünstig in der Innenstadt nahe dem Ludwigsluster Schloss mit seinem Schlosspark und der Schlosskirche.

Hausbeschreibung:
Komfort auf hohem Niveau in 37 Zimmern mit 72 Betten. Das Haus wurde mit viel Liebe zum Detail restauriert und bietet seinen Gästen stilvolle gemütliche Gastlichkeit, verbunden mit einem sehr aufmerksamen Service, einer ausgezeichneten Küche und exzellenten Frühstücksbuffet.
Ein gut sortierter Weinkeller gehört genauso dazu, wie der zweimal im Jahr stattfindende Operettenball und weitere interessante Veranstaltungen.
Kostenfreie Parkplätze stehen Ihnen in der Tiefgarage zur Verfügung.

Preise:
Einzelzimmer: von DM 88,00 bis DM 120,00
Doppelzimmer: von DM 130,00 bis DM 185,00

Kreditkarten: EC, Visa, Diners

Abbildung 1: Beispiel eines HTML-Dokumentes

In Abbildung 2 ist ein Ausschnitt aus dem zum Beispiel-Dokument gehörenden Abstract dargestellt.

2.2 Anfragen auf Basis von Abstracts

Anfragen der GETESS-Suchmaschine werden nicht nur auf Basis von Stichworten der Dokumente beantwortet, sondern in erster Linie werden die Abstracts als Grundlage benutzt.

```

<Getess>
  <HOTEL>
    <HOTEL_1> hotel Mecklenburger Hof </HOTEL_1>
  </HOTEL>
  <ORT>
    <ORT_1> Ludwigslust </ORT_1>
    <ORT_2> Ludwigslust (linde-PL-N-strasse) 40-44 </ORT_2>
  </ORT>
  <GEBIET>
    <GEBIET_1> Mecklenburg </GEBIET_1>
  </GEBIET>
  <ZIMMERAUSSTATTUNG>
    <ZIMMERAUSSTATTUNG_1> telefon </ZIMMERAUSSTATTUNG_1>
    <ZIMMERAUSSTATTUNG_2> bett </ZIMMERAUSSTATTUNG_2>
  </ZIMMERAUSSTATTUNG>
  <ZIMMER>
    <ZIMMER_1> zimmer </ZIMMER_1>
    <ZIMMER_2> einzelzimmer </ZIMMER_2>
    <ZIMMER_3> doppelzimmer </ZIMMER_3>
  </ZIMMER>
  <PREIS>
    <PREIS_1> ez-min 88 </PREIS_1>
    <PREIS_2> ez-max 120 </PREIS_2>
    <PREIS_3> dz-min 130 </PREIS_3>
    <PREIS_4> dz-min 185 </PREIS_4>
  </PREIS>
  <PREIS_WAEHRUNG>
    <PREIS_WAEHRUNG_1> dm </PREIS_WAEHRUNG_1>
  </PREIS_WAEHRUNG>
  <UNTERKUNFT_ZAHLUNGSMOEGELICHKEIT>
    <UNTERKUNFT_ZAHLUNGSMOEGELICHKEIT_1> kredit-karte </UNTERKUNFT_ZAHLUNGSMOEGELICHKEIT_1>
  </UNTERKUNFT_ZAHLUNGSMOEGELICHKEIT>
  <BETRIEB_ZAHLUNGSMOEGELICHKEIT>
    <BETRIEB_ZAHLUNGSMOEGELICHKEIT_1> ec </BETRIEB_ZAHLUNGSMOEGELICHKEIT_1>
    <BETRIEB_ZAHLUNGSMOEGELICHKEIT_2> visa </BETRIEB_ZAHLUNGSMOEGELICHKEIT_2>
    <BETRIEB_ZAHLUNGSMOEGELICHKEIT_3> diners </BETRIEB_ZAHLUNGSMOEGELICHKEIT_3>
  </BETRIEB_ZAHLUNGSMOEGELICHKEIT>
  <GASTRONOMIE>
    <GASTRONOMIE_1> weinkeller (sortier) </GASTRONOMIE_1>
  </GASTRONOMIE>
</Getess>

```

Abbildung 2: Ausschnitt aus dem gebildeten Abstract

Im folgenden werden wir anhand von zwei Beispiel-Anfragen erläutern, inwiefern Abstracts bei der Beantwortung genutzt werden können und inwiefern die Beantwortung via Abstracts qualitativ bessere, d.h. exaktere und benutzer-adäquatere Ergebnisse liefert.

Anfrage: Ich möchte ein Doppelzimmer unter 150 DM.

Eine Suchmaschine, die diese Anfrage adäquat beantworten möchte, muß innerhalb der Textbeschreibungen die richtigen Teil-Strings als Preise erkennen und diese entsprechend interpretieren. Die GETESS-Suchmaschine erkennt die verschiedenen Schreib- und Definitionsweisen von Preisen und bildet die Preisstrukturen auf ein abstraktes Konzept im Abstract ab. Bei der Beantwortung der Anfrage spielen dann die verschiedenen Schreibweisen keine Rolle mehr, denn der Preisvergleich (d.h. < 150 DM) wird über den Wert des abstrakten Konzeptes realisiert.

In ähnlicher Weise werden auch andere Konzepte, z.B. Kredit-Karten oder Öffnungszeiten, behandelt.

Anfrage: Ich suche ein Doppelzimmer mit Swimmingpool.

Der Benutzer, der diese Anfrage stellt, sucht mit hoher Wahrscheinlichkeit eine Unterkunft (z.B. ein Hotel, eine Ferienwohnung, etc.), die Doppelzimmer hat und mit einem Swimmingpool ausgestattet ist. Suchmaschinen, die auf Stichwort-Basis agieren, würden nur solche Dokumente finden, die die einzelnen Begriffe 'Doppelzimmer' und 'Swimmingpool' in einem Dokument vereinen bzw. nur einen der beiden Begriffe besitzen. Beschreibungen von Unterkünften, die mehrere WWW-Dokumente umfassen und bei

denen die beiden Suchbegriffe auf einzelne Dokumente verteilt sind, die jedoch inhaltlich zur selben Unterkunft gehören, würden nicht als Suchergebnis gefunden. Durch die Nutzung von Abstracts ist es möglich, verteilt auf mehreren Dokumenten vorliegende Informationen zusammenzufassen und als inhaltlich zusammengehörend in die Anfragebeantwortung einfließen zu lassen.

Die Präsentation des Suchergebnisses kann dann so gestaltet werden, daß der im Sinne des Benutzers relevante Schwerpunkt - hier der Name der Unterkunft - für die Ergebnisbeschreibung genutzt wird, da die entsprechenden Abstract diese Informationen enthalten.

Ergebnisse des GETESS-Projektes: Erweiterte Anfragemächtigkeit. Die beiden Beispielen illustrieren, daß bislang folgende allgemeine Ergebnisse bzgl. der Anfragemächtigkeit im Projekt GETESS erreicht wurden:

- Ein höherer Abstraktionsgrad kann in den Anfragen genutzt werden; d.h. abstrakte (z.B. Unterkunft) und konkrete (z.B. Hotel) Konzepte sowie die Beziehung zwischen diesen Konzepten kann bei der Beantwortung ausgenutzt werden.
- Eine Abstraktion hinsichtlich sprachlicher Eigenheit in Bezug auf den Inhalt wird erreicht; d.h. durch die Nutzung von Konzepten und Repräsentationen von Konzepten werden auch Suchergebnisse mit adäquatem Inhalt aber sprachlicher inadäquater Repräsentation geliefert.
- Die Auswahl von Suchergebnissen aus der Datenbank auf Basis der Abstracts mittels Datenbank-Anfragesprache ermöglicht es, auch einzelne Dokumente inhaltlich in einen Zusammenhang zu stellen. Damit können Informationen, die auf einzelne WWW-Dokumente verteilt wurden, über Konzepte und letztendlich durch Datenbank-Attribute zusammengefaßt werden.

Die Erstellung von Abstracts wird durch den GETESS-Sammelagenten realisiert, der im folgenden näher beschrieben wird.

3 Analyse von WWW-Dokumenten - Erstellung von Abstracts

Der GETESS-Sammelagent (Gatherer) hat die Aufgabe, Internet-Domänen zu durchlaufen und Daten aus den Quellen des Internets einzusammeln, diese Daten aufzubereiten und Abstracts zu erstellen.

3.1 Harvest - Die Basis des GETESS-Gatherers

Als Grundlage für den GETESS-Gatherer wurde der Harvest-Gatherer [Har] gewählt. Harvest⁴ bietet ein Internetsuchsystem mit einer verteilten Architektur. Die Funktionalität des Harvest-Gatherers umfaßt die Extraktion von Stichwörtern aus verschiedenen Text-Formaten (HTML, PS, etc.) und die Extraktion von HTML-Meta-Tags sowie die Speicherung der Stichworte und Meta-Tags in einem Stichwort-Index. Ebenfalls im Funktionsumfang enthalten sind Update-Mechanismen, die aufgrund von definierten Verfallsdaten eine erneute Analyse durch den Sammelagenten definieren.

3.1.1 Funktionsweise des Sammelagenten

Der Ausgangspunkt für den Gatherer stellen eine oder mehrere gegebene Internet-URL's dar, bei denen der Sammelagent einen Suchdurchlauf startet. Der Gatherer geht davon aus, daß eine Daten-Datei existiert, in der die gesammelten Daten des vorigen Gathering-Laufs in einem bestimmten Format abgelegt sind. Existiert eine solche Daten-Datei nicht, sind keine Daten vorhanden, und die einzige Aufgabe des Gatherers besteht darin, neue Daten einzusammeln und zu speichern.

Zunächst wird das Einsammeln der Dokumente vorbereitet, indem die gegebenen URL's dahingehend geprüft werden, ob sie neu oder schon in der Daten-Datei vorkommen. Bei den in der Datensammlung vorkommenden Daten wird das Refresh-Time-Attribut geprüft. Wenn die Zeit seit dem letzten Update abgelaufen ist, wird das Dokument neu gelesen. Ein weiterer Schritt ist das Entfernen von Informationen aus der Daten-Datei, deren Time-To-Live-Zeit seit dem letzten Update abgelaufen ist. Davon betroffen

⁴'Currently, there are hundreds of Harvest applications on the Web (for example, the CIA, the NASA, the US National Academy of Science, and the US Government Printing Office), as this software is on the public domain.' [BYRN99]

sind die Dokumente, deren Update seit längerem nicht mehr vollzogen werden konnte, da vermutlich das Dokument im Internet nicht mehr verfügbar ist.

Im zweiten Schritt wird das Essence-System initiiert. Im Essence-System werden die Dokumente eingesammelt, der Typ bestimmt, der jeweilige Summarizer (ein Programm zum Parsen eines Dokumentes eines bestimmten Typs) gestartet und die Daten zwischengespeichert.

Im dritten Schritt wird die Datenbasis neu aufgebaut. Dazu wird aus der bestehenden Daten-Datei, aus den neu gesammelten Daten und aus den geänderten Daten eine neue Daten-Datei durch Verschmelzen gewonnen.

Als letztes wird der Sammel-Dämon gestartet, der ständig verfügbar ist und die Daten für den Zugriff durch das Abfragesystem (Broker) bereitstellt.

3.1.2 Das Essence-System

Das Essence-System ist für die Informationsextraktion zuständig. Es ist nötig, diesen internen Teil von Harvest zu betrachten, da an dieser Stelle das Parsen der Dokumente und die Erstellung der zusätzlichen Attribute durchgeführt wird. Man wird sehen, daß diese Informationen für die Integration der Abstracts von wesentlicher Bedeutung sind. Im folgenden wird gezeigt, wie das Essence-System intern abläuft.

Zunächst ist es sinnvoll, zwischen neu zu sammelnden und aus der Datenbasis zu erneuernde Dokumente zu unterscheiden. Bei den zu erneuernden Dokumenten ist ein erneutes Erstellen von Attributen, die das Dokument beschreiben, nur dann erforderlich, wenn sich das Dokument geändert hat. Dazu wird ein spezielles Attribut verwendet, in dem eine Prüfsumme des Dokumentes gespeichert wird. Der Vergleich, ob sich ein Dokument verändert hat, erfolgt über den Vergleich von Prüfsummen.

Bei neuen Dokumenten und zu erneuernden, veränderten Dokumenten wird nun durch den Type-Recognition-Schritt der Typ bestimmt. Je nach Typ wird dann eine eventuelle Vorverarbeitung⁵ nötig und der passende Summarizer ausgewählt. Bevor die Dokumente zum Summarizer geschickt werden, werden einige Metadaten der Dokumente bestimmt und damit Dokumente herausselektiert, die nicht unbedingt durch einen Summarizer geparkt werden müssen (z.B. aufgrund ihres Typs).

Für das **Summarizing** existiert für jeden, zu parsenden Typ ein externes Programm. Mittels dieser Summarizer-Programme werden die gesammelten Dokumente geparkt. Dabei wird eine Umsetzung von Wörtern und strukturellen Informationen — wenn vorhanden — zu Attribut-Wertpaaren vorgenommen. Neben den global definierten Attributen existieren dokumentspezifische Attribute. Die dokumentspezifischen Attribute werden aus herausgefilterten Worten und strukturellen Informationen gebildet. Dazu gibt es intern eine Art Stopwortliste, die allerdings nicht konfigurierbar ist. Die hier generierten Datensätze werden zusammen mit den Metadaten später beim Broker indexiert und können angefragt werden.

Nach dem Summarizing gibt es noch ein sogenanntes Postsummarizing. In dieser Phase wird der komplette Datensatz, also Metadaten und dokumentinterne Daten, für die Daten-Datei zusammengestellt. An dieser Stelle ist es möglich extern den Datensatz noch zu beeinflussen.

Am Ende dieser Prozesse wurde für jedes Dokument ein Datensatz erstellt, welcher vom Gather-Dämon für Abfragezwecke bereit gestellt wird.

3.2 Integration der Abstract-Analyse in den GETESS-Gatherer

Die Erstellung der Abstracts erfolgt durch das SMES-System. SMES⁶ und führt eine flache natürlichsprachige Analyse der Texte durch.

Neben der Extraktion von Stichworten im GETESS-Gatherer war es das Ziel, auch die Abstracts-Bildung für jedes eingesammelte Dokument im Gathering Prozess auszuführen. Dazu wurden verschiedene konzeptionelle Lösungen hinsichtlich ihrer Vor- und Nachteile betrachtet [Bru99]. Im folgenden wird die realisierte Lösung beschrieben, die in Abbildung 3 graphisch dargestellt wird.

Unter den betrachteten Realisierungsmöglichkeiten ist die Integration von SMES und Harvest beim Summarizing innerhalb des Essence-Systems des Gatherers die einfachste und schnellste Methode. Desweiteren bietet die gewählte Lösung einen hohen Grad an Modularität und Erweiterbarkeit. Durch die SMES-Client-Server-Anbindung und die damit gewährleistete hohe Unabhängigkeit der beiden Systeme ist die Weiterentwicklung des SMES-Systems und des Sammelagenten ohne größere Probleme

⁵Vorverarbeitung — wie z.B. Entpacken gepackter Dokumente oder Zerlegen von sogenannten Multi-Files.

⁶SMES - a fast and robust information extraction core system for real-world German text processing

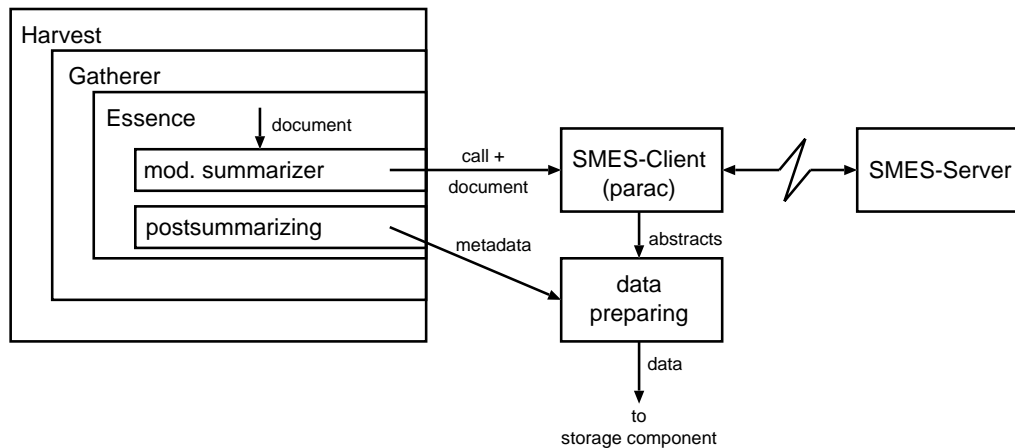


Abbildung 3: Integration von SMES in den GETESS-Gatherer

möglich. Als wesentlichster Nachteil der gewählten Integrationslösung ist jedoch zu nennen, daß es weniger Flexibilität und Optimierungsmöglichkeiten hinsichtlich der Parallelisierung des Such- und des Analyse-Prozesses gibt. Alle Vor- und Nachteile betrachtend, stellt die gewählte Realisierungsmöglichkeit für den GETESS-Prototyp eine vom Aufwand/Nutzen sehr günstige Gathering-Parsing-Integration dar. Deshalb wurde diese Realisierung prototypisch implementiert.

Für die Realisierung dieser Integrationsmöglichkeit wurde ein Webserver, ein Harvest-System und ein SMES-Client installiert und konfiguriert.

Im vorhergehenden Abschnitt ist das Summarizing als ein Schritt innerhalb des Gatherers beschrieben, bei dem mittels Summarizer, ein Datensatz erstellt wird. Zu diesem Zeitpunkt ist eine vollständige Kopie des Originaldokumentes gespeichert. Hauptaufgabe ist es nun, den SMES-Client zu starten und diese Kopie zu übergeben. Da bei GETESS momentan nur HTML-Dokumente betrachtet werden, wird an dieser Stelle nur der HTML-Summarizer betrachtet.

Der bei Harvest bevorzugte HTML-Summarizer ist ein SGML-Parser (SGML ist eine Metasprache zur Dokumentbeschreibung, HTML ist eine Instanz von SGML), der unter Angabe des Dokumenttyps das Dokument strukturell in bestimmte Bestandteile zerlegt. Aufgerufen wird beim Gathering dann eine Datei, deren Namen sich aus Dokumenttyp und dem Suffix "sum" zusammensetzt. Die Datei "HTML.sum" ist also das zuständige Summarizing-Programm. Diesem Programm wird der Name Dokumentkopie übergeben. Das Programm ruft den SGML-Summarizer auf und übergibt den Dokumenttyp (HTML) und den Namen plus Pfad der Dokumentkopie. Die Erstellung des Datensatzes übernimmt dann komplett der aufgerufene SGML-Summarizer. Nach dessen Beendigung wird auch der HTML-Summarizer beendet.

Der HTML-Summarizer wurde nun so konzipiert, daß er den SMES-Client aufruft und das Dokument übergibt. Der SMES-Client wiederum liefert ein Abstract im XML-Format, welches im Summerizer an die Datenbank zur Speicherung weitergegeben wird.

Der GETESS-Gatherer mit integrierter Abstract-Funktionalität sammelt derzeit wöchentlich Daten, bildet zu diesen Daten Abstracts und stellt diese Abstracts in einer Datenbank bereit.

3.3 Aktuelle und geplante Erweiterungen des GETESS-Gatherers

Erste Testversuche mit der GETESS-Suchmaschine verliefen erfolgreich, zeigten jedoch auch einige notwendige Erweiterungen. Im folgenden werden wir Erweiterungsideen hinsichtlich der Sprachklassifikation und der Kontextanalyse beschreiben.

Sprachklassifikation. Eine Spezifik des Internets ist es, daß Dokumente mit dem gleichen Inhalt in mehreren Sprachen vorhanden sind. Für eine Internet-Suchmaschine ist es wichtig zu erkennen,

- in welcher natürlichen Sprache ein Dokumenttext verfaßt wurde und
- welche Dokumente den gleichen Inhalt besitzen, aber diesen in verschiedenen Sprachen repräsentieren.

Die Erkennung von natürlichen Sprachen und das Zusammenfassen von Dokumenten wird innerhalb des GETESS-Gatherers vorgenommen. Dazu wurde ein heuristisches Vorgehen zur Sprachklassifikation erarbeitet [DG00], das der Abstract-Analyse vorgeschaltet ist und erkennt, in welcher natürlichen Sprache⁷ der Dokumenttext verfaßt wurde.

Das Ziel der Sprachklassifikation ist es, zu einem der Abstract-Analyse die gewählte natürliche Dokument-Sprache zu übergeben und damit den zeitlichen Aufwand der linguistischen Analyse und Abstract-Bildung erheblich zu verringern. Zum anderen ist es Ziel, dem Benutzer auf eine Suchanfrage hin Dokumente in seiner gewählten Sprache in der Ergebnisliste 'weiter oben' zu präsentieren und die anderssprachigen Dokumente mit dem gleichen Inhalt erst an zweitrangiger Stelle.

Kontextanalyse. Im direkten Zusammenhang mit dem Zusammenfassen von Dokumenten in verschiedenen Sprachen steht die Kontextanalyse. Das Ziel der Kontextanalyse [BDB⁺00] ist es, inhaltlich zusammengehörige Dokumente zu Dokumentklassen zusammenzufassen und die Abstract-Bildung auf die gesamte Klasse anzuwenden (und nicht mehr nur auf ein Dokument der Klasse). Ausgangspunkt dafür ist die Tatsache, daß Dokumente häufig - wie schon im Abschnitt zur Abstract-Bildung beschrieben - inhaltlich zusammen gehören und dem Benutzer im Suchergebnis auch als ein komplexes Ergebnis präsentiert werden sollen.

Dazu werden derzeit Mechanismen entwickelt, um unter bestimmten Gegebenheiten Klassen von Dokumenten zu definieren, um aus Teilen von Dokumenten einer Klasse im Kontext Abstracts zu bilden und diese komplexen Abstracts in der Datenbank zu speichern.

4 Speicherung von Abstracts in Datenbanken

Innerhalb des GETESS-Projektes sollen für Dokumente im WWW die wesentlichen Inhalte bestimmt und in Form von sogenannten Abstracts gespeichert werden. Dabei beschreibt eine umfangreiche Ontologie die Struktur der Daten in den abzuleitenden Abstracts. Bei der Abstractbildung werden konkrete Werte für die modellierten Strukturen ermittelt. Diese Informationen werden im GETESS-Projekt u.a. in objektrelationalen Datenbanken gespeichert. Die Struktur der Datenbank durch die Ontologie bestimmt, die Tupel der Datenbank resultieren aus den Abstracts. Abbildung 4 zeigt diese Zusammenhänge.

4.1 Drei Varianten zur Speicherung von Informationen

Die Speicherung von objektrelationalen Datenbanken ist nur eine mögliche Realisierung der Speicherung von Abstract-Informationen. Für semistrukturierte Daten ist es günstiger, Abstract-Daten in Datenbanken zu speichern, die bestimmte Informationen als XML-Datentypen zusammenfassen. Eine dritte Variante der Speicherung ist die Verwendung von XML-Dokumenten, um eine Speicherung unabhängig von einem Datenbanksystem zu realisieren. Abbildung 5 stellt diese drei Methoden zur Speicherung von Abstractinformationen dar.

Die Variante 2 in dieser Abbildung ist der allgemeinste Fall. Variante 1 ist ein Spezialfall von Variante 2, bei dem keine XML-Datentypen auftreten, also alle Informationen strukturiert dargestellt werden. Variante 3 ist in gewisser Hinsicht auch ein Spezialfall, bei dem quasi nur ein Attribut auftritt, liegt aber außerhalb eines Datenbanksystems und ist somit von der Umsetzung gesondert zu betrachten.

Im folgenden werden die einzelnen Teilaufgaben erläutert, die zu lösen sind, um die einzelnen Speichervarianten umzusetzen. Dabei werden auch die Vor- und Nachteile der einzelnen Varianten genannt.

4.2 Speicherung in objektrelationalen Datenbanken

Vor- und Nachteile. Bei der vollständigen Abbildung der Abstracts in objektrelationale Datenbanken tritt häufig der Fall auf, daß dabei sehr große Datenbank-Schemata entstehen und die resultierenden

⁷Der Ansatz wurde bisher für Deutsch, Englisch und Französisch getestet und wird derzeit für Japanisch konzipiert.

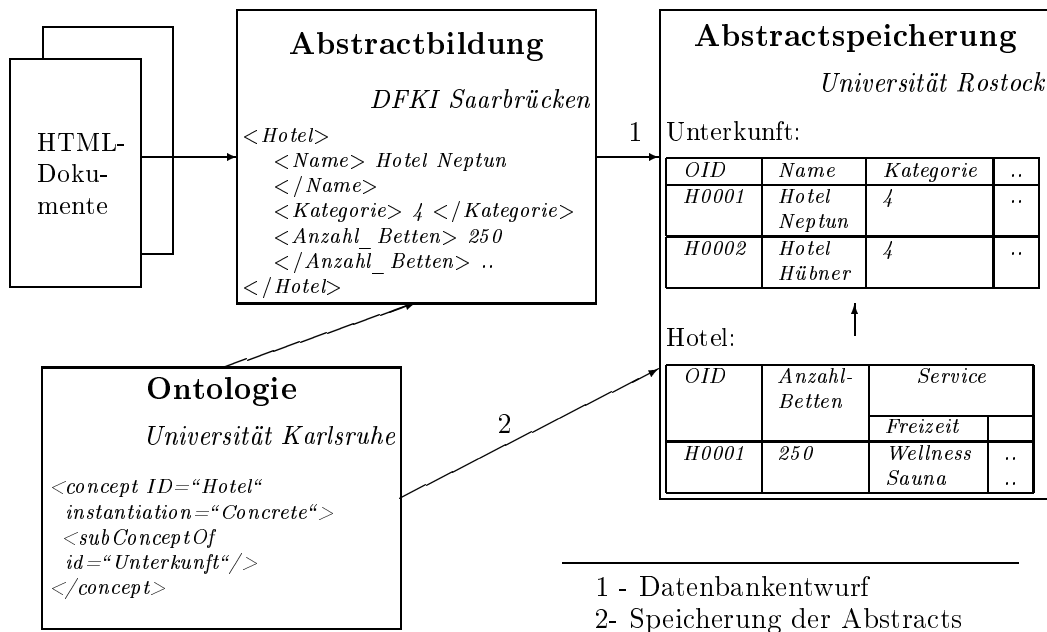


Abbildung 4: Zusammenwirken von Ontologie, Abstractbildung und Abstractspeicherung

Datenbanken nur schwach besetzt sind. Ursache dafür ist, daß die analysierten Webseiten die klassischen Merkmale semistrukturierter Daten ausweisen, diese widerspiegeln sich auch in den Abstracts. Der große Vorteil dieser Variante ist jedoch die Verwendung von Datenbank-Anfragesprachen.

Datenbank-Entwurf. Die Ontologieinformationen lassen sich in einem objektrelationalen Datenbankentwurf umsetzen.

Die Ontologie (in KIF, F-Logic oder XML) wird geparkt und es wird ein Baum aufgebaut, der die Ontologieinformationen enthält. Die SQL-Anweisungen CREATE TYPE/TABLE werden entsprechend generiert, die Reihenfolge dieser Anweisungen ist nicht beliebig, deshalb wird der Baum entsprechend ausgelesen.

Beim Erstellen der Datenbanken werden die in der Ontologie dargestellten Informationen über is-a-Hierarchien in analoger Weise in die Datenbank übernommen. Informationen über Attribute und Typen werden zu den Relationen hinzugefügt.

Alle Umwandlungsschritte werden in Metainformationen dokumentiert. Die Metainformationen haben zwei Bedeutungen. Zum einen sind sie notwendig, um die Umsetzung der Ontologie in die entsprechenden Datenbankstrukturen nachvollziehen zu können. Dieses wird beim Einfügen von konkreten Abstract-Informationen in die Datenbank benötigt. Zum anderen lassen sich die Ontologie-Informationen auf dieser Ebene anfragen.

Speicherung der Daten. Bei der Speicherung der Abstracts in objektrelationalen Datenbanken werden diese geparkt. Es erfolgt eine Umwandlung in INSERT-Operationen in die Datenbanken. Die dazu notwendigen Informationen werden als Metainformationen der Abstracts gehalten.

Realisierung von Anfragen. Anfragen an objektrelationale Datenbanken können über SQL realisiert werden, hier sind keine zusätzlichen Arbeiten notwendig.

4.3 Umsetzung der Ontologie in hybride Datenbanken

Unter hybriden Datenbanken verstehen wir objektrelationale Datenbanken, in denen Informationen mit dem Datentyp XML gespeichert werden können.

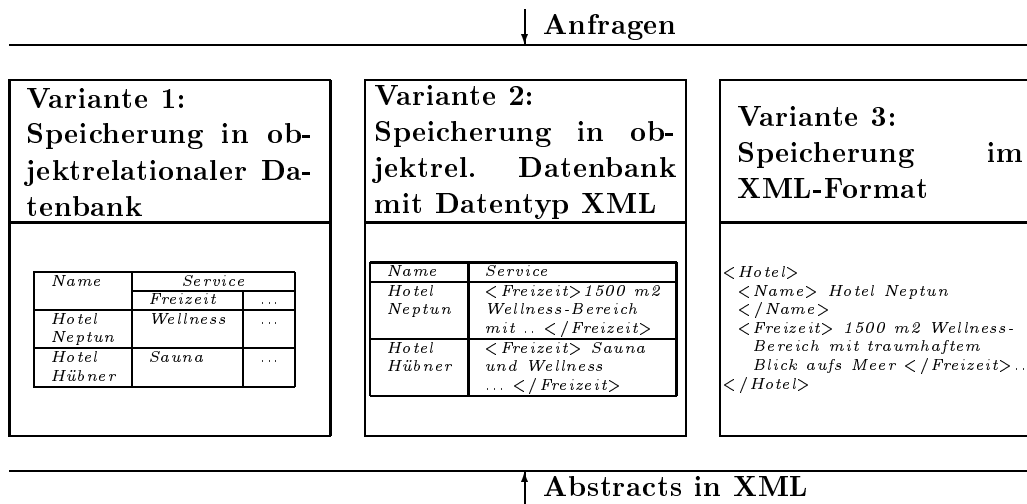


Abbildung 5: Drei Methoden zur Speicherung von Abstract-Informationen

Vor- und Nachteile. Bei dieser Varianten werden die häufig in den Abstracts vorhandenen und die häufig angefragten Informationen strukturiert in Datenbanken abgebildet, für diese hat man dann die Möglichkeit, Datenbank-Anfragefunktionalität einzusetzen. Durch die Zusammenfassung der anderen Informationen aus den Abstracts zu XML-Datentypen erreicht man eine kompaktere Speicherung. Man benötigt hier jedoch eigene Tools zur Anfrage. Zunehmend werden kommerzielle Datenbanksysteme diese jedoch anbieten.

Datenbank-Entwurf. Die Entscheidung, welche Attribute in der Datenbank strukturiert dargestellt werden und welche als XML-Datentypen zusammengefaßt werden, wird im wesentlichen von zwei Merkmalen bestimmt:

- konkreten Daten
- bekannten Anfragen

Attribute, für die in vielen Dokumenten Werte gefunden wurden und die häufig angefragt werden, sollen dabei direkt in die Datenbank aufgenommen werden, während Attribute, für die nur selten Werte gefunden werden und die selten oder nie in Anfragen vorkommen, zusammengefaßt und als XML-Datentypen gespeichert werden. Der Algorithmus zur Bestimmung dieses Entwurfes wird in [KM00] dargestellt.

Speicherung der Daten. Beim hybriden Ansatz werden einige Attribute direkt gespeichert, andere bleiben als XML-Typen zusammen. Die Informationen darüber, welche das jeweils betrifft, findet sich in den Metainformationen. Die INSERT-Anweisungen werden dementsprechend generiert.

Realisierung von Anfragen. In [Por99] wurde ein Prototyp konzipiert, der aufbauend auf einem Text Extender eine XML-Erweiterung für objektrelationale Datenbanken (Informix und DB2) vorschlägt und für DB2 implementiert.

Der Vorteil dieses Zuganges ist, daß der Text Extender von DB2 bzw. das Excalibur Text Data Blade unter Informix als Basis verwendet werden kann, um solche Funktionen wie Synonymsuche, Wortstammreduktion, Fuzzy-Suche, usw. nutzen zu können. Im Text Extender kann der Index nur auf den gesamten XML-Dokumenten gebildet werden. Die XML-Dokumente können also zunächst nur als Volltext aufgefaßt werden, auf diesem wird eine Vorselektion für Anfragen durchgeführt. Es ist nach der Verwendung des TextExtenders ein zweiter Durchlauf durch das Dokument notwendig, bei dem die XML-Struktur analysiert wird, um Anfragen zu beantworten.

In zunehmendem Maße werden sich kommerzielle Tools für diese Aufgabe eignen.

4.4 Umsetzung der Ontologie in DTDs.

Vor- und Nachteile. Bei dieser Variante werden die Abstracts als XML-Dokumente gespeichert. Die Speicherung selbst ist damit relativ einfach realisierbar, die Probleme liegen hier auf der effizienten Realisierung von Anfragen auf den XML-Strukturen.

DTD-Entwicklung. Es läßt sich ein Algorithmus angeben, der aus den Ontologieinformationen DTDs ableitet.

Speicherung der Daten. Bei der Speicherung von XML-Dokumenten muß überprüft werden, ob die Eigenschaften Wohlgeformtheit und Konformität erfüllt sind und entsprechende Umwandlungen zur Anpassung der XML-Dokumente an die jeweilige DTD vorgenommen werden.

Anfragen an XML-Strukturen. Im vorherigen Abschnitt wurde beschrieben, wie XML-Strukturen innerhalb eines Datenbanksystems angefragt werden können. Die gleichen Funktionen müssen auch außerhalb eines Datenbanksystems realisiert werden. Auch hierfür existieren kommerzielle Tools, die eingesetzt werden können. Allerdings ergeben sich gegenwärtig noch Probleme, weil nicht absehbar ist, welche XML-Anfragesprache zukünftig eingesetzt werden wird.

Die Anfragen an die drei Varianten zur Speicherung von Abstractinformationen unterscheiden sich stark, deshalb wird im Projekt eine einheitliche Sprache eingesetzt, die dann die Umsetzung auf diese Varianten realisiert. Diese wird im Abschnitt 5 vorgestellt.

4.5 Unabhängigkeit der Abstract-Speicherung von der gewählten Domäne

Alle drei Varianten der Abstractspeicherung sind unabhängig von der konkreten Domäne. Da die Verfahren zum Entwurf und zur Speicherung der Informationen jeweils als automatische Verfahren realisiert wurden, kann die Datenbanklösung für jedes beliebige Anwendungsgebiet verwendet werden. Notwendig ist dazu nur eine eine Ontologie, die die entsprechende Domäne beschreibt.

5 Abfrage von Abstracts aus Datenbanken

Die Extraktion der zur Beantwortung einer Nutzeranfrage erforderlichen Daten aus der Abstract-Datenbank wird durch die im Rahmen des Projektes entwickelte Anfragesprache IRQL [HP00] realisiert. IRQL ist dabei als interne Anfragemöglichkeit zu betrachten: die Generierung von IRQL-Anfragen aufgrund einer Nutzeranfrage oder eines Dialogablaufs wird durch die Dialogkomponente vorgenommen. IRQL ist damit völlig unabhängig von einer bestimmten Ontologie bzw. Domäne allgemein.

Die Zielsetzung beim Entwurf von IRQL⁸ war die Integration von Anfragemöglichkeiten an strukturierte und semi-strukturierte Daten sowie von Information Retrieval Techniken. Damit können die nach der Abbildung der analysierten Daten auf die Strukturen der Abstract-Datenbank (siehe Abschnitt 4) gespeicherten Daten in einer Weise angefragt werden, die durch existierenden Anfragesprachen nicht möglich ist.

Durch die Integration der Anfragemöglichkeiten aus den drei Bereichen

1. Datenbankanfragesprachen,
2. Anfragesprachen für semi-strukturierte Daten und
3. Information Retrieval

ist IRQL in der Lage, die jeweiligen Eigenschaften

- Attributierung,
- Restrukturierung einschließlich Verknüpfungen,
- implizite Typumwandlung bei Operationen auf inkompatiblen Typen,
- Unterstützung variabler bzw. heterogener Schemata,

⁸Information Retrieval Query Language

- inhaltsbasierte Suche,
- Ranking und
- Relevance Feedback

orthogonal zu kombinieren. Die Implementierung von IRQL erfolgt auf existierenden Lösungen, wie etwa objekt-relationalen oder relationalen Datenbanksystemen oder Volltext-Systemen wie Fulcrum.

Nach diesem Überblick über die Zielsetzung beschreiben wir nun das zugrundeliegende Datenmodell.

5.1 Datenmodell

Unser Datenmodell stellt eine Erweiterung eines objekt-relationalen Datenmodells dar. Zu den unterstützten Datentypen zählen die üblichen atomaren Datentypen `integer`, `float`, `boolean` und `string`; die zusammengesetzten Typen `set`, `bag`, `list`, `array` sowie der Strukturkonstruktor `struct`. Es existiert ein Typ `named`, der beispielsweise zur Modellierung von XML-Daten verwendet wird und die Anwendung typ-spezifischer Operationen auf die jeweiligen Daten⁹ erlaubt. Die Modellierung heterogener, semi-strukturierter Daten erlaubt der Datentyp `doc`, der dem `struct`-Konstruktor ähnelt, jedoch keinerlei Anforderungen (wie etwa Homogenität) an die enthaltenen Attribut-Wert-Paare stellt. Außerdem führen Operationen auf Daten dieses Typs niemals zu einem Typfehler.

Während die Modellierung heterogener, semi-strukturierter Daten nach der beschriebenen Weise bereits in anderen Sprachen existiert, ist die durch unser Datenmodell unterstützte Abstraktion von Attributwerten neuartig. Abstraktion von Attributwerten bedeutet hier, daß eine Menge von Attributen durch einen einzigen Bezeichner angesprochen werden kann. Dieser Bezeichner verhält sich dann wie ein "normales" Attribut und kann in weiteren Abstraktionsebenen verwendet werden. Als Beispiel verweisen wir auf Abbildung 6. Hier ist `complete_content` eine Abstraktion der Attribute `meta_data` und `text`, die

document	source	complete_content						
		metadata			text			
article		authors	title	year	abstract	sections	appendix	references
	http://e-lib.informatik.uni-rostock.de/2000/DBIS/IRQL/irql.ps	Andreas Heuer Denny Priebe	Integrating a Query Language for Structured and Semi- ...	2000	In this paper we describe the basic ideas and concepts behind the ...	During the last years the WWW became generally accepted as a medium to publish various kinds of information (documents). ...		[1] Serge Abiteboul. Querying Semi-Structured Data. In Foto N. Afrati and Phokion Kolaitis, editors, Database Theory - ICDT'97

Abbildung 6: Abstraktion von Attributen

beide wieder Abstraktionen anderer Attribute sind. In einer Anwendung kann dann `complete_content` beispielsweise als ein atomarer Typ behandelt und etwa für die Volltextsuche verwendet werden.

5.2 Sprache

Als Ausgangspunkt für IRQL verwenden wir den kürzlich verabschiedeten Standard SQL99 [ANS99a, ANS99b]. Erweiterungen umfassen die bereits aus der Vorstellung des Datenmodells ersichtliche veränderte Definition der unterstützten Operationen sowohl auf strukturierten als auch auf semi-strukturierten Daten als auch auf Ebene der Sprache "sichtbare", syntaktische Erweiterungen. Dazu zählen

- reguläre Pfadausdrücke und Pfadvariablen¹⁰, die die Navigation und Anfragen bzgl. heterogenen Daten erlauben.
- die Verwaltung und Manipulation von Extensionen. In IRQL existiert eine vordefinierte Extension (`d_world`), die alle Dokumente bezeichnet. Aufgrund bestimmter Kriterien können weitere, kleinere Extensionen ermittelt werden. Zu den unterstützten Parametern zählen die Erreichbarkeit ab einer

⁹im Fall XML beispielsweise Zugriff auf Elemente

¹⁰bekannt aus Lorel

URL, die Sprache der Dokumente, der Typ (ggf. benannt) und die Domäne der Dokumente (im Fall GETESS: Tourismus).

- zu den unterstützten Möglichkeiten aus dem Bereich des Information Retrieval zählen inhaltsbasierte Suche, soundex-Suche und proximity-Suche. Jede der implementierenden Klauseln kann (je nach Anwendbarkeit) weiter parametrisiert werden: unterstützt werden Fragetermgewichtung, Vorkommenshäufigkeit und eine Fehlerfunktion. Weiterhin ist ein Ranking aufgrund nutzerdefinierter Rankingfunktionen möglich.

5.3 IRQL im Vergleich mit anderen Ansätzen

IRQL vereint Konzepte aus den Bereichen Datenbankanfragesprachen, Anfragesprachen für semi-strukturierte Daten und Information Retrieval. Wir geben in diesem Abschnitt einen Überblick über Arbeiten auf diesen Gebieten und vergleichen diese mit unserem Ansatz. Wir verfolgen dabei das Ziel, die Charakteristika von IRQL herauszuarbeiten.

Eine Vielzahl existierender Suchmaschinen (wie z.B. Altavista oder Infoseek) nutzen Techniken des Information Retrieval, um die Suche nach (Web-)Dokumenten aufgrund von Stichworten, Phrasen und booleschen Verknüpfungen dieser Kriterien zu realisieren. Anfragen, welche die Struktur der indizierten Dokumente beachten, sind nur in einigen Fällen möglich. Prinzipiell realisieren derartige Suchmaschinen ausschließlich Selektionen: Konzepte wie Restrukturierung (Projektionen) oder gar Verknüpfungen (Joins) verschiedener Dokumente, die wir in IRQL umzusetzen planen, werden nicht unterstützt.

Ein System, welches die Attributierung der indizierten Daten unterstützt, ist freeWAIS-sf [PFH95, Pfe95]. Die Zerlegung von Dokumenten in Paare von Attribut und entsprechendem Wert erfolgt aufgrund des Dokumenttyps. Ein vordefinierter Typ ist etwa HTML, weitere Typen können vom Benutzer definiert werden. Unterstützte Anfragetypen sind auch hier nur Selektionen, wahlweise über dem gesamten Dokument oder in bestimmten Feldern. FreeWAIS-sf erlaubt die Suche anhand von Freitext, Phrasen, Wildcards, Soundex- und Proximity-Ausdrücken sowie booleschen Verknüpfungen dieser Kriterien. Vergleichsoperationen auf numerischen Werten werden ebenfalls unterstützt. Die in freeWAIS-sf unterstützte Attributierung wird in unserem Ansatz durch Möglichkeiten zur Restrukturierung von Dokumenten erweitert.

Die Nachteile der ausschließlichen Nutzung von Techniken des Information Retrieval für Anfragen an semistrukturierte Daten [Abi97, Clu97] wurden in der Vergangenheit von mehreren Autoren erkannt. Daraus resultierten Vorschläge bzw. Implementierungen von Anfragesprachen, welche ebenfalls Konzepte aus dem Feld der Datenbankanfragesprachen integrierten. Einen Literaturüberblick zu diesem Thema gibt [FLM98].

Lorel [AQM⁺97] ist die Anfragesprache des Systems Lore [MAG⁺97]. Die Syntax der Sprache ist OQL angelehnt. Semistrukturierte Daten werden durch Verwendung von OEM-Graphen als Datenmodell und durch Erweiterungen der Sprache gegenüber OQL unterstützt. Die Spracherweiterungen betreffen

- die Verwendung impliziter Typumwandlungen und
- den Einsatz von Pfadvariablen.

Pfadvariablen unterstützen Anfragen an unbekannte oder teilweise bekannte Schemata, durch implizite Typumwandlungen wird die Heterogenität semistrukturierter Daten abgedeckt. Einige Formen der inhaltsbasierten Suche (u.a. soundex) werden durch entsprechende Prädikate zur Verfügung gestellt. Ein Nachteil des verwendeten Datenmodells ist die fehlende Unterstützung geordneter Collections, so daß auf Sprachebene beispielsweise kein Ranking unterstützt werden kann.

WebSQL [AMM97] basiert auf dem relationalen Datenmodell und realisiert eine SQL-ähnliche Anfragesprache. Zu den zusätzlich umgesetzten Konzepten zählen Extensionsbildung aufgrund von Inhalt und Erreichbarkeit von Web-Dokumenten sowie Pfadausdrücke. Die Attributierung von Web-Dokumenten erfolgt aufgrund der existierenden Tags. Eine Restrukturierung der Daten, wie wir sie in IRQL planen, ist nicht möglich.

Ein gegenüber OEM-Graphen verbessertes Datenmodell wird in WebOQL [AM98] verwendet. Hypertrees bieten eine günstigere Möglichkeit zur Modellierung geschachtelter Strukturen und unterstützen zusätzlich geordnete Collections. Die Verwendung des "Web" als Datentyp ist die Grundlage einer Zahl von angebotenen Restrukturierungsoperationen. Die Anfragesprache basiert auf an die OQL-Syntax angelehnte SFW-Klauseln, erweitert beispielsweise um Konstrukte zur Ergebniserzeugung. Inhaltsbasierte

Anfragen werden in Form eines grep-Operators unterstützt. IRQL wird die in WebOQL umgesetzten Konzepte um Möglichkeiten des Information Retrieval, wie Fragetermgewichtung und Ranking erweitern.

UnQL [BDHS96] liegt ebenfalls ein Graph als Datenmodell zugrunde. Die Anfragesprache unterstützt Operationen wie Selektion, Projektion, Join und Gruppierung sowie Pfadausdrücke auf semistrukturierten Daten. Die realisierten Prädikate setzen im wesentlichen Bedingungen an die Erreichbarkeit von Knoten um. Sowohl geordnete Collections als auch inhaltsbasierte Suche werden nicht unterstützt.

W3QL [KS95] ist die Anfragesprache des W3QS im Stil von SQL. Der Schwerpunkt bei der Konzeption von W3QL wurde auf die Wiederverwendung existierender Werkzeuge gelegt, so daß beispielsweise Prädikate zur Realisierung der inhaltsbasierten Suche durch externe Programme umgesetzt werden. Sowohl Nestung von Anfrageklauseln als auch Restrukturierung werden nicht unterstützt.

Im Vordergrund der Strudel-Anfragesprache StruQL [FFLS97] steht die Möglichkeit zur Restrukturierung der existierenden Daten. In StruQL werden semistrukturierte Daten als OEM-Graph modelliert. Zu den unterstützten Anfrageprimitiven zählen daher Navigation mittels Pfadausdrücken, Projektion und Selektion sowie zur Restrukturierung existierender und Erzeugung neuer Graphen. Die Möglichkeit der Verwendung nutzerdefinierter Prädikate gestattet potentiell auch die Realisierung der inhaltsbasierten Suche.

WebLog [LSS96] basiert auf SchemaLog und unterstützt Attributierung und inhaltsbasierte Suche durch Verwendung von built-in oder nutzerdefinierten Prädikaten. Eine Restrukturierung der Daten ist möglich. Eine Gemeinsamkeit von WebLog und IRQL ist die Formulierung rekursiver Anfragen.

In WQL [LSCH98] werden sowohl das Web als auch die Struktur der einzelnen Dokumente modelliert. Durch die Anfragesprache werden Projektion, Selektion, Sortierung und Gruppierung realisiert. Anfragen an den Inhalt und Struktur der Web-Dokumente sind möglich. Nicht unterstützt werden Extensionsbildung, Nestung und Restrukturierung.

Neben den in den letzten Absätzen vorgestellten Anfragesprachen speziell für semistrukturierte Daten existieren auch Ansätze aus dem Bereich der Datenbanksysteme. So wird etwa durch SQL/MM [SQL95] ein Volltext-Datentyp erklärt, der inhaltsbasierte Anfragen an die so dargestellten Texte gestattet. Eine unterschiedlich ausgeprägte Realisierung dieses Ansatzes findet sich in kommerziellen Systemen in Form von Text-Extendern, Data-Blades, usw. wieder.

6 Zusammenfassung

Anliegen des Projektes GETESS ist die Bereitstellung einer textorientierten Suchmaschine für WWW-Dokumente. Um zielsicher Informationen im Internet zu finden, wird in GETESS die Idee verfolgt, die zu indizierenden Dokumente inhaltlich zu analysieren, die Ergebnisse dieser Analyse unter Verwendung einer Ontologie in Form von *Abstracts* zusammenzufassen und diese Zusammenfassungen in Datenbanken zu speichern.

In diesem Artikel wurde die Erstellung der Abstracts sowie deren Speicherung in Datenbanken betrachtet. Für die Beantwortung einer natürlichsprachlichen Suchanfrage wird auf die Abstract-Datenbank mittels einer speziellen Anfragesprache (IRQL) zugegriffen. Der Aufbau der IRQL wurde ebenfalls in diesem Artikel beschrieben.

Allgemein wurde ein Ansatz präsentiert, der die Kopplung von Suchmaschine und Datenbank-Technik beschreibt. Dabei ist der gewählte Ansatz weitgehend unabhängig von der für GETESS gewählten Domäne Tourismus und kann somit auf andere Suchmaschinen übertragen werden. Voraussetzung für die Adaption ist jedoch eine Beschreibung der aktuellen Domäne in Form einer Ontologie, um einen Datenbank-Entwurf für die Abstract-Strukturen erstellen zu können.

Ein Prototyp der GETESS-Suchmaschine wurde implementiert und wird derzeit getestet.

Literatur

- [Abi97] Abiteboul, S.: Querying Semi-Structured Data. In: Afrati, F. N.; Kolaitis, P. (Hrsg.): *Database Theory - ICDT '97, 6th International Conference, Lecture Notes in Computer Science*, Band 1186, S. 1–18. Springer Verlag, Delphi, Greece, Januar 1997.

- [AM98] Arocena, G. O.; Mendelzon, A. O.: WebOQL: Restructuring Documents, Databases, and Webs. In: *Proceedings of the Fourteenth International Conference on Data Engineering*, S. 24–33. IEEE Computer Society Press, Orlando, Florida, USA, Februar 1998.
- [AMM97] Arocena, G. O.; Mendelzon, A. O.; Mihaila, G. A.: Applications of a Web Query Language. In: *Proceedings of the 6th International WWW Conference*. Santa Clara, California, 1997.
- [ANS99a] *ANSI/ISO/IEC International Standard (IS) Database Language SQL – Part 1: SQL/Framework, ISO/IEC 9075-1:1999 (E)*, September 1999.
- [ANS99b] *ANSI/ISO/IEC International Standard (IS) Database Language SQL – Part 2: Foundation (SQL/Foundation), ISO/IEC 9075-2:1999 (E)*, September 1999.
- [AQM⁺97] Abiteboul, S.; Quass, D.; McHugh, J.; Widom, J.; Wiener, J. L.: The Lorel Query Language for Semistructured Data. *International Journal on Digital Libraries*, Band 1, Nr. 1, S. 68–88, 1997.
- [BDB⁺00] Bruder, I.; Düsterhöft, A.; Becker, M.; Bedersdorfer, J.; Neumann, G.: GETESS: Constructing a Linguistic Search Index for an Internet Search Engine. In: *NLDB 2000 — Proceedings of International Conference of Application of Natural Language to Information Systems*. Paris, France, jun 2000.
- [BDHS96] Buneman, P.; Davidson, S. B.; Hillebrand, G. G.; Suciu, D.: A Query Language and Optimization Techniques for Unstructured Data. In: Jagadish, H. V.; Mumick, I. S. (Hrsg.): *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, SIGMOD Record*, Band 25(2), S. 505–516. Montreal, Quebec, Canada, Juni 1996.
- [Bru99] Bruder, I.: Integration von Harvest und Smes in GETESS, Dezember 1999.
- [BYRN99] Baeza-Yates, R.; Rebeiro-Neto, B.: *Modern Information Retrieval*. ACM Press, Addison Wesley, New York, 1999.
- [Clu97] Cluet, S.: Modeling and Querying Semi-Structured Data. In: Pazienza, M. T. (Hrsg.): *Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology, International Summer School, SCIE-97, Lecture Notes in Computer Science*, Band 1299, S. 192–213. Springer Verlag, Frascati, Italy, 1997.
- [Coma] Understanding and Comparing Web Search Tools.
<http://web.hamline.edu/administration/libraries/search/comparisons.html>.
- [Comb] Search Engines Shoot-out - Top Engines Compared.
<http://coverage.cnet.com/Content/Reviews/Compare/Search2/>.
- [DG00] Düsterhöft, A.; Gröticke, S.: A Heuristic Approach for Recognizing a Document's Language Used for the Internet Search Engine GETESS. In: *NLIS 2000 — Proceedings of International Workshop of Natural Language to Information Systems*. Greenwich, UK, sep 2000.
- [DHK⁺99] Düsterhöft, A.; Heuer, A.; Klettke, M.; Priebe, D.; Prager, B.; Pretzel, J.; Wrenger, B.: GETESS: Ein Analyse- und Suchdienst für Texte im Internet. In: *2. IuK-Tage MV*, Juni 1999.
- [FFLS97] Fernandez, M. F.; Florescu, D.; Levy, A. Y.; Suciu, D.: A Query Language for a Web-Site Management System. In: *SIGMOD Record*, Band 26(3), S. 4–11, 1997.
- [FLM98] Florescu, D.; Levy, A. Y.; Mendelzon, A. O.: Database Techniques for the World-Wide Web: A Survey. In: *SIGMOD Record*, Band 27(3), S. 59–74, 1998.
- [Har] Harvest. <http://harvest.transarc.com>.
- [HP00] Heuer, A.; Priebe, D.: Integrating a Query Language for Structured and Semi-Structured Data and IR Techniques. In: *Proceedings of the 11th International Workshop on Database and Expert Systems Applications (DEXA 2000)*. IEEE Computer Society Press, September 2000. To appear.
- [Kir98] Kirsch, S.: The future of internet search, 1998.
<http://software.infoseek.com/stk/presentations/sigir.ppt>.
- [KM00] Klettke, M.; Meyer, H.: Xml and object-relational database systems - enhancing structural mappings based on statistics. In: *WebDB 2000 — Third International Workshop on the Web and Databases*. Dallas, may 2000.

- [KS95] Konopnicki, D.; Shmueli, O.: W3QS: A Query System for the World-Wide Web. In: Dayal, U.; Gray, P. M. D.; Nishio, S. (Hrsg.): *VLDB'95, Proceedings of 21th International Conference on Very Large Data Bases*, S. 54–65. Morgan Kaufmann Publishers, Zurich, Switzerland, September 1995.
- [LSCH98] Li, W.-S.; Shim, J.; Candan, K. S.; Hara, Y.: WebDB: A Web Query System and its Modeling, Language, and Implementation. In: *Proceedings of the IEEE Forum on Research and Technology Advances in Digital Libraries, IEEE ADL'98*, S. 216–227. Santa Barbara, CA, USA, April 1998.
- [LSS96] Lakshmanan, L. V. S.; Sadri, F.; Subramanian, I. N.: A Declarative Language for Querying and Restructuring the WEB. In: *Proceedings: Sixth International Workshop on Research Issues in Data Engineering — Interoperability of Nontraditional Database Systems*, IEEE-CS 1996, S. 12–21. New Orleans, Louisiana, USA, Februar 1996.
- [MAG⁺97] McHugh, J.; Abiteboul, S.; Goldman, R.; Quass, D.; Widom, J.: Lore: A Database Management System for Semistructured Data. In: *SIGMOD-Record*, Band 26(3), S. 54–66, September 1997.
- [Pfe95] Pfeifer, U.: *freeWAIS-sf*. Universität Dortmund, Oktober 1995. Manual of the enhanced freeWAIS distribution.
- [PFH95] Pfeifer, U.; Fuhr, N.; Huynh, T.: Searching Structured Documents with the Enhanced Retrieval Functionality of freeWAIS-sf and SFgate. In: *Proceedings of The Third International World-Wide Web Conference*. Darmstadt, Germany, April 1995.
- [Por99] Porst, B.: Untersuchungen zu Datentypenweiterungen für XML-Dokumente und ihre Anfragemethoden am Beispiel von DB2 und Informix. Diplomarbeit, Universität Rostock, 1999.
- [SBB⁺99] Staab, S.; Braun, C.; Bruder, I.; Düsterhöft, A.; Heuer, A.; Klettke, M.; Neumann, G.; Prager, B.; Pretzel, J.; Schnurr, H.-P.; Studer, R.; Uszkoreit, H.; Wrenger, B.: A System for Facilitating and Enhancing Web Search. In: *IWANN '99 — Proceedings of International Working Conference on Artificial and Natural Neural Networks*. Alicante, ES, 1999.
- [SEW] Search Engine Watch. <http://www.searchenginewatch.com>.
- [SQL95] *ISO Working Draft, SQL Multimedia and Application Packages (SQL/MM), Part 2: Full-Text*, September 1995.
- [TRE] Text Retrieval Conference. <http://trec.nist.gov/>.