Lehrstuhl für Datenbanken und Informationssysteme
Prof. Dr. Bernd Walter
Universität Trier
Fachbereich IV

Diplomarbeit zum Thema

"Regelbasierte Extraktion und asymmetrische Fusion bibliographischer Informationen"

Oliver Hoffmann
30. September 2009

o.hoffmann@coworkersnet.de

Zusammenfassung

Die Digitale Bibliothek DBLP stellt im Bereich der Informatik eine populäre und wichtige Quelle bibliographischer Informationen dar und wird von Wissenschaftlern auf der ganzen Welt genutzt. Dies liegt nicht zuletzt daran, dass bei der Erfassung der Daten eine größtmögliche Sorgfalt an den Tag gelegt wird, die eine manuelle Bearbeitung unerlässlich macht. Diese Arbeitsschritte sind jedoch äußerst zeit- und arbeitsintensiv und oftmals fehlen die benötigten menschlichen Ressourcen, um weitere Publikationen zu erfassen. Aus diesem Grund soll versucht werden, einige der Arbeitsschritte zur Erfassung und Pflege bibliographischer Daten weitestgehend zu automatisieren, um somit bei gleichzeitiger Beibehaltung der Datenqualität eine Steigerung der Produktivität herbei zu führen.

Die vorliegende Arbeit stellt zwei Konzepte vor, um diese Aufgabe zu übernehmen. Zum einen soll die Erfassung bibliographischer Daten von Webservern wichtiger Verlage oder digitaler Bibliotheken automatisiert werden, indem Methoden der Informationsextraktion zum Einsatz kommen. Zum anderen sollen die auf diese Weise gewonnenen Datensätze – aber auch solche, die bereits in DBLP vorhanden sind – durch die Fusion mit anderen Datenquellen um zusätzliche Informationen angereichert, oder fehlerhafte Daten korrigiert werden. Hierzu werden wir zunächst einige theoretische Überlegungen zur Fusion verschiedener bibliographischer Informationen anstellen und eine geeignete Notation einführen, mit deren Hilfe sich derartige Fusionsprobleme beschreiben lassen, um dann äußerst spezifische, in der Praxis häufig auftretende, Probleme zu untersuchen und mittels der im Rahmen der vorliegenden Arbeit erstellten Software zu lösen.

Auch wenn die hier diskutierten Konzepte sicherlich keine allgemeingültigen und in jedem Fall Erfolg versprechenden Allround-Lösungen darstellen, so kann durch Einsatz der beschriebenen Konzepte und der entsprechenden Software eine erhöhte Leistungssteigerung bei der Erfassung und Pflege der bibliographischen Daten in DBLP erreicht werden.

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ACM	**A**ssociation for **C**omputing **M**achinery
AI	**A**rtificial **I**ntelligence
AJAX	**A**synchronous **J**avaScript **a**nd **X**ML
ASCII	**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange
ASP	**A**ctive **S**erver **P**ages
BHT	**B**ibliography **H**yper**T**ext
BMC	**B**io**M**ed **C**entral
CFML	**C**old**F**usion **M**arkup **L**anguage
CGI	**C**ommon **G**ateway **I**nterface
CSS	**C**ascading **S**tyle **S**heets
DARPA	**D**efense **A**dvanced **R**esearch **P**rojects **A**gency
DBLP	**D**igital **B**ibliography & **L**ibrary **P**roject
DOI	**D**igital **O**bject **I**dentifier
DOM	**D**ocument **O**bject **M**odel
EE	**E**lectronic **E**dition
EUDL	**E**uropean **U**nion **D**igital **L**ibrary
FAQ	**F**requently **A**sked **Q**uestions
GUI	**G**raphical **U**ser **I**nterface
HTML	**H**yper**t**ext **M**arkup **L**anguage
HTTP	**H**yper**t**ext **T**ransfer **P**rotocol
ICST	**I**nstitute for **C**omputer Sciences, **S**ocial-Informatics and **T**elecommunications Engineering
IDF	**I**nternational **D**OI **F**oundation
IE	**I**nformation **E**xtraction
IEEE	**I**nstitute of **E**lectrical and **E**lectronics **E**ngineers
IGI	**I**dea **G**roup **I**ncorporated

IIS	**I**nternet **I**nformation **S**ervices
IR	**I**nformation **R**etrieval
ISBN	**I**nternational **S**tandard **B**ook **N**umber
JSP	**J**ava**S**erver **P**ages
LNCS	**L**ecture **N**otes in **C**omputer **S**cience
MathML	**Math**ematical **M**arkup **L**anguage
MUC	**M**essage **U**nderstanding **C**onferences
NLP	**N**atural **L**anguage **P**rocessing
PDF	**P**ortable **D**ocument **F**ormat
PHP	**P**HP: **H**ypertext **P**reprocessor (rekursives Akronym)
SIAM	**S**ociety for **I**ndustrial and **A**pplied **M**athematics
SSI	**S**erver **S**ide **I**ncludes
ST	**S**cience & **T**echnology
STM	**S**cience, **T**echnology, **M**edicine
TOC	**T**able **O**f **C**ontents
UML	**U**nified **M**odeling **L**anguage
URL	**U**niform **R**esource **L**ocator
UTF	**U**nicode **T**ransformation **F**ormat
W3	$\widehat{=}$ WWW
WWW	**W**orld **W**ide **W**eb
XML	E**x**tensible **M**arkup **L**anguage

Inhaltsverzeichnis

Einleitung

Motivation

"DBLP - 1 Million Einträge", so lautete die Überschrift der Einladung zu zwei Festvorträgen, die die Abteilung Informatik der Universität Trier im Sommer 2008 zu Ehren der digitalen Bibliothek DBLP veranstaltete. Mit Gastvorträgen von Gottfried Vossen und François Bry sowie einem anschließenden Umtrunk wurde die Überschreitung der magischen Millionengrenze erfasster Publikationen gebührend gefeiert. Von einer aus eher experimentellen Gründen ins Leben gerufenen Sammlung weniger Inhaltsverzeichnisse der Bereiche "Datenbanksysteme und Logikprogrammierung" hat sich DBLP innerhalb von fünfzehn Jahren zu einer der wichtigsten Quellen bibliographischer Informationen verschiedenster Fachrichtungen des Bereichs der Informatik entwickelt.

Im Gegensatz zu anderen wissenschaftlichen Datensammlungen wie CiteSeer[1] oder Google Scholar[2] nutzt DBLP jedoch keine vollkommen automatisch ablaufenden Programme (Robots, Spiders etc.) zur Indexierung neuer Daten, um deren Qualität auf einem möglichst hohen Niveau anzusiedeln. Informationen zu einzelnen Publikationen werden aktiv – beispielsweise durch Herunterladen aus dem Internet – oder passiv – beispielsweise durch Zusendung durch einen Verlag – beschafft, in ein geeignetes Eingabeformat transformiert und mittels manueller oder halbautomatischer Verfahren einer Qualitätsprüfung unterzogen, bevor sie in den Datenbestand aufgenommen werden ([Reu07]). Ein schwerwiegendes Problem bei der Vergrößerung des Datenbestandes stellt daher die äußerst knappe Zahl an menschlichen Ressourcen dar, die zur Erfassung und Pflege der bibliographischen Daten zur Verfügung stehen.

Die Vorliegende Arbeit verfolgt das Ziel, die Erfassung und Pflege jener Daten in geringem Maße zu automatisieren. Stets wiederkehrende Aufgaben, wie die Transformation von Inhaltsverzeichnissen diverser Journale oder Konferenzbände von den Servern großer Verlage in ein geeignetes Eingabeformat, sollen automatisiert werden, um auf diese Weise Zeit und Arbeitskraft einzusparen, die an anderen Stellen dringend benötigt wird. Hier werden Methoden der *Informationsextraktion*, eines Forschungsgebiets der Informatik,

[1] `http://citeseer.ist.psu.edu`
[2] `http://scholar.google.com`

zum Einsatz kommen, mit deren Hilfe Webseiten großer Verlage oder publizierender Gesellschaften durchforstet werden können.

Zudem sollen einfache Verfahren entwickelt werden, um die Qualität bestehender wie auch neu zu erfassender Daten aufzubereiten, indem zusätzliche Informationen anderer Quellen zu Rate gezogen werden. Oftmals sind Informationen in mehreren Datenquellen zu finden, beispielsweise in digitalen Bibliotheken, Konferenzprogrammen oder auf privaten Webseiten der Autoren. Hier wird es unser Ziel sein, durch eine geschickte Ausnutzung der uns zur Verfügung stehenden Daten zusätzliche Informationen zu erhalten. Durch die *Fusion* jener Daten können anschließend Fehler korrigiert oder unvollständige Datensätze ergänzt werden.

Heute, im September 2009, umfasst der Datenbestand von DBLP bereits über 1,2 Millionen Datensätze, und täglich kommen im Durchschnitt etwa 375 hinzu, wobei die Tendenz, gerade bei den Zeitschriftenartikeln, stark steigend ist.[3] Da mittels der im Rahmen dieser Arbeit entwickelten Software neue Datensätze noch schneller hinzugefügt werden können und Verbesserungen nicht mehr rein manuell vollzogen werden müssen, ist eine weitere Steigerung der Produktivität zu erwarten – wodurch die zweite Million möglicherweise in eine noch nähere Zukunft rückt. Wir können also stark davon ausgehen, dass der nächste Umtrunk keine fünfzehn Jahre auf sich warten lassen wird.

Kapitelüberblick

Gemäß dem Titel "Regelbasierte Extraktion und asymmetrische Fusion bibliographischer Informationen" werden wir uns in der vorliegenden Arbeit mit der Gewinnung und Aufbereitung bibliographischer Daten beschäftigen. Dies geschieht anhand von Datensätzen, die dem Datenbestand von DBLP hinzugefügt bzw. innerhalb dieses Bestandes verbessert werden sollen. Dabei werden wir uns zunächst ausschließlich mit der Beschaffung neuer Daten beschäftigen, später dann mit der Fusion gleicher Datensätze aus unterschiedlichen Quellen.

Zunächst werden wir uns einen groben Überblick über den Aufbau der Arbeit verschaffen: Abbildung 0.1 zeigt deren schematischen Aufbau. Die Nummern der einzelnen Kapitel werden, ihren jeweiligen Inhalten entsprechend, innerhalb dreier sich schneidender Mengen dargestellt, welche die drei behandelten Themenbereiche symbolisieren. Man erkennt, dass die beiden Themengebiete "Informationsextraktion" (1) und "bibliographische Daten" (2) zunächst separat, danach in Verbindung miteinander (3, 4) behandelt

[3] Diese Informationen stammen von Florian Reitz, einem wissenschaftlichen Mitarbeiter des Lehrstuhls für Datenbanken und Informationssysteme an der Universität Trier, und wurden anhand historischer Daten seit Ende der 90'er Jahre ermittelt.

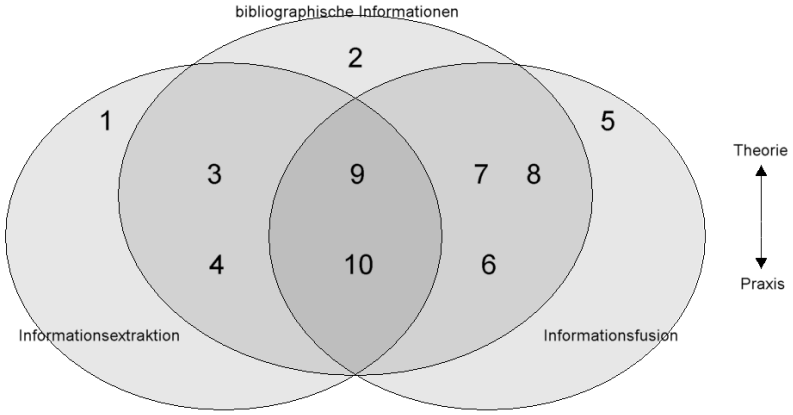


Abb. 0.1: Gliederung der vorliegenden Arbeit
Quelle: eigene Erstellung

werden. Anschließend werden wir uns mit dem dritten Bereich, der "Informationsfusion" (5) beschäftigen, welcher im weiteren Verlauf der Arbeit (6, 7, 8) ebenfalls mit den bibliographischen Daten in Verbindung gebracht wird. In den beiden letzten Kapiteln (9, 10) werden schließlich alle drei Themenbereiche miteinander kombiniert. Innerhalb der jeweiligen Mengen lässt sich anhand der vertikalen Anordnung der Kapitelnummern erkennen, ob diese eher theoretisch oder eher praktisch orientiert sind. Dies soll jedoch lediglich eine Tendenz erkennen lassen. So fällt beispielsweise auf, dass sich der Extraktion bibliographischer Informationen (3, 4) von Seiten der Theorie angenähert wird, während bei der Fusion bibliographischer Informationen (6, 7, 8) der Weg über die Praxis gewählt wurde. Dies liegt in der logischen Struktur jener Kapitel begründet, die wir nun genauer betrachten werden.

Kapitel 1 liefert zunächst einen Überblick über das Gebiet der Informationsextraktion (IE), deren Entwicklung und Anwendungsmöglichkeiten. Es werden verschiedene Dimensionen der IE vorgestellt, mittels derer unterschiedliche Extraktionssoftware kategorisiert werden kann. Unser besonderes Augenmerk wird hier den so genannten "Wrappern" gelten: Softwaretools, die bei der Extraktion regelbasiert vorgehen und die gesuchten Informationen durch Betrachtung ihrer unmittelbaren Umgebung identifizieren.

Im Themengebiet der bibliographischen Informationen wird in **Kapitel 2** zunächst die Datensammlung DBLP vorgestellt. Hier werfen wir einen besonderen Blick auf die internen Datenformate, in welchen die bibliographischen Informationen vorliegen, da die zur Arbeit gehörige Software in der Lage sein muss, jene Formate zu lesen und auch zu erzeugen. Hiernach erfolgt ein kurzer Abriss über die gewünschte Datenqualität, um uns klar zu machen, in welcher Form und Genauigkeit wir die zu erfassenden Daten bevorzugen. Den Abschluss des Kapitels bildet die Beschreibung eines Datenmodells, welches wir im weiteren Verlauf der Arbeit zur Beschreibung der Extraktions- und Fusionsvorgänge nutzen möchten und welches sich ebenfalls in der erstellten Software widerspiegelt.

Um einen Eindruck über die zu erwartenden Aufgaben und Probleme bei der Extraktion bibliographischer Daten zu erhalten, werden wir in **Kapitel 3** eine Reihe für die Informatik wichtiger Verlage und digitaler Bibliotheken, die über das WWW zugänglich sind, untersuchen, und einer genauen Prüfung unterziehen. Dies wird uns helfen, Strategien zu entwickeln, mit denen die Wrapper-Software eine Extraktion vornehmen kann.

Kapitel 4 beschäftigt sich schließlich mit der konkreten Umsetzung dieser Software und zeigt die Ziele der Extraktion sowie den Ablauf eines solchen Extraktionsvorgangs – sowohl theoretisch als auch anhand eines umfangreichen Beispiels.

In **Kapitel 5** werden wir uns dann mit dem Themengebiet der Informationsfusion beschäftigen. Wir werden sehen, dass der direkte Begriff eher im Bereich der Sensortechnik angesiedelt ist und in unserem Falle der Begriff der *Datenfusion* zutreffender scheint; doch existiert hier in der Literatur keine klare Abgrenzung jener Begriffe, weshalb wir sie stets synonym verwenden werden. Zudem werden wir das Datenintegrationsmodell von Bleiholder und Naumann kennen lernen, welches uns im weiteren Verlauf der Arbeit helfen wird, die einzelnen Phasen der Fusion exakt zu benennen.

Um zu wissen, welche Ziele wir mit der Fusion bibliographischer Daten verfolgen, ist es nun zunächst notwendig, in **Kapitel 6** einen Blick auf die praktischen Anwendungsgebiete zu werfen, in welchen uns eine Fusion nützlich erscheint. Wir werden zudem einige grundlegenden Definitionen treffen und diskutieren, in welcher Weise der Benutzer befähigt sein sollte, auf die Fusion Einfluss zu nehmen.

Da bei der Fusion der Datensätze stets deren Semantik eine Rolle spielt, ist es notwendig, für alle Objekte unseres in Kapitel 2 definierten Datenmodells festzulegen, wann diese fusioniert werden können und wie das entsprechende Ergebnis aussieht. Hierzu werden wir in **Kapitel 7** zunächst die Fusion *komplexer* Objekte, d.h. solcher, die aus mehreren gleichartigen Objekten bestehen (wie beispielsweise einer Liste von Autorennamen, die aus einzelnen Objekten, den Autorennamen, besteht) betrachten. Hier ist ein Algorithmus von Nöten, der in der Lage ist, *Partner* – d.h. Paare von Objekten, die miteinander fusioniert werden können – zu finden. Anschließend müssen die *Singles*, d.h. all jene Objekte, die bei diesem Vorgang übrig geblieben sind, betrachtet werden.

Kapitel 8 dagegen widmet sich der Fusion *simpler* Objekte, d.h. solcher Objekte, die entsprechend nur aus einem einzelnen Wert bestehen. Hier werden geeignete Definitionen getroffen, wann diese fusioniert werden können und wie das Fusionsergebnis aussehen soll. Einen besonderen Status nehmen hierbei die Autorennamen ein, bei welchen es sich prinzipiell um Listen von Namensteilen handelt, die jedoch als Einheit betrachtet und zusammenhängend verarbeitet werden müssen.

Während bei den bis hierher betrachteten Fällen eine Asymmetrie nur in der Gewichtung der beiden Eingabequellen zu finden war – wir werden im Falle einer Unsicherheit den Informationen der erste, primären Quelle stets größeres Vertrauen schenken als denen

der zweiten, sekundären Quelle – werden wir in den letzten beiden Kapitel die Unstrukturiertheit der zweiten Quelle erhöhen und somit eine stärkere Asymmetrie beider Quellen herbeiführen. Daher werden wir uns in **Kapitel 9** zunächst einen Überblick über jene Problematik verschaffen, bei welcher die zuvor studierten Methoden der Informationsextraktion *und* -Fusion zum Einsatz kommen werden. Hierzu wird zunächst eine Studie durchgeführt, bei welcher 100 Webseiten, die Programme zu Konferenzen aus dem Bereich der Informatik enthalten, nach verschiedenen Kriterien untersucht werden.

Anschließend werden in **Kapitel 10** drei verschiedene Strategien entwickelt, mittels derer die genannte Problematik angegangen werden kann. Wir werden die unterschiedlichen Ansätze diskutieren und auf die im vorherigen Kapitel gewonnenen Testdaten anwenden, um Informationen über deren Güte zu erhalten. Wir werden sehen, dass an dieser Stelle noch ein erheblicher Spielraum für weitere Forschung besteht, zu welcher die vorliegende Arbeit ermutigen möchte. Auch die Fusion mit anderen unstrukturierten Quellen, beispielsweise Konferenzprogrammen in Form von PDF-Dateien, wird hierbei angesprochen. Abschließend werden wir die Fusion mit der wohl größten verfügbaren, unstrukturierten Datenquelle betrachten: dem WWW. Wir werden sehen, dass eine Websuchmaschine wie beispielsweise Google[4] als Orakel genutzt werden kann, um weitere Informationen zu bestehenden Daten zu erhalten.

In **Anhang A** findet sich schließlich eine detailliertere Beschreibung der zu dieser Arbeit gehörige Software, die das konkrete Zusammenspiel aller beteiligten Klassen näher erläutert. **Anhang B** liefert eine kurze Dokumentation der Bedienung eben jener Software. Um die konkrete Entwicklung weiterer Informationsextraktionstools zu fördern, liefert **Anhang C** ein kleines Tutorial, das die Konstruktion eines neuen Wrappers – von der Studie der HTML-Seiten bis hin zur Formulierung geeigneter Testfälle – erläutert. **Anhang D** und **Anhang E** enthalten schließlich die Ergebnisse der beiden durchgeführten Studien in tabellarischer Form.

Der Quellcode der im Rahmen dieser Diplomarbeit erstellten und dokumentierten Software liegt der Arbeit auf CD-ROM bei und ist mit zahlreichen Kommentaren versehen.

[4] `http://www.google.com`

Kapitel 1

Informationsextraktion

Bei der *Informationsextraktion* (engl. Information Extraction, IE) handelt es sich um ein eigenständiges Forschungsgebiet der Informatik, das wegen der Fülle an Informationen, die beispielsweise und vor allem das World Wide Web (WWW) zu bieten hat, in den letzten Jahrzehnten immer mehr an Bedeutung gewonnen hat und weiterhin gewinnt. Darüber, was genau unter IE zu verstehen ist, existieren je nach Standpunkt verschiedene Sichtweisen, die sich in den Details unterscheiden.

Aus Sicht der Computerlinguistik (Natural language processing, NLP), dem Forschungsgebiet, aus welchem sich die IE ursprünglich entwickelte (siehe Abschnitt 1.1), besteht die IE darin, einem Text, der in natürlicher Sprache verfasst ist, relevante Informationen zu entnehmen ([Cun06]). Neumann ging 2001 sogar noch weiter und forderte nicht nur die Entnahme der Informationen, sondern auch deren Strukturierung ([Neu01]). Moens und Hiemstra bezeichnen die IE auch als "automatic content recognition", deren Ziel die Identifizierung von Einheiten (Personen, Orte, Produkte) und deren semantischer Attribute (z.B. Bewertungen von Personen oder Produkten, Beziehung zwischen den Einheiten) darstellt ([MH09]).

Die so genannte "Web-IE" dagegen beschäftigt sich mit der Extraktion von Informationen speziell aus Dokumenten des WWW, welche, im Gegensatz zu Texten der natürlichen Sprache, entsprechend der verwendeten Seitenbeschreibungssprache (HTML, XML) bereits eine klarere Struktur aufweisen. Hier ist es möglich, völlig andere Strategien zur Informationsextraktion zu entwickeln, bei denen beispielsweise die Struktur des zu Grunde liegenden 'Document Object Models' (DOM) ausgenutzt wird (wie z.B. in [LGM08] und [LG09]), und/oder die Tatsache, dass Daten innerhalb von Webseiten oftmals in Form von Tabellen oder Listen auftreten ([CHJ02]). Wir werden in der vorliegenden Arbeit ebenfalls Web-IE betreiben und uns daher ab Abschnitt 1.5 genauer mit dieser Form der IE beschäftigen.

Auch wenn die IE in Bezug auf Textquellen am weitesten entwickelt ist, so hält sie auch in andere Medien Einzug, beispielsweise bei der Personenerkennung in Bildern oder Videos ([MH09]).

Oftmals wird die Extraktion von Informationen auch nicht als eigenständige Aufgabe, sondern eher als Vorbereitung für weiter führende Aufgaben, beispielsweise die Auswertung mittels 'Data Mining' (siehe Abschnitt 1.3.3), angesehen. Auch der Informationsfusion, mit welcher wir uns in Kapitel 5 befassen werden, geht oftmals eine entsprechende Informationsextraktion voraus.

1.1 Entwicklung der IE

Die Ursprünge der IE im heutigen Sinne liegen im Ende der 80er Jahre des vergangenen Jahrhunderts und gehen vor allem auf den Bereich des NLP zurück. Zwar ist bereits in den 60er Jahren von "fact extraction" die Rede, doch gab es keinerlei formale und weithin akzeptierten Definitionen ([Cun06]).

Die wichtigsten, noch heute gültigen Grundlagen wurden innerhalb der sieben "Message Understanding Conferences" (MUC) gelegt, welche zwischen 1987 und 1997 von der "Defense Advanced Research Projects Agency" (DARPA), der zentralen Forschungs- und Entwicklungseinrichtung des U.S. Verteidigungsministeriums, veranstaltet wurden und je einen besonderen Schwerpunkt behandelten, beispielsweise die Analyse terroristischer Aktivitäten (MUC-3 und MUC-4) oder Personalwechsel in der Wirtschaft (MUC-6) [AI99]. Eine gute Übersicht über MUC-1 bis 6 liefert [GS96].

In den beiden letzten Jahrzehnten hat sich die IE darauf aufbauend weiter entwickelt. Die ersten IE-Systeme arbeiteten stets regelbasiert, wobei deren Regeln manuell codiert wurden. Da dies im allgemeinen Fall einige Nachteile mit sich bringt (eine genauere Analyse hierzu erfolgt in Abschnitt 1.6), entwickelten sich ab der Mitte der 90er Jahre Techniken, um eben jene Regeln automatisch zu erstellen ([KWD97]). Um jedoch auch solche Quellen bearbeiten zu können, deren Struktur ein größeres 'Rauschen' (im engl. *noisy*) aufwies, wurden statistische Methoden entwickelt. Zum heutigen Stand der Forschung werden beide Methoden (regelbasiert und statistisch) je nach Einsatzgebiet verwendet; es gibt "keinen klaren Gewinner" ([Sar08]).

Mit der Zunahme der Informationsflut, vor allem im Internet, gewinnt die IE, wie bereits eingangs erwähnt, immer weiter an Bedeutung, und deren Erforschung und Verbesserung wird daher mit wachsendem Interesse betrieben. War noch vor einigen Jahren stets die Gewinnung äußerst spezieller Informationen einer fest vorgeschriebenen Domäne das Ziel, so wird vor allem im Bereich der "Künstlichen Intelligenz" (Artificial Intelligence, AI) an der Erstellung umfangreicher Wissensdatenbanken mittels Informationsextrak-

tion aus großen, in natürlicher Sprache geschriebenen Texten gearbeitet ([DP08]), um 'Turing's Traum', "the dream of a machine we can talk with just like a person, and which is therefore (at least) our intellectual equal" näher zu gelangen ([Sch06]).

1.2 Anwendungsgebiete der IE

Neben den in den vorherigen Abschnitten bereits genannten Möglichkeiten, bei denen IE sinnvoll zum Einsatz kommen kann, existiert eine weitere, nahezu unbegrenzte Anzahl von Anwendungsgebieten. An dieser Stelle soll daher nur eine knappe Auswahl unterschiedlicher Anwendungen genannt werden.

Preisvergleiche (Comparision shopping) Webportale wie beispielsweise "billiger.de"[1], die das jeweils günstigste Angebot diverser Konsumartikel bereitstellen, erfreuen sich immer größerer Beliebtheit. Diese bestehen intern aus Datenbanken, die mittels Web-IE ständig aktuell gehalten werden, indem Informationen diverser Online-Shops extrahiert und entsprechend ausgewertet werden. Auch die Verknüpfung derartiger Produktpreise mit Produktinformationen, die wiederum in natürlicher Sprache verfassten Berichten entstammen, stellt eine sinnvolle Anwendung dar ([SALM01]).

Zitat-Datenbanken Im WWW existieren zahlreiche Zitat-Datenbanken, zu deren populärsten Vertretern 'CiteSeer'[2] und 'Google Scholar'[3] zählen. Diese extrahieren Informationen zu Publikationen und Autoren aus unterschiedlichsten Quellen (Konferenzseiten, persönlichen Webseiten der Autoren etc.) und stellen Datenbanken zur Verfügung, die dabei helfen, "die wichtigsten Arbeiten auf dem Gebiet der wissenschaftlichen Forschung zu ermitteln."([Goo09]). Die automatische Erstellung solcher Datenbanken benötigt Informations- und Struktur-Extraktion von Webseiten und/oder PDF-Dokumenten auf unterschiedlichsten Ebenen ([Sar08]).

Anwendungen in der Bio-Informatik Auch im Bereich der Bio-Informatik gibt es einige interessante Anwendungen, von denen die Erkennung so genannter Protein-Protein-Interaktionen (PPI) in online verfügbaren, wissenschaftlichen Veröffentlichungen zu den wichtigsten zählt. Viele dieser Beziehungen sind ausschließlich innerhalb der Fachliteratur in Texten, die in natürlicher Sprache verfasst sind, beschrieben und müssen extrahiert werden, um sie in Datenbanken zu speichern und statistisch auswerten zu können ([KT04]).

[1] `http://www.billiger.de`
[2] `http://citeseerx.ist.psu.edu`
[3] `http://scholar.google.com`

Anwendungen in der Notfall-Medizin Die Unterstützung ärztlichen Handelns in Notfällen kann durch Bereitstellung der relevanten Informationen erfolgen. Da diese meist in textueller Form vorliegen, dem jeweiligen Arzt zur Sichtung des Datenmaterials jedoch nur extrem wenig Zeit zur Verfügung steht, können entsprechende Systeme die Ärzte effektiv in deren Tätigkeit unterstützen ([HGE07]). Hier steht also, im Gegensatz zu den vorherigen Beispielen, vor allem die zeitliche Komponente der IE im Vordergrund.

Eine ausführlichere Liste typischer Anwendungsgebiete ist beispielsweise in [Sar08] auf den Seiten 264-268 zu finden.

1.3 Abgrenzung zu ähnlichen Forschungsgebieten

Im Folgenden sollen die Gemeinsamkeiten, aber auch die Unterschiede zu verwandten Forschungsgebieten aufgezeigt werden.

1.3.1 Abgrenzung zum IR

Information Extraction (IE) und *Information Retrieval* (IR) sind eng miteinander verwandte Forschungsgebiete der Informatik. Beiden ist das Ziel gemeinsam, relevante Informationen aus einer ansonsten evtl. unüberschaubaren Datenmenge zu filtern und entweder einem Anwender oder einer weiter verarbeitenden Anwendung zur Verfügung zu stellen. Dabei arbeitet ein IR-System jedoch stets mit kompletten Dokumenten: Es sucht jene Dokumente, die für eine bestimmte Anfrage relevant sind. Anschließend ist es die Aufgabe des Benutzers, die Dokumente zu lesen und die für ihn relevanten Informationen herauszufinden. Eine IE-Anwendung dagegen analysiert die Dokumente, versucht deren Semantik zu erfassen und liefert nur die jeweils relevanten Informationen. Sie nimmt dem Benutzer praktisch die Aufgabe, das Dokument vollständig zu lesen, ab und präsentiert sofort die gewünschten Informationen. Daraus folgt logischerweise, dass IE-Systeme erheblich komplexer als IR-Systeme sind und bei der Erstellung einen höheren Grad an Wissen erfordern, dafür aber gerade bei großen Datenmengen eine enorme Zeitersparnis garantieren ([Cun06]). Abbildung 1.1 zeigt bildlich den Unterschied zwischen IR und IE: Ziel ist es nicht, ganze Dokumente zu finden, sondern sehr eingegrenzte Ergebnisse in Form von Einheiten (Entities).

Eine besondere Herausforderung im Hinblick auf eine semantische Suche ist es, IE und IR miteinander zu verbinden, indem die mittels der Extraktion gefundenen Ergebnisse während des Retrievals verwendet werden, um die Gewichtung der als relevant erachteten Dokumente zu verändern ([MH09]).

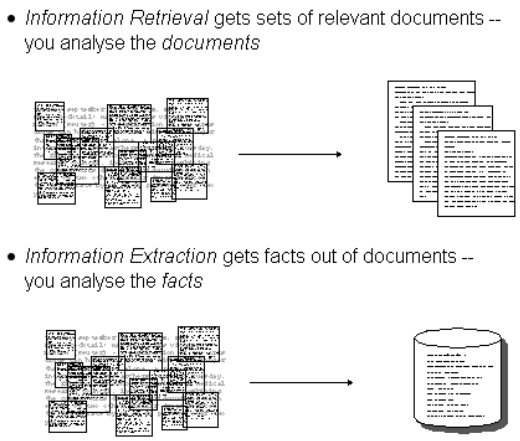


Abb. 1.1: Unterschied zwischen IR und IE: 'Information Retrieval' liefert Mengen relevanter Dokumente, 'Information Extraction' dagegen liefert gleich die entsprechenden Fakten. *Quelle:* `http://gate.ac.uk/ie`

1.3.2 Abgrenzung zur 'Text Summarization'

Ein ebenfalls aus dem Bereich der Computerlinguistik stammendes Forschungsgebiet der 'Information Compression' beschäftigt sich u.a. mit der *Text Summarization* (Textzusammenfassung). Hierbei liegt der Schwerpunkt darin, vollständige Dokumente semantisch zu erfassen und ihre 'Kernaussage' in möglichst kompakter Form wiederzugeben. Dabei geht es also nicht um die Identifikation einzelner Entitäten, sondern um eine Verkürzung des gesamten Textes, der die wichtigsten Aussagen der Quelle beinhalten muss ([SJ09]).

1.3.3 Abgrenzung zum 'Data-Mining'

Das *Data-Mining* beschäftigt sich ebenfalls mit der Gewinnung von Informationen aus Text- oder Bildquellen. Hier sind einige Überschneidungen zur IE erkennbar, doch geht es beim Data-Mining primär um die "Erkennung von Regularitäten in Daten" und weniger um deren Semantik. In Bezug auf Dokumente des WWW spricht man hier auch von 'Web-Mining', welches sich nicht nur mit den "Regularitäten in Texten und Multi-Media Objekten" sondern auch mit der "Erkennung von Regularitäten in der Benutzung von Web Dokumenten", sowie "in der Struktur von Web Dokumenten und ihrer Relationen" beschäftigt ([EHS04]).

1.3.4 Abgrenzung zum 'Screen Scraping'

Das *Screen Scraping*, zu Deutsch etwa "Bildschirm auskratzen" beschäftigt sich damit, für menschliche Benutzer bestimmte Inhalte (meist Inhalte von Webseiten, was zum Begriff des 'Web Scraping' führt) maschinell zu erfassen und zu verwerten. Damit überschneidet sich dieses Gebiet eindeutig mit der IE bzw. Web-IE. Das 'Screen Scraping' ist jedoch eher dem Gebiet des Hacking zuzuordnen, und es lastet ihm ein negativ besetzter Beigeschmack an. Die Gewinnung der Daten dient meist nicht wissenschaftlichen, sondern eher kommerziellen Zwecken, wobei man sich im Hinblick auf die Legalität oftmals in juristischen Grauzonen bewegt.[4] Eine umfassende Beschreibung moderner 'Screen Scraping'-Technologien bietet [Sch07].

1.4 Strukturiertheit einer Datenquelle

In der Literatur existieren verschiedene Definitionen der Strukturiertheit einer Eingabequelle, die sich oftmals in den Details unterscheiden. Fest steht i.d.R. der Begriff *strukturierter* Daten, welche sich dadurch auszeichnen, dass ihre spezielle Semantik klar und eindeutig erkennbar ist. Dies ist sowohl bei Datenbanken im engeren Sinne der Fall, aber auch bei allen Dateiformaten, die eine klare Zuordnung ermöglichen, wie beispielsweise in unserem Fall die DBLP-Records oder die BHT-Dateien (siehe Kapitel 2.2). Eine exakte, formale Definition strukturierter Daten findet sich beispielsweise in [AGM03].

Sarawagi bezeichnet jede Quelle, die nicht nach obiger Definition strukturiert ist, als unstrukturiert ([Sar08]). Lam et al. vermeiden den Begriff "unstrukturiert" völlig und sprechen statt dessen von "free text" in der gleichen Bedeutung ([LGM08]).

Oftmals wird jedoch noch eine weitere Unterscheidung der Quelle vorgenommen, in unstrukturierten und semi-strukturierten Text. Als *unstrukturiert* werden hierbei meist Texte einer natürlichen Sprache bezeichnet, wie beispielsweise in [CKGS06]. Besonders das Fachgebiet des NLP beschäftigt sich auch heute noch mit der Extraktion diverser Informationen aus derartigen Texten. HTML-Seiten beispielsweise werden dagegen oftmals als semi-strukturiert bezeichnet, da sie einer fest definierten Syntax gehorchen (je nach Version der HTML- oder XML-Syntax). Mittels des 'Document Object Model' (DOM) lässt sich darin eine klare, hierarchische Struktur erkennen, die genutzt werden kann, um

[4] In einem Gerichtsbeschluss vom 28. Mai 2009 hat das Hamburger Oberlandesgericht "eine wegweisende Entscheidung zum 'Screen Scraping' getroffen", indem der Klage des Flugunternehmens Ryanair statt gegeben wurde, welches gegen eine Website klagte, auf welcher die von Ryanair angebotenen Flugtickets weiterverkauft wurden, indem einem Besucher die entsprechenden, mittels 'Screen Scraping' von der Ryanair-Website gewonnenen Informationen präsentiert wurden.
Quelle: `http://www.silicon.de/lifestyle/web/0,39038976,41500978,00/gericht+verbiete t+_screen+scraping.htm`

die Position relevanter Informationen zu ermitteln. Auch hier ist eine weitere Unterscheidung zwischen manuell oder mittels eines Templates erstellten HTML-Seiten möglich, da letztere i.d.R. einer klareren und einheitlicheren Struktur folgen ([CKGS06]).

Eine exakte Grenze zwischen unstrukturierten und semi-strukturierten Daten kann jedoch nicht gefunden werden und ist stets von der jeweiligen Sichtweise abhängig. Wir werden in der vorliegenden Arbeit der Literatur folgend stets von *strukturierten* Daten sprechen, wenn deren spezielle Semantik klar und eindeutig erkennbar ist. Da wir uns nicht mit in natürlicher Sprache verfassten Texten beschäftigen, werden wir Sarawagis Ansatz folgend sämtliche Texte, die nicht in strukturierter Form vorliegen, als *unstrukturiert* bezeichnen und keine weitere Unterscheidung in semi-strukturierte Texte vornehmen, wohl wissend, dass der Grad der Unstrukturiertheit stark variieren kann.

Doch auch der Übergang zwischen strukturierten und unstrukturierten Daten ist oftmals fließend. XML-Dokumente, wie beispielsweise die XML-Version der DBLP[5], weisen oftmals strukturierte Daten auf. XHTML als eine Anwendung der XML weist jedoch Daten, die sich beispielsweise innerhalb einer Tabellenzeile befinden, noch lange keine Semantik zu; daher handelt es sich in diesem Falle nach obiger Definition um unstrukturierte Daten (z.B. "`<td>Hans Wurst</td>`"). Wurde aber beispielsweise CSS (Cascading Style Sheets, eine Technik, um das Layout einer HTML-/XHTML-Seite zu gestalten ohne Einfluss auf deren Inhalt zu nehmen) in der Art benutzt, dass jedem semantischen Attribut ein entsprechender Klassenname zugeordnet wurde (beispielsweise in "`<td class="author">Hans Wurst</td>`"), so handelt es sich im engeren Sinne wiederum um strukturierte Daten. Wir werden im Kontext von XHTML-/HTML-Seiten jedoch stets von unstrukturierten Daten sprechen.

1.5 Dimensionen der IE

In einer Studie aus dem Jahre 2008 kategorisiert Sarawagi das weite Gebiet der Informationsextraktion in die folgenden fünf Dimensionen ([Sar08]):

Typ der zu extrahierenden Strukturen Hier kann es sich um einzelne Entitäten handeln (beispielsweise Personen- oder Firmennamen), Beziehungen der Entitäten untereinander (z.B. "Person X leitet Firma Y"), einzelne Attribute, oder ganze Listen oder Tabellen.

Typ der unstrukturierten Quelle Dies können beispielsweise kurze Strings oder umfangreiche Dokumente sein, die frei oder unter Zuhilfenahme von Templates erstellt wurden.

Typ der Eingaberesource zur Extraktion Verschiedene Typen sind denkbar, wie beispielsweise strukturierte Datenbanken, markierter unstrukturierter Text oder linguistische Tags.

[5] `http://dblp.uni-trier.de/xml`

Extraktionsmethode Hier werden zwei grundlegende Methoden der Extraktion unterschieden, regelbasiert und statistisch. Wir werden uns in Abschnitt 1.6 genauer mit der von uns gewählten Methode, der regelbasierten Extraktion, auseinander setzen.

Ausgabe der Extraktion Die Ausgabe kann als annotierter Text oder in Form einer Datenbank erfolgen.

Die im Rahmen dieser Arbeit erstellte Extraktionssoftware kann nach dieser Kategorisierung spezifiziert werden. Wir werden uns um die Extraktion von Attributen aus Dokumenten bemühen, welche in den meisten Fällen mittels Templates erstellt wurden. Der Typ der Ressource sind somit HTML-Seiten. Dabei werden wir stets regelbasiert vorgehen und die Ergebnisse in strukturierter Form ausgeben.

Chang et al. untersuchen verschiedene Tools speziell zur regelbasierten Web-IE und definieren hierzu insgesamt drei Dimensionen ([CKGS06]):

Schwierigkeitsgrad Je nach Strukturierung der Quelle variiert die Schwierigkeit der Informationsextraktion. Bei strukturierten Quellen ist diese Aufgabe nahezu trivial, während ihre Schwierigkeit bei zunehmender Unstrukturiertheit wächst. Auch zahlreiche andere Faktoren können den Schwierigkeitsgrad bestimmen, beispielsweise der Umgang mit fehlenden Attributen oder Attributen, die gleich mehrere Werte besitzen.

Extraktionstechnik Hier unterscheidet man, in welcher Form die Quelle vorliegt (meist als String) und entsprechend bearbeitet wird, sowie die verschiedenen Herangehensweisen an die Problematik der Extraktion. Es gibt bottom-up- oder top-down-Ansätze, Techniken der Mustererkennung oder der logischen Programmierung, die Nutzung regulärer Grammatiken oder logischer Regeln. Zudem wird unterschieden, ob ein Dokument während des Extraktionsvorgangs nur einmal oder mehrmals bearbeitet werden muss.

Automatisierungsgrad Gerade im Bereich lernender IE-Systeme (siehe Abschnitt 1.6) kann hier unterschieden werden, ob bzw. wie viele Trainingsbeispiele manuell erstellt oder angewendet werden müssen. Manche IE-Systeme fungieren als so genannte *Crawler* bzw. *Robots* und durchsuchen das WWW selbst nach Informationen, während bei anderen die entsprechenden URLs der zu bearbeitenden Websites angegeben werden müssen.

Der Schwierigkeitsgrad der Extraktion, den unsere Software bewältigen muss, ist wegen der Bearbeitung unstrukturierter und meist mittels Templates erstellter HTML-Seiten im mittleren Bereich anzusiedeln. Fehlende Attribute müssen in geeigneter Weise behandelt werden. Zur Extraktion werden regelbasierte Techniken der Mustererkennung (pattern matching) auf Grundlage regulärer Ausdrücke genutzt. Jedes Dokument muss stets nur einmal bearbeitet werden, doch hin und wieder wird es nötig sein, weitere Dokumente zur Gewinnung spezieller Informationen zu untersuchen. Der Grad der Automatisierung ist recht gering, da wir die Extraktionsregeln allesamt manuell erstellen werden (vgl. Abschnitt 1.6.1).

1.6 Regelbasierte IE: Wrapper

Wie bereits erwähnt lassen sich grundsätzlich zwei Extraktionsmethoden unterscheiden: regelbasiert oder statistisch. Statistische Extraktionssysteme gliedern sich weiterhin in zwei Untergruppen auf, jene, die auf dem 'Hidden Markov Modell' beruhen und jene, die sich auf maximale Entropie stützen. Einen guten Überblick über jenes Thema liefert Kapitel 3 in [Sar08].

Wir werden uns in der vorliegenden Arbeit stets mit regelbasierten Extraktionsmethoden beschäftigen. Extraktionssoftware, die regelbasiert vorgeht, wird in der Literatur allgemein als *Wrapper* bezeichnet.

1.6.1 Generierung von Wrappern

Wrapper extrahieren die gewünschten Informationen also stets anhand fest definierter "Extraktionsregeln". Hierbei existieren zwei klassische Strategien: Solche, die auf Mustererkennung (pattern matching) beruhen und solche, die mittels Grammatiken aus dem Forschungsgebiet der 'Formalen Sprachen' arbeiten. Beide Ansätze besitzen Vor- und Nachteile, und in der Praxis findet man oftmals Kombinationen dieser beiden Strategien, wie beispielsweise in [HFAN98] beschrieben.

Zudem unterscheidet man Wrapper bezüglich der Art, wie diese generiert werden: *manuell* (hand-coded) oder *lernend* (learning-based). In den Anfängen der IE wurden Extraktionssysteme meist mittels manuell codierter Regeln entwickelt. Diese Systeme benötigen stets 'Experten', die sich sowohl mit der Programmierung der Software als auch mit den domainspezifischen Besonderheiten der zu extrahierenden Daten auskennen, um robuste Extraktionsregeln aufstellen zu können ([Sar08]). Zudem lassen sich von Hand codierte Wrapper stets nur auf eine Art von Daten anwenden, diejenigen, für welche ihre Regeln aufgestellt wurden. Es ist nicht ohne Weiteres möglich, einen solchen Wrapper auf andere Quellen als die zu Beginn geplanten anzuwenden, weshalb die manuelle Codierung in höchstem Maße zeitaufwendig ist ([LGM08]).

Daher wurden schon bald lernende Systeme entwickelt, die lediglich eine Reihe so genannter *Trainingsdaten* benötigen, mit deren Hilfe die Software eigene Regeln erstellt. Solche Systeme können auch von Personen zufriedenstellend bedient werden, die mit den technischen Details der Software weniger vertraut sind. Eine Kenntnis der domainspezifischen Daten ist jedoch auch hier erforderlich ([Sar08]). Durch das Hinzufügen weiterer Testdaten können lernende Systeme recht schnell an neue Gegebenheiten angepasst werden und zeichnen sich daher durch eine weit höhere Flexibilität aus als manuell codierte Systeme ([LGM08]).

Im Bereich der lernenden Wrapper ist derzeit in Bezug auf das WWW vor allem die Technik der "Wrapper induction" ([Kus00]), bei welcher Informationen über die Struktur einer Quelle mittels Trainingsseiten erfasst und somit 'gelernt' werden, weit verbreitet. Bestehende Systeme unterscheiden sich hier in der Art, wie die Regeln generiert werden. Ältere Systeme wie beispielsweise WIEN ("Wrapper Induction ENvironment", [KWD97]) oder STALKER

([MMK99]) benötigen eine Reihe manuell beschrifteter Testseiten, um daraus allgemeine Regeln formulieren zu können.

Neuere Systeme nutzen statt dessen interaktive Eingabemechanismen, durch welche ein Benutzer mit minimalem Aufwand während des Extraktionsvorgangs benötigte Informationen direkt in visuelle Interfaces eingibt. Das System *Lixto* beispielsweise generiert Extraktionsregeln in einer eigens konzipierten, logischen Sprache *Elog*, indem der Benutzer entsprechende Markierungen in einer visuellen Benutzeroberfläche vornimmt und somit einzelnen Bereichen einer Webseite eine Semantik zuweist ([BFG01]). In einer von Irmak und Suel entwickelten Software deklariert der Benutzer so genannte *training tupels* durch Markierung mittels der Maus direkt im Webbrowser ([IS06]).

Arasu und Garcia-Molina nutzen dagegen die im Webbereich, beispielsweise bei Produkten des Versandhauses Amazon[6] verwendeten Templates aus, indem die entsprechenden Webseiten dahin gehend untersucht werden, welche Bereich zum Template gehören und in welchen Bereichen wechselnde Daten vorhanden sind ([AGM03]). Dieser Ansatz kommt daher völlig *ohne* entsprechende, vom Benutzer festzulegende Trainingsdaten aus. Das gleiche Ziel verfolgen auch Papadakis et al. sowie Vuong und Gao, indem sie eine Webseite nach gleichartigen Strukturen durchsuchen und diese zu Clustern zusammenfassen, wodurch relevante Datenregionen ausfindig gemacht werden können ([PSRV05] und [VG07]).

Trotz der o.g. klaren Vorteile, die lernende Wrapper besitzen, werden wir dennoch in der vorliegenden Arbeit zunächst stets manuell codierte Regeln nutzen. Dies liegt darin begründet, dass unser Ziel nicht die allgemeine Lösung eines weit gefassten Extraktionsproblems, sondern die spezielle Lösung einer äußerst eng gefassten Aufgabe, der Extraktion bibliographischer Informationen für DBLP ist. Da die Informationsquellen in diesem Zusammenhang stark begrenzt sind, ist der Mehraufwand der Programmierung eines lernenden Wrappers nicht nötig, um dennoch die erforderlichen Ergebnisse zu erzielen. In einer Studie der für diese Zwecke relevanten Extraktionsquellen (siehe Kapitel 3) werden wir zudem sehen, dass einige unserer Datenquellen streng von der Konvention anderer Server abweichen. Ein lernender Wrapper müsste hier äußerst flexibel vorgehen, während unsere manuell codierte Lösung speziell auf die jeweiligen Anforderungen eingehen kann. Zudem muss eine Navigation innerhalb der Websites erfolgen, um gezielt einzelne Hefte oder Bände einer Publikation zu erfassen, was in manuell codierter Form ebenfalls einfach und effektiv umgesetzt werden kann. DBLP strebt ohnehin keine völlig automatisierte Gewinnung von Daten an, denn der Hauptaugenmerk liegt auf deren Qualität, weshalb vielfältig manuelle Bearbeitung der neu gewonnenen Daten eingesetzt wird (vgl. Kapitel 2). Ziel ist es, jene Arbeit zu vereinfachen und wiederkehrende Aufgaben zu einem gewissen Teil zu automatisieren. Ein Wrapper mit manuell gepflegtem Regelsatz bietet hier den idealen Kompromiss aus Aufwand und Nutzen.

Bei der Fusion einer strukturierten Quelle mit einer Konferenz-Website in HTML-Format (siehe Kapitel 10.1) werden wir dagegen einen kleinen Schritt in die Richtung lernender Systeme gehen. Dort werden wir, ähnlich der in [AGM03] beschriebenen Vorgehensweise (Identifikation von Templates innerhalb der Websites von 'Amazon'), Regeln zur Extraktion der gewünschten Daten anhand automatisch generierter Trainingsdaten erzeugen.

[6] `http://www.amazon.com`

1.6.2 Klassifizierungen der Wrapper

Je nachdem, mit welcher Strategie Wrapper bei der Suche nach relevanten Informationen in unstrukturierten Quellen vorgehen, lassen sich diese in verschiedene Klassen unterteilen, welche beispielsweise in [KWD97] und [Kus00] exakt beschrieben und bezüglich ihrer Komplexität miteinander verglichen werden. An dieser Stelle soll auf die formalen Definitionen, die obigen Artikeln entnommen werden können, verzichtet werden. Es wird jedoch recht hilfreich sein, die dortigen Klassifizierungen zu kennen, um somit die Herangehensweise an das Problem regelbasierter IE besser erfassen zu können.

LR-Wrapper Die simpelste Form eines Wrappers ist der so genannte LR-Wrapper. Bei dieser Art des Wrappers werden lediglich Paare von links- (L) und rechtsseitigen (R) Begrenzern angegeben, mittels derer die zu ermittelnden Attribute innerhalb einer Seite identifiziert werden können.

Betrachten wir das folgende, beispielhafte HTML-Dokument:

```
<html><head><title>Beispielseite</title></head>
<body>
    <h1>Beispielseite</h1>
    Zu Beginn einige unwesentliche Informationen...
    <h2>Interessante, bibliographische Daten:</h2>
    <strong>Titel 1</strong> (<em>Autoren 1</em>)<br />
    <strong>Titel 2</strong> (<em>Autoren 2</em>)<br />
    <strong>Titel 3</strong> (<em>Autoren 3</em>)<br />
    <hr />
    Hier folgen wieder uninteressante Informationen...
</body>
</html>
```

Um die relevanten Titel- und Autoreninformationen zu extrahieren, könnte ein LR-Wrapper definiert werden, welcher die Begrenzer (L = "`<strong>`") und (R = "`</strong>`") zur Extraktion der Titel, sowie (L = "`(<em>`") und (L = "`</em>)`") zur Extraktion der jeweils zugehörigen Autoreninformationen verwendet, um die gewünschten Titel/Autoren-Paare (`Titel 1`/`Autoren 1`), (`Titel 2`/`Autoren 2`) und (`Titel 3`/`Autoren 3`) zu erhalten.

HLRT-Wrapper Ein HLRT-Wrapper ist die Erweiterung eines LR-Wrappers. Bei diesem wird zunächst der Teil, in welchem der relevante Inhalt zu finden ist, vom Rest des Dokuments separiert, indem zwei Begrenzer angegeben werden, die den *head* (H) und *tail* (T) identifizieren. Diese Bereiche werden ignoriert, während der übrig bleibende *body* gemäß einem LR-Wrapper bearbeitet wird. Dies verhindert, dass irrelevante Informationen im Ergebnis erscheinen. Lee und Geierhos bezeichnen einen solchen *body* auch als 'minimal data region', d.h. die kleinste

Region (innerhalb eines HTML-Dokuments), in welcher sich die relevanten Daten befinden ([LG09]).

Stellen wir uns das obige Beispiel mit einer modifizierten vierten Zeile vor:

```
Zu <strong>Beginn</strong> einige <em>unwesentliche</em> Informationen...
```

Der oben definierte LR-Wrapper würde auch diese Zeile bearbeiten und ein völlig sinnloses Titel/Autoren-Paar (`Beginn`/`unwesentliche`) erhalten. Um dies zu vermeiden wären (H = "`</h2>`") und (T = "`<hr/>`") mögliche Trennsymbole eines HLRT-Wrappers. Die Seite würde anhand der entsprechenden Begrenzer unterteilt, so dass der *body* nur die relevanten Informationen enthielte.

OCLR-Wrapper Ein OCLR-Wrapper stellt ebenfalls eine Erweiterung des LR-Wrappers dar. Auch bei diesem werden zusätzliche Begrenzer eingeführt, die den Anfang (*opening*, O) und das Ende (*closing*, C) eines jeden relevanten Tupels markieren.

Stellen wir uns obiges Beispiel erneut in etwas modifizierter Form vor, indem die Zeilen mit den relevanten Daten wie folgt aussähen:

```
<strong>1.</strong> - <strong>Titel 1</strong> (<em>Autoren 1</em>)<br />
<strong>2.</strong> - <strong>Titel 1</strong> (<em>Autoren 1</em>)<br />
<strong>3.</strong> - <strong>Titel 1</strong> (<em>Autoren 1</em>)<br />
```

Unsere bisher definierten Wrapper würden jeweils nach dem ersten Auftreten eines `<strong>`-Tags suchen und stets an der falschen Stelle beginnen. Durch Definition eines OCLR-Wrappers mit (O = "`</strong>`" und beispielsweise (C = "`<br />`") könnte dies im Beispiel vermieden werden, da die Suche nach den Attributen jedes einzelnen Tupels erst jeweils hinter dem ersten `</strong>`-Tag beginnen würde.

HOCLRT-Wrapper Diese Art von Wrapper kombiniert die Vorteile der beiden zuletzt vorgestellten Wrapper. Zunächst wird wie bei einem HLRT-Wrapper ein *body* identifiziert, der dann im Sinne eines OCLR-Wrappers bearbeitet wird.

N-LR- und N-HLRT-Wrapper Allen bisher vorgestellten Wrapper-Sorten ist gleich, dass sie lediglich dann Daten extrahieren können, wenn diese in einem listenartigen Aufbau vorliegen, d.h. ein neuer Datensatz erst nach dem Ende des vorherigen beginnt. Es ist jedoch auch möglich, dass Daten ineinander verschachtelt (engl: *nested*, daher das Präfix "N-") sind. In solchen Fällen ist es möglich, Wrapper zu konstruieren, die nach dem Prinzip eines LR- bzw. HLRT-Wrappers arbeiten, jedoch die verschachtelten Datensätze korrekt extrahieren können.

Kapitel 2

Bibliographische Daten in DBLP

Ziel der vorliegenden Arbeit ist es, bibliographische Informationen für DBLP zunächst zu extrahieren und später (ab Kapitel 5) zu fusionieren. Daher ist es an dieser Stelle zunächst notwendig, einen Blick auf DBLP (Abschnitt 2.1), die dort repräsentierten Daten sowie deren interne Darstellung zu werfen (Abschnitt 2.2). Zudem müssen wir festlegen, was wir in diesem Kontext unter 'Datenqualität' verstehen wollen, um vor allem bei der Fusion entscheiden zu können, wann ein Datensatz 'besser' ist als ein anderer (Abschnitt 2.3). Am Ende dieses Kapitels (Abschnitt 2.4) werden wir ein an die Implementierung der beiliegenden Software angepasstes Datenmodell definieren, welches wir im weiteren Verlauf dieser Arbeit zur Beschreibung diverser Sachverhalte nutzen können.

2.1 DBLP

Die digitale Bibliothek DBLP[1] wurde im Jahre 1993 von Michael Ley als simpler Test der damals aktuellen Webtechnologie erschaffen. Anfänglich bestand sie lediglich aus einigen Inhaltsverzeichnissen (Tables of Contents, TOCs) wichtiger Konferenzberichte und Zeitschriften aus dem Bereich der Datenbanksysteme und Logikprogrammierung, was auch zu dem Akronym DBLP ("**D**ata **B**ases and **L**ogic **P**rogramming") führte ([Ley02]). Im Verlauf der weiteren Jahre wurden immer weitere bibliographische Daten aller möglichen Bereiche der Informatik hinzugefügt, weshalb auch die Bedeutung des Akronyms nachträglich verändert wurde. Nach Angaben der DBLP-FAQ ([DBL09], "What is the meaning of "DBLP"?") sollte das Akronym nun als "**D**igital **B**ibliography & **L**ibrary **P**roject" gelesen oder schlicht als Eigenname akzeptiert werden.

Im Juni 2002 beinhaltete DBLP bereits über 286.000 Publikationen ([Ley02]), im Frühjahr 2008 wurde bereits der Millionste Eintrag gezählt. Aktuell (September 2009) sind laut [Ley09] bereits über 1,2 Millionen Artikel in DBLP enthalten, wobei die Tendenz stets steigend anzusehen ist. Heute stellt DBLP im Bereich der Informatik eine populäre und wichtige Quelle bibliographischer Informationen dar und wird von Wissenschaftlern auf der ganzen Welt genutzt.

[1] `http://dblp.uni-trier.de`

Die in DBLP erfassten Daten sind unter dem URL `http://dblp.uni-trier.de` online verfügbar. Diese Repräsentation der Daten werden wir im Folgenden auch als HTML-Version von DBLP bezeichnen. Alternativ liegt der gesamte Datenbestand auch als XML-Datei[2] vor, zu welcher eine entsprechende DTD[3] existiert. Sowohl die HTML- als auch die XML-Version werden täglich aus dem aktuellen internen Datenbestand von DBLP konstruiert ([Ley09]). Abbildung 2.1 zeigt ein Beispiel einer HTML-Seite sowie einen Ausschnitt aus der Datei `dblp.xml`.

Abb. 2.1: HTML- und XML-Version von DBLP: Links eine beispielhafte Übersichtsseite einer Konferenz, rechts ein Ausschnitt aus der `dblp.xml`
Quellen: `http://dblp.uni-trier.de/db/conf/sigsoft/fse2009.html`, `http://dblp.uni-trier.de/xml/dblp.xml`

2.2 Datenformate

Die Technik, welche sich hinter DBLP verbirgt, ist äußerst simpel gehalten, um einen minimalen Aufwand bei der Administration und der Portierung des Systems auf andere Plattformen zu gewährleisten. Die internen Datenformate, mit welchen DBLP arbeitet, sind zum einen in XML, zum anderen in einem HTML-ähnlichen Format, welches als BHT (Bibliography Hypertext) bezeichnet wird, gehalten:

> **"DBLP = bibliographic records + BHT-files"** ([Ley09]).

[2] `http://dblp.uni-trier.de/xml/dblp.xml`
[3] `http://dblp.uni-trier.de/db/about/dblp.dtd`

Da es sich bei all diesen Formaten um strukturierte Textdateien handelt, können die Daten sowohl automatisiert als auch manuell bearbeitet werden. Sowohl die Dateien der Ausgabeformate (HTML und XML) als auch die internen Dateien sind in reinem ASCII-Zeichensatz codiert, d.h. sie enthalten nur Zeichen mit Codes < 128. Einige benannte Entities für Umlaute, Ligaturen oder diverse andere Sonderzeichen, die dem ISO-8859-1-Standard (Latin-1) entsprechen, sind in der DBLP-DTD definiert. In wenigen Fällen werden auch numerische Entities (z.B. `&#231;`) zur Codierung besonderer Zeichen verwendet.

Da wir im weiteren Verlauf der vorliegenden Arbeit oftmals mit jenen internen Formaten arbeiten werden, ist es zunächst notwendig, diese genauer zu beschreiben.[4]

2.2.1 DBLP-Records

DBLP-Records (im Folgenden auch kurz als *records* bezeichnet) sind kleine Dateien in XML-konformer Syntax, welche den größten Teil der Information beinhalten. Die Elemente dieser XML-Struktur tragen Namen der in BibTeX[5] geläufigen Attribute und können demnach Informationen über Titel, Autoren, digital verfügbare Ressourcen etc. zu jeweils einer Publikationseinheit enthalten.[6] Eine der frühen Designentscheidungen bzgl. DBLP brachte jenes Format hervor, welches einfacher zu handhaben und vor allem zu parsen ist als tatsächliche BibTeX-Records. Dennoch ist eine Konvertierung von DBLP-Records in BibTeX-Records problemlos möglich.

Jedes *record* besitzt einen eindeutigen Schlüssel (`key`), der Pfad und Dateinamen innerhalb des Verzeichnisbaumes von DBLP (`/dblp/publ`) entspricht. So ist beispielsweise das *record* mit `key="journals/pvldb/Ley09"` unter `/dblp/publ/journals/pvldb/Ley09` zu finden. Der Schlüssel wird bei Eintrag des Datensatzes in DBLP automatisch generiert und setzt sich im Wesentlichen aus der Publikationsform (`journals` oder `conf`), einem eindeutigen Bezeichner für die entsprechende Zeitschrift oder Konferenz (in obigem Beispiel `pvldb`), sowie einem eindeutigen Dateinamen – der aus den Namen des Autors, der Co-Autoren, dem Veröffentlichungsjahr sowie evtl. weiteren Zeichen zur Wahrung der Eindeutigkeit, generiert wird – zusammen.

Größter Unterschied zu BibTeX ist, dass innerhalb der *records* mehrere Autorennamen in einzelnen `<author>`-Tags untergebracht sind und nicht wie bei BibTeX in nur einem Feld. Die *records* werden daher in [Ley09] auch als "BibTeX records in XML syntax + ϵ" bezeichnet. In Abbildung 2.3 auf Seite 24 sind drei Beispiele für DBLP-Records zu finden.

[4] Sämtliche diesbezüglichen Informationen entstammen [Ley09] oder wurden in persönlichen Gesprächen mit Michael Ley erfragt.

[5] `http://www.bibtex.org`

[6] Eine Publikationseinheit kann beispielsweise ein einzelner Artikel einer Zeitschrift ('article') oder eines Konferenzbandes ('inproceedings') sein, aber auch eine übergeordnete Struktur ('proceedings'), durch welche eine komplette Zeitschriften-/Buchserie oder ein Konferenzband abgebildet wird.

Jedes *record* für sich ist, sofern es in `<dblp>`-Tags eingeschlossen wird, valide bzgl. der DBLP-DTD. Dies hat den Vorteil, dass zur Erstellung der Datei `dblp.xml` lediglich sämtliche *records* hintereinander aufgelistet werden müssen.

2.2.2 BHT

Wegen der starken Orientierung an BibTeX können nicht alle bibliographischen Informationen in den *records* untergebracht werden. Bände umfangreicher Konferenzen sind oftmals in einzelne 'Sessions' unterteilt, und in Zeitschriften werden oftmals thematische Gruppierungen der Artikel vorgenommen. Hieraus lassen sich Zwischenüberschriften ('section headers') gewinnen, die einem menschlichen Betrachter gerade bei großen Datenmengen von mehreren hundert Artikeln eine erhöhte Übersichtlichkeit bescheren. BibTeX liefert jedoch keine Möglichkeit, solche zusätzlichen Informationen innerhalb der Records unterzubringen, und somit liegen hier auch die Grenzen der DBLP-Records.

Um diese Missstände auszugleichen, greift DBLP auf ein simples Dateiformat zurück, welches dem klassischen HTML, erweitert um einige weitere Tags, entspricht. Dieses Format wird als "bibliography hypertext" (BHT) bezeichnet. Abbildung 2.3 auf Seite 24 zeigt Beispiele solcher BHT-Dateien.

Der Aufbau jener Dateien ist äußerst simpel. Bibliographische Daten, die innerhalb eines *records* abgebildet sind (oder werden sollen), stehen innerhalb von `<ul>`-Tags und beginnen stets mit `<li>`.[7] Alle anderen Informationen, für die es keine Entsprechung in BibTeX gibt, stehen außerhalb der `<ul>`-Tags.

Neue Daten, die in DBLP eingegeben werden, sollten in einem Format vorliegen, das dem endgültigen Format möglichst ähnlich ist ([DBL09], "What is the preferred format to enter publications into DBLP?"). Aus diesem Grund wurde zur Eingabe ebenfalls das BHT-Format gewählt.

Da das BHT-Format also sowohl zur internen Repräsentation der TOC-Seiten, als auch als Eingabeformat für neue Datensätze genutzt wird, wir oftmals aber genau spezifizieren wollen, welches dieser Formate gemeint ist, wollen wir hier eine klare Unterscheidung treffen.

$\mathsf{BHT}_{\mathsf{cite}}$ Die zuvor beschriebenen, internen BHT-Dateien unterscheiden sich von jenen BHT-Dateien, in welchen neue Daten erfasst werden, u.a. durch das Vorhandensein der `<cite>`-Tags. Diese Tags verweisen auf entsprechende *records*, indem sie deren Schlüssel beinhalten. Wir werden dieses Format daher im Folgenden als $\mathrm{BHT}_{\mathrm{cite}}$ bezeichnen. Die Dateien verfügen bereits

[7] Entgegen der XML-Konvention wird hier jedoch historisch bedingt kein End-Tag (`</li>`) erwartet. Dies war auch in alten Versionen von HTML nicht notwendig und wird von allen Browsern korrekt umgesetzt.

über einige zusätzliche Header- und Footer-Informationen, die denen der fertigen HTML-Seite gleichen (bzw. identisch mit diesen sind).

Eine BHT_{cite}-Datei stellt also lediglich ein 'Gerüst' dar, mit dessen Hilfe eine HTML-Seite generiert werden kann. Dieses interne Format wird uns erst im Bereich der Fusion (Szenario F-2' in Kapitel 6.1.3) begegnen.

$\text{BHT}_{\text{c/j}}$ Im Gegensatz zu den o.g. BHT_{cite}-Dateien werden wir auch BHT-Dateien nutzen, um neue Daten in den Bestand von DBLP einzufügen. Diesen fehlen i.d.R. noch Header und Footer, und sie tragen vor allem noch sämtliche Informationen der Artikel in sich. Beim Eintrag der Daten in DBLP werden jene Dateien aufgespalten; ein Vorgang, der im folgenden Abschnitt 2.2.3 beschrieben wird. Wie bereits erwähnt stehen sämtliche Informationen, die bei diesem Prozess in *records* verwandelt werden, innerhalb von `<ul>`-Tags, alle anderen Informationen, insbesondere evtl. vorhandene Zwischenüberschriften, außerhalb.

Bei jenen Daten außerhalb der `<ul>`-Tags existieren geringfügige Unterschiede, je nachdem, ob ein *journal* oder eine *conference* erfasst werden soll:

Bei Zeitschriften, Zeitungen, Newslettern und anderen Publikationsformen, die eine klare Aufteilung in Volumes (und evtl. Issues) besitzen, werden eben jene Informationen, sowie – falls bekannt – Erscheinungsmonat und -jahr, als Zwischenüberschrift publiziert. All diese Informationen stehen innerhalb von `<h2>`-Tags. Dies hat den Vorteil, dass auf der resultierenden HTML-Seite eine klare Unterteilung der einzelnen Bände und Hefte sichtbar ist. Können innerhalb der Artikel weitere Zwischenüberschriften identifiziert werden, so sind diese zwischen den `<ul>`-Tags der einzelnen Artikel ohne eine entsprechende Auszeichnung (markup) zu finden. Diese Form einer BHT-Datei soll aus Gründen der Unterscheidbarkeit fortan als BHT_{j} bezeichnet werden.

Konferenzbände, einzelne Bücher und andere, keiner strengen Reglementierung bzgl. eines Volumes unterworfene Publikationen werden dagegen in einem geringfügig anderen Format erfasst, welches wir daher mit BHT_{c} bezeichnen werden. Diese Dateien unterscheiden sich von denen der *journals* darin, dass ihre Zwischenüberschriften beliebige Zeichenketten beinhalten dürfen. Diese sind ebenfalls in `<h2>`-Tags zu finden. Existieren Unterüberschriften, so können diese mittels `<h3>`, `<h4>` oder gar `<h5>`-Tags weiter verschachtelt werden. Zudem enthält der Kopfbereich jener Dateien oftmals zusätzliche Informationen zu einer Konferenz, die in eben jener Form in die HTML-Seiten einfließen.

Im Folgenden werden wir die zu erfassenden Publikationen stets als *journal*, Journal oder Zeitschrift bezeichnen, wenn die resultierende BHT-Datei vom Typ BHT_{j} ist. Hierunter können demnach auch Zeitungen, Newsletter oder Buchserien fallen. Analog nennen wir solche Publikationen, die in einer BHT_{c}-Datei erfasst werden, *conference* bzw. Konferenz. Spielt die Art der Datei keine Rolle, so werden einfach von einer $\text{BHT}_{\text{c/j}}$-Datei – in der Bedeutung von "eine Datei im BHT_{c}- oder BHT_{j}-Format" – sprechen, um sie lediglich vom internen BHT_{cite}-Format abzugrenzen.

Obige Einteilung in Zeitschriften und Konferenzen trifft zwar in den meisten Fällen zu, ist aber keineswegs verpflichtend. Die LNCS-Serie von Springer (vgl. Kapitel 3.2.10) ist eine Buchserie, deren Bände fortlaufende Volume-Nummern tragen. Dennoch werden die entsprechenden bibliographischen Daten in BHT_c-Dateien erfasst, da es sich meist um Publikationen einer festen Konferenz handelt.

2.2.3 Zusammenspiel der DBLP-Dateiformate

Neue Daten, die in DBLP eingetragen werden sollen, werden zunächst manuell oder mit Hilfe eines entsprechenden Wrappers in $BHT_{c/j}$-Dateien gesammelt. Um die Daten einer solchen Datei in DBLP aufzunehmen, müssen lediglich die einzelnen Artikelinformationen in *records* umgewandelt werden. Bei diesem Prozess wird der eindeutige 'key' erzeugt, welcher i.d.R. dem Pfad innerhalb des Verzeichnisbaumes von DBLP entspricht. In der Eingabe-BHT-Datei wird an der Stelle, an welcher die Informationen standen, ein entsprechendes `<cite>`-Tag eingefügt, dessen `key`-Attribut eben jenem Schlüssel entspricht. Auf diese Weise werden also aus der ursprünglichen $BHT_{c/j}$-Datei eine BHT_{cite}-Datei sowie einzelne *records* generiert.

Zur Erstellung der TOC-Seiten werden nun eben jene Informationen wieder miteinander verknüpft und im Stile von SSI[8] in einem täglichen Update-Prozess in reines HTML verwandelt. Dabei werden die speziellen BHT-Tags (`<cite>`, `<footer>`, ···) ersetzt, während das grundsätzliche Gerüst der Seite (die `<h2>`, `<h3>`,...-Tags sowie die `<ul>`/`<li>`-Elemente) beibehalten werden kann.

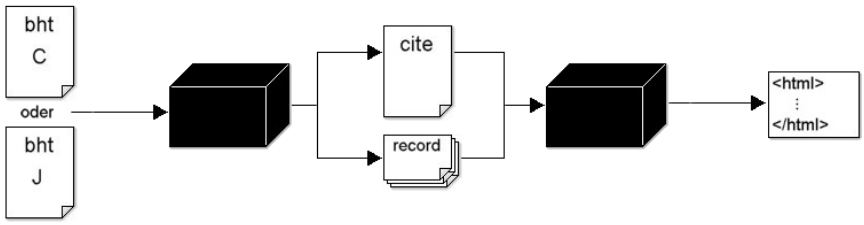


Abb. 2.2: Schematische Darstellung des Zusammenspiels der Datenformate in DBLP: Eine BHT_j-Datei wird in *records* sowie eine BHT_{cite}-Datei aufgespalten. Anschließend können diese Daten zu einer HTML-Seite kombiniert werden.

Abbildung 2.2 zeigt diese Vorgänge schematisch und in vereinfachter Weise. Die schwarzen Kästen (black boxes) deuten Vorgänge an, die mittels existierender Software ablaufen und deren interne Vorgehensweise für uns irrelevant ist. Abbildung 2.3 auf der folgenden Seite zeigt beispielhaft bibliographische Daten in einer BHT_j-Datei, deren Aufspaltung in BHT_{cite} und *records*, sowie die daraus resultierende HTML-Seite.

[8] "Server Side Includes", eine Technik, mittels der Webserver dynamische HTML-Seiten generieren.

Abb. 2.3: Beispiel des Zusammenspiels der Datenformate in DBLP: Oben eine BHT_j-Datei der Daten, die in DBLP aufgenommen werden sollen. Diese wird in eine BHT_{cite}-Datei (Mitte links) und *records* (Mitte rechts) aufgespalten. Die Daten werden anschließend zu einer HTML-Seite zusammen gefügt (unten, `http://dblp.uni-trier.de/db/journals/jfp/jfp14.html`).

2.3 Datenqualität

In der Literatur werden vielfältige Dimensionen der Datenqualität definiert, doch existiert hier kein festgeschriebener Standard. Je nach Anwendungsgebiet kann die Definition von Datenqualität stark variieren ([SMB05]). Einen guten Überblick über dieses Thema sowie einen direkten Bezug auf DBLP liefert [Reu07] (Kapitel 2, Seiten 9-30).

In [LR06] wird, ebenfalls im Kontext von DBLP, zwischen "process driven" (prozessorientiertem) und "data driven" (datenorientiertem) Qualitätsmanagement unterschieden.

Zu erstgenannter Kategorie, dem "*Prozess Driven Quality Management*", zählen Aspekte wie die Ausgewogenheit der Daten oder deren zeitnahe Erfassung (timeliness). Um eine Ausgewogenheit zu gewährleisten werden grundsätzlich nur vollständige Hefte einer Zeitschrift oder komplette Konferenzbände in den Datenbestand von DBLP aufgenommen. Hierdurch verringert sich die Gefahr einer Bevorzugung einzelner Personen. 'Timeliness' dagegen stellt vor allem wegen der äußerst knappen menschlichen Ressourcen ein Problem dar. Hier kann eine enorme Qualitätssteigerung durch die automatische Extraktion der bibliographischen Daten erreicht werden, mit welcher wir uns in Kapitel 4 auseinander setzen werden.

"*Data Driven Quality Management*" bezieht sich dagegen auf inhaltliche Aspekte der Daten wie beispielsweise deren Vollständigkeit oder Fehlerfreiheit. Hier nennen Ley und Reuther zwei Typen von Strategien: "data edits" und "database bashing" ([LR06]). Strategien der ersten Art (*data edits*) zeichnen sich dadurch aus, dass spezielle Regeln (*business rules*) auf den Datenbestand angewendet werden, um bestimmte Datenkonstellationen zu entdecken. In DBLP wird hier beispielsweise 'Personal Name Matching' innerhalb der Coautor-Netzwerke betrieben, um auf diese Weise Synonyme und Homonyme (vgl. Abschnitt 2.3.2) im Datenbestand aufzufinden ([RWL+06] und [Reu07]). Bei den *database bashing*-Strategien dagegen werden bestehende Daten mit solchen anderer Quellen verglichen, um auf diese Weise Datensätze zu vervollständigen oder Fehler zu korrigieren. Eben diesen Ansatz werden wir bei der Fusion bestehender Datensätze mit einer zweiten Quelle (Kapitel 6 bis 10) verfolgen.

Zunächst ist es jedoch erforderlich, exakt festzulegen, was wir in Bezug auf unsere bibliographischen Daten unter "guter Qualität" verstehen wollen. Dabei werden wir nur solche Aspekte betrachten, die rein syntaktisch entscheidbar sind, um eine Umsetzung in die Software zu ermöglichen.

2.3.1 Titel

Die Qualität eines Titels zeichnet sich dadurch aus, dass dieser vollständig und korrekt ist. Dies lässt sich rein syntaktisch jedoch nicht beurteilen, was eine Aussage über die Qualität eines Titels äußerst erschwert. Kryptische Symbole und Befehlssequenzen, die oftmals von einer direkten Kopie aus TEX/LATEX resultieren, sollten jedoch nach Möglichkeit entfernt werden. Zudem sollte der Titel sowohl Groß- als auch Kleinbuchstaben enthalten. Eine spezielle

Anforderung von DBLP, die jedoch weniger mit Qualität denn mit interner Repräsentation der Daten zusammenhängt, ist die, dass jeder Titel mit einem Interpunktionszeichen (Punkt, Ausrufezeichen oder Fragezeichen) enden muss.

2.3.2 Autorennamen

Bei den Autorennamen spielt die Qualität eine besonders große Rolle. Ein wichtiges Ziel von DBLP ist es, möglichst vollständige Autorennamen zu gewinnen, um entsprechende reale Personen voneinander abgrenzen, oder Publikationen ein und derselben Person zuordnen zu können. Im erstgenannten Fall handelt es sich um Homonyme, d.h. gleiche Namen, die unterschiedliche reale Personen bezeichnen, während der zweite Fall Synonyme behandelt, also unterschiedliche Namen, hinter welchen sich ein und dieselbe Person verbirgt. Eine ausführliche Behandlung dieser Problematik in Bezug auf DBLP kann in [Reu07] gefunden werden.

Die Namen sollten vollständig ausgeschrieben sein, d.h. bei Autoren, die mehrere Vornamen besitzen, sollen diese komplett vorhanden und nicht abgekürzt sein. Oftmals sind jedoch, gerade bei Zweit- und Drittnamen lediglich Initialen (also – meist – einbuchstabige Abkürzungen des Vornamen, z.B. “H.” statt “Heinz”) angegeben. Ebenso wie die Titel sollten die Autorennamen nicht ausschließlich aus Großbuchstaben bestehen und müssen evtl. nach einfachen Regeln in normale Groß- und Kleinschrift konvertiert werden – was sich bei den Namen im Gegensatz zu den Titeln als recht einfach erweist, da hier global gültige Regeln definiert werden können.

Alle Namen sollten der europäischen Konvention folgend in der Reihenfolge ‘Vorname(n) Nachname’ vorliegen, doch ist eine eindeutige Zuordnung der Namensteile – gerade bei Namen aus dem asiatischen Raum – oftmals nicht möglich (vgl. [Ley09]). Ein einzelner Autorname sollte zudem kein Komma enthalten, da dieses Satzzeichen zur Abgrenzung einzelnen Autorennamen innerhalb einer Liste benutzt wird.

2.3.3 Seitenangaben

Als Seitenangabe wird, wie es bei wissenschaftlichen Veröffentlichungen innerhalb von Zeitschriften und Konferenzbänden üblich ist, ein Seitenrang ‘von - bis’ erwartet (also z.B. “3-11”). Eine einzelne Seitenzahl ist nur dann erwünscht, wenn der Artikel lediglich eine einzelne Seite belegt. Ist bekannt, dass ein Artikel sich über mehrere Seiten erstreckt, jedoch nicht auf welcher Seite er endet, so wird dies ebenfalls in den Seitenangaben kenntlich gemacht – nach derzeitiger Konvention mittels eines nachgestellten Minuszeichens, also z.B. “3-”. Ist keine Angabe bekannt, so darf dieses Feld auch leer gelassen werden; die derzeitige Konvention hält hierfür das Symbol ‘0-’ bereit.

Hin und wieder ist im Datenbestand von DBLP auch zu erkennen, dass im Feld *pages* keine tatsächliche Seitenangabe, sondern lediglich eine vom Verlag publizierte Artikelnummer ein-

getragen wurde[9]. Ebenso ist in manchen Fällen lediglich die Anzahl der Seiten, über welche sich ein Artikel erstreckt, angegeben. Diese beiden Fälle sind zwar zulässig, zeugen jedoch von einer geringeren Datenqualität als die Angabe der tatsächlichen Seitennummern.

2.3.4 Electronic Edition (EE)

Das Element EE (Electronic Edition) ist optional, jedoch zeugt sein Vorhandensein von höherer Qualität des Datensatzes. Hier soll ein Verweis auf eine Quelle im WWW angegeben werden, in welcher nähere Informationen zum entsprechenden Artikel verfügbar sind. Dabei handelt es sich i.d.R. um eine Abstract-Seite.

In älteren Datensätzen fehlt dieser Eintrag oftmals völlig, oder es wurde ein URL angegeben, der evtl. – bedingt durch eine Umstrukturierung des Servers oder einen Domainwechsel – nicht mehr gültig ist. In neueren Datensätzen werden an dieser Stelle daher grundsätzlich DOI-Links bevorzugt.

Ein Digital Object Identifier (DOI) ist ein global eindeutiger Bezeichner zur Identifikation jeglicher Art "geistigen Eigentums". Ähnlich einer ISBN, mittels welcher ein Buch eindeutig identifiziert werden kann, werden DOIs hauptsächlich im Bereich des WWW verwendet. Ein DOI hat eine feste Syntax und besteht aus einem Präfix und einem Suffix, welche durch einen Querstrich (Slash) voneinander getrennt werden, beispielsweise `10.1000/186`. Der Präfix wird hierbei von der 'International DOI Foundation' (IDF) zentral vergeben, während die Gestaltung des Suffixes in der Hand des jeweiligen Verlegers liegt – ähnlich der Handhabung des Domain- und Localparts eines URL. Dabei sind dem jeweiligen Verleger keinerlei Vorgaben gemacht, in welcher Art und Weise er den Suffix zu wählen hat. Er kann ein eigenes Benennungsschema kreieren, oder auf ein evtl. bereits bestehendes Schema (z.B. eine ISBN) zurückgreifen.

Mit Hilfe von DOIs ist es möglich, die entsprechende digitale Ressource direkt im WWW aufzufinden, indem man diese entweder in den DOI-Resolver einträgt, der unter dem URL `http://dx.doi.org/` erreichbar ist, oder indem man den URL eben jenes Proxyservers voranstellt. Bei dem im obigen Beispiel genannten DOI handelt es sich beispielsweise um den DOI des "DOI-Handbuches", dem die in diesem Abschnitt vermittelten Informationen entnommen sind, und welches man entsprechend unter der Adresse `http://dx.doi.org/10.1000/186` als PDF-Dokument vorfindet ([Pas06]).

Die Besonderheit solcher DOI-Links ist hierbei ihre Persistenz. Während sich URLs im Laufe der Zeit ändern, bleibt ein DOI-Link stets erhalten. Ändert sich der Ort der Ressource, so muss lediglich der Eintrag im DOI-Server verändert werden – ein für den Benutzer völlig transparenter Vorgang. Der Vorteil von DOI-Links liegt damit klar auf der Hand: Selbst wenn ein Verleger seinen Webauftritt radikal umgestaltet, die innere Struktur verändert oder gar die Domain wechselt, ist ein und derselbe Artikel nach wie vor unter demselben Link verfügbar –

[9] Siehe hierzu beispielsweise Daten des BMC (vgl. Abschnitt 3.2.3):
In DBLP: `http://dblp.uni-trier.de/db/journals/bmcbi/bmcbi7.html`,
bei BMC: `http://www.biomedcentral.com/1471-2105/7`

ein Vorteil, den herkömmliche URLs nicht bieten. Im Bereich der digitalen Bibliotheken hat sich das DOI-Konzept daher recht weit verbreitet, wobei die entsprechenden Links i.d.R. auf die Abstract-Seiten der entsprechenden Artikel verweisen. Für den Datenbestand von DBLP sind sie daher äußerst interessant und sollten stets an Stelle instabiler URLs verwendet werden.

Wie wir gesehen haben, muss man streng genommen zwischen einem DOI (d.h. dem Bezeichner) und einem DOI-Link (d.h. einem URI, der den DOI beinhaltet) unterscheiden. Da wir einen Link zu einer 'Electronic Edition' suchen, werden wir also stets einen solchen DOI-*Link* suchen, welcher das Protokoll (`http://`) und einen Proxyserver (i.d.R. "`dx.doi.org`") beinhaltet. Wir werden im weiteren Verlauf dieser Arbeit bei solchen DOI-Links dennoch von DOIs sprechen, da dies dem allgemeinen Sprachgebrauch entspricht, auch wenn diese Bezeichnung wie oben beschrieben nicht ganz exakt ist. Ist dagegen von URLs die Rede, so wollen wir darunter 'normale', d.h. instabile URLs verstehen, und diese stets im Gegensatz zu persistenten DOI-Links betrachten.

Es sei noch erwähnt, dass einige Verlage oder DLs eigene Proxyserver zur Verfügung stellen, über welche eine schnellere Auflösung der eigenen DOIs möglich ist. Die ACM (vgl. Kapitel 3.2.1) bietet beispielsweise unter dem URL `http://doi.acm.org` einen solchen Server an. Selbstverständlich können alle DOIs der ACM auch über die Standard-Adresse abgerufen werden.[10]

Zur Wichtigkeit der Nutzung von DOIs sollen an dieser Stelle kurz zwei Beispiele geliefert werden, auf die wir in späteren Kapiteln genauer eingehen werden:

Springer LNCS Bei sehr vielen älteren Bänden der LNCS-Reihe[11] fehlen die DOIs; es sind – historisch bedingt – lediglich einfache URLs vorhanden, die teilweise nicht mehr korrekt funktionieren. Betrachten wir beispielsweise Volume 1500 aus dem Jahre 1999. In DBLP finden wir unter `http://dblp.uni-trier.de/db/conf/ewspt/sp1999.html` eine Auflistung sämtlicher Artikel dieses Bandes. Jeder Artikel enthält einen Link, der aus dem EE-Element ausgelesen ist. Für den ersten Artikel ("The Software Process: Modelling and Technology") lautet der entsprechende URL

`http://springerlink.metapress.com/openurl.asp?genre=article&issn=0302-9743&volume=1500&spage=0001`.

Folgt man diesem Link, so landet man im Inhaltsverzeichnis des Springer-Servers:

`http://springerlink.metapress.com/content/t59n3wqy16v7/`.

[10] Lautet ein DOI innerhalb des ACM-Portals beispielsweise "10.1145/1035570.1035571", so sind die Ziele der beiden folgenden DOI-Links identisch:
`http://doi.acm.org/10.1145/1035570.1035571`,
`http://dx.doi.org/10.1145/1035570.1035571`.

[11] für detailliertere Informationen zum Springer-Verlag und den "Lecture Notes in Computer Science" (LNCS) siehe Abschnitt 3.2.10

Dies ist äußerst unschön und streng genommen inkorrekt, da der Link zur entsprechenden Abstract-Seite führen sollte. Verfolgt man auf der DBLP-Seite einen beliebigen anderen Link jenes Bandes, so gelangt man jeweils zur gleichen TOC-Seite: Mindestens einer der Parameter des in DBLP gespeicherten URLs (höchstwahrscheinlich der letzte: *spage*) ist veraltet und führt nicht mehr zur gewünschten Abstract-Seite – obwohl diese existiert und mittlerweile auch einen DOI des Artikels (`http://dx.doi.org/10.1007/3-540-49205-4_1`) beinhaltet. Mit eben dieser Problematik werden wir uns im späteren Verlauf dieser Arbeit, in Szenario F-2'$_{\text{LNCS}}$ (Kapitel 6.1.4) noch eingehend beschäftigen.

IGI-Pub ⟶ IGI-Global Die ehemalige "IDEA Group Incorporated" (IGI)[12] betrieb bis zum Juni 2009 ihre DL unter dem URL `http://www.igi-pub.com`. Der Verlag verzichtete völlig auf die Nutzung von DOIs sondern griff auf interne Identifikationsnummern (IDs) zurück, die jedoch äußerst 'stabil' waren und sind, d.h. nicht dem o.g. Problem unterworfen, dass Artikel plötzlich nicht mehr gefunden werden können. Daher schien die Verwendung von DOIs nicht notwendig.

Im Juni 2009 jedoch änderte der Verlag sein äußeres Erscheinungsbild und trägt nun den Namen 'IGI-Global'. Dies wirkte sich auch und vor allem auf die Domain seiner digitalen Bibliothek aus, denn seither findet man alle Informationen unter der neuen Domain `http:\www.igi-global.com`.

Die neue Webseite enthält keinen Hinweis mehr auf ihre Vergangenheit; alle Links tragen den Präfix der neuen Domain. Die alte Domain (`igi-pub.com`) besteht noch immer und leitet einen Besucher sofort auf die neue Domain (`igi-global.com`) um, weshalb auch alle alten in DBLP eingetragenen EE-Links noch immer ans korrekte Ziel führen. Und dennoch: Sieht man die Sache etwas enger, so beinhaltet der Datenbestand von DBLP seit jener Umstellung im Juni 2009 in allen Artikeln des Verlags *veraltete* Informationen. Hätte IGI-Global auf DOIs zurück gegriffen, so hätte die Umstellung der Domain keinerlei Auswirkung auf die Aktualität der gespeicherten Daten gehabt.

2.3.5 Band, Heft, Monat, Jahr

Separate Angaben eines Bandes (*volume*), Heftes (*issue*), Publikationsmonats (*month*) und -jahres (*year*) werden bei den journals benötigt. Bei Konferenzen werden diese Angaben nicht weiter strukturiert, sondern einfach so wie vom Verleger angegeben nach DBLP kopiert.[13]

Bei journals hingegen wird versucht, möglichst exakte Angaben über die genannten Werte zu erhalten. Der Wert *volume* sollte hierbei möglichst immer vorhanden sein, meist auch die Angabe eines *issues*. Der Publikationsmonat ist optional, hier kann allerdings auch ein Zeitraum

[12] für detailliertere Informationen zu diesem Verlag siehe Kapitel 3.2.8

[13] Beispielsweise: Robotics, Intelligent Systems and Signal Processing, 2003. Proceedings. 2003 IEEE International Conference on – Volume: 2, Date: 8-13 Oct. 2003,
vgl. `http://ieeexplore.ieee.org/xpl/tocresult.jsp?isnumber=28679&isYear=2003`

(z.B. “September-December”) oder eine Jahreszeit (“Spring”, “Summer”, “Autumn”/“Fall”, “Winter”) angegeben sein. Die Jahreszahl hingegen sollte möglichst vorhanden sein und einen einzelnen, vierstelligen Wert (z.B. “2009”) enthalten. Hat ein Verleger hier eine Zeitspanne, also beispielsweise “2008-2009” angegeben, so sollte versucht werden, die einzelnen Artikel ihrem Publikationsjahr zuzuordnen. Dies wird sicher nicht immer möglich sein; eine Angabe mehrerer Jahre ist jedoch allgemein unüblich, wodurch jener Fall nur äußerst selten auftritt.

2.3.6 Zwischenüberschriften

Das Vorhandensein so genannter *section headers* (Zwischenüberschriften) stellt an sich bereits eine gesteigerte Datenqualität dar. Zwischenüberschriften helfen, die Datensätze bei der HTML-Darstellung von DBLP zu untergliedern und die Seiten, gerade wenn sie lange Listen von Artikeln enthalten, übersichtlicher und optisch ansprechender zu gestalten. Sofern es möglich ist, sollte daher versucht werden, jeden Artikel einer ‘section’ (und ggf. einer oder mehrerer ‘subsections’) zuzuordnen. Für die interne Qualität der Zwischenüberschriften gelten die gleichen Aussagen wie bei den *titles*. Die section headers werden sowohl in der Software als auch im weiteren Verlauf der Arbeit auch kurz als *sections* (bzw. *subsections*, *subsubsections*) bezeichnet. Dies ist zwar streng genommen nicht ganz korrekt, vereinfacht jedoch die Schreibweise und ist an diesen Stellen aus dem Kontext stets klar ersichtlich.

2.4 Definition eines geeigneten Datenmodells

Bevor wir in uns in den folgenden Kapiteln den Aufgaben der Extraktion und Fusion bibliographischer Daten widmen, soll hier ein allgemeines Datenmodell für eben jene definiert werden. Damit wird eine Grundlage zur Beschreibung theoretischer Sachverhalte, wie wir sie beispielsweise in den Kapiteln 7 und 8 im Kontext der Fusion zweier strukturierter Datenquellen verstärkt benötigen werden, gelegt. Gleichzeitig wurde darauf geachtet, das Datenmodell derart zu wählen, dass eine einfache Implementierung in die Software begünstigt wird. So können Analogien aller hier vorgestellten Konzepte im Programmcode gefunden werden (siehe hierzu auch Anhang B.4).

Einen einzelnen Artikel werden wir – gemäß der Terminologie innerhalb von DBLP – im Folgenden stets als *Record* bezeichnen und diesen ggf. mittels R abkürzen. Als größte Gruppe an Records, die gleichzeitig verarbeitet wird, werden wir einen Konferenz- oder Zeitschriftenband betrachten. Dies rührt von der Tatsache her, dass dies in DBLP die größte Menge an Artikellisten ist, welche gemeinsam auf einer HTML-Seite angezeigt wird. Mehrere Volumes werden, ebenso wie mehrere Konferenzbände, dagegen auf separaten Seiten angezeigt. Intern spiegelt sich dies in den Datenformaten wieder. So wird eine BHT-Datei niemals Daten unterschiedlicher Konferenz- oder Zeitschriftenbände enthalten. Eine solche Gruppe werden wir fortan als *Record-Liste* (L^R) bezeichnen. Der Begriff einer *Menge* wird hier bewusst vermieden, da es sich bei den o.g. Listen keinesfalls um Mengen im mathematischen Sinne handelt: Zum einen spielt bei den enthaltenden Records die Reihenfolge eine Rolle, zum anderen ist es

durchaus möglich, dass eine Liste Dubletten enthält, deren Eliminierung uns beispielsweise bei der Fusion beschäftigen wird.

Eine Record-Liste besteht also aus einzelnen Records, die thematisch zusammengehören, da sie einem gemeinsamen Band entstammen. Wir können sie formal als Tupel

$$L^R = (R_1, R_2, \cdots, R_n)$$

schreiben. Natürlich kann eine solche Liste auch leer sein, dann schreiben wir entweder "$L^R = ()$" oder, in Analogie zur Software, "$L^R = null$". Im folgenden werden wir stets die Aussagen "ein Objekt ist leer", "ein Objekt ist nicht gesetzt" sowie "ein Objekt hat den Wert *null*" synonym verwenden.

Da wir auch andere Tupel betrachten werden, die andere Sorten von Objekten beinhalten können, werden wir allgemein

$$L = (o_1, o_2, \cdots, o_n)$$

schreiben, wobei L (bzw. L^o) stets für eine solche Liste steht, o_i für beliebige Objekte der gleichen Art (z.B. wie oben Records, aber auch Autorennamen oder Namensteile). Diese Tupel, die mehrere Objekte des gleichen Types beinhalten, werden wir auch als *komplexe Objekte* bezeichnen.

Im Gegensatz hierzu stehen die *simplen Objekte*, die nur einen einzigen Wert im Sinne unserer bibliographischen Daten beinhalten. Das größte simple Objekt ist ein Record, d.h. die Abstraktion eines Zeitschriften- oder Konferenz-Artikels. Dieses beinhaltet zwar eine große Zahl weiterer Objekte, doch diese sind allesamt unterschiedlich und werden daher nicht als Liste nach obiger Definition angesehen.

Die mittels der Definition getroffene Unterscheidung in simple und komplexe Objekte wird sich vor allem bei der Fusion zweier gleichartiger Objekte (siehe Kapitel 6 bis 8) als unbedingt erforderlich erweisen. Simple Objekte lassen sich direkt fusionieren, d.h. man kann sich bei der Betrachtung zweier simpler Attribute (*title*, *ee*, *volume* etc.), aber auch bei der Betrachtung zweier Records, sofort um die Frage des Fusionsergebnisses kümmern (siehe Kapitel 8). Im Gegensatz hierzu ist es bei der Fusion zweier komplexer Objekte (Recordlisten, Autorenlisten) zunächst erforderlich, geeignete Fusionspartner zu finden, d.h. solche simple Objekte, die sinnvoll miteinander fusioniert werden können, da es sich unter großer Wahrscheinlichkeit um solche Objekte handelt, die die gleiche reale Instanz (d.h. den gleichen Artikel, die gleiche Person) bezeichnen (siehe Kapitel 7). Eine Sonderrolle nehmen hierbei die Autorennamen als Liste von Namensteilen ein. Diesen widmen wir uns in Kapitel 8.3.

Die einzelnen Objekte, aus welchen sich ein Record zusammensetzt, werden wir auch als *Attribute* bezeichnen. Ein Record definiert sich demnach nur durch seine Attribute, ebenso wie sich zwei Artikel beispielsweise anhand ihrer Titel, Autoren etc. identifizieren und unterscheiden lassen. Allgemein werden wir Attribute mit einem kleinen a abkürzen. Meinen wir ein bestimmtes Attribut, z.B. den Titel eines Artikels, so benennen wir es entsprechend mit seiner englischen Bezeichnung *title*.

Abbildung 2.4 liefert einen Überblick über das gesamte hier definierte Datenmodell.

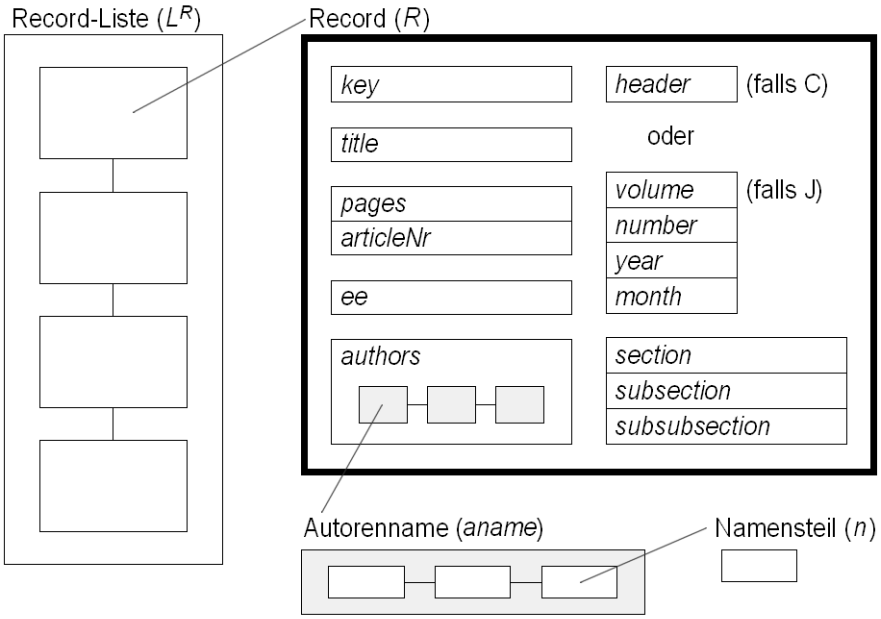


Abb. 2.4: Datenmodell der bibliographischen Daten

Wir sehen hier, wie bereits erwähnt, dass eine Record-Liste aus einzelnen Records besteht. Jedes Record besitzt eine Reihe von Attributen, wobei es sich mit Ausnahme von *authors* stets um simple Attribute handelt. Bei *authors* dagegen handelt es sich wiederum um eine Liste (L^{aname}) aus einzelnen Autorennamen, welche ihrerseits wieder aus einzelnen Namensteilen bestehen (also formal L^n, wenn wir Namensteile mit n bezeichnen).

Das *header*-Attribut spielt nur bei den *conferences* eine Rolle, während die Attribute *volume*, *issue*, *year* und *month* lediglich im Kontext der *journals* benötigt werden. Die jeweils anderen Attribute sind daher stets leer bzw. haben den Wert *null*.

Neben dem *pages*-Attribut existiert auch ein Attribut *articleNr*, welches wir nutzen können, um Artikelnummern abzuspeichern und eine Record-Liste anhand dieser Nummern zu sortieren.

Wir sehen an dieser Darstellung auch, dass evtl. vorhandene Zwischenüberschriften (*section*, *subsection*, *subsubsection*) – im Gegensatz zur internen Repräsentation in DBLP – in jedem Record separat gespeichert werden und nicht innerhalb der Recordliste. Dies ermöglicht es uns zum einen, das Listenobjekt recht einfach zu halten, hat aber zum anderen auch und vor allem den enormen Vorteil, dass derartige Informationen auch bei Umsortierung der Liste (z.B. durch Ordnung nach Seitennummern) stets erhalten bleiben.

Das Attribut *key* werden wir erst im Blick auf die Fusion benötigen, wenn wir bereits im Datenbestand enthaltene DBLP-Records einlesen, die bereits einen eindeutigen Schlüssel erhalten haben. Bei der Extraktion werden wir niemals Schlüssel vergeben – dies erfolgt ausschließlich durch die weiter verarbeitenden Skripte.

Kapitel 3

Praxisstudie: Informationsextraktionsquellen

Ziel der Wrapper wird es sein, automatisch bibliographische Daten von Websites diverser Verlage oder aus digitalen Bibliotheken zu lesen und diese in ein einheitliches Format zu bringen, so dass ein direkter Import nach DBLP möglich ist. Dieses Kapitel soll eine Übersicht über einige Verlage und DLs bieten, die für DBLP von besonderer Bedeutung sind und für die im Rahmen der vorliegenden Arbeit Wrapper erstellt wurden. Durch das Studium dieser Websites kann ein Einblick in die Vielfältigkeit der Probleme, die sich beim Gewinnen relevanter Informationen ergeben, gewonnen werden.

3.1 Vorgehensweise

Bei der Beschreibung der für unsere Zwecke relevanten Websites soll ein Eindruck über die dort verfügbaren Daten, deren Qualität, sowie die Navigation über die Verlags-Websites gewonnen werden. Auch technische Aspekte wie die Art der Parameterübertragung (GET- oder POST-Methode), die Verwendung unterschiedlicher Webtechnologien (JavaScript, PHP, ASP, JSP, CGI, CSS etc.) sowie die Struktur der empfangenen HTML-Seiten werden hierbei untersucht.

In einer speziellen Untergliederung werden jeweils die folgenden Punkte behandelt:

Allgemeine Informationen Dieser Abschnitt liefert eine kurze Beschreibung des jeweiligen Verlags bzw. der Organisation und dessen/deren Publikationen, vor allem in Hinblick auf deren Relevanz für die Informatik und damit für den Datenbestand von DBLP. Es wird ein ungefährer Überblick über Anzahl und Art der in unserem Kontext interessanten Publikationen gegeben. Sämtliche Mengenangaben beziehen sich dabei auf den aktuellen Zeitpunkt (September 2009) und wurden i.d.R. den Informationsseiten des jeweiligen Verlagsservers entnommen. Sämtliche Abbildungen der Screenshots wurden ebenfalls – sofern nicht anders angegeben – im September 2009 erstellt.

Seitenstruktur und Datenbestand Dieser Abschnitt stellt zunächst jeweils ein kleines Tutorial dar, um zu zeigen, auf welche Weise man die Daten manuell gewinnen kann. Der interessierte Leser erhält einen Überblick über die Seitenstruktur und die Navigation bei der Suche nach Publikationen. Weiterhin werden Besonderheiten aufgezeigt, die bei der Programmierung der Wrapper berücksichtigt werden müssen, beispielsweise die Verwendung von zusammengefassten oder aufgespaltenen Volumes oder Issues oder das Vorhandensein so genannter Supplements: Sonderhefte einer Publikationsreihe, die mehr oder weniger regelmäßig erscheinen und ebenfalls in den Datenbestand von DBLP aufgenommen werden sollen.

Anschließend werden die einzelnen Datensätze analysiert und die relevanten Informationen auf deren Verfügbarkeit und Vollständigkeit (Datumsangaben, Titel, Autorennamen, Seitennummern, URLs oder DOIs, Zwischenüberschriften), sowie auf die Qualität der Daten bzgl. Fehlerfreiheit und Informationsreichtum hin überprüft.

Technische Details An dieser Stelle wird auf das Format der übertragenen Seiten, deren Zeichencodierung, HTTP-Übertragungsmodus sowie verwendete server- und clientseitige Technologien eingegangen. Spezielle Verfahren wie das Setzen von Cookies oder die Verwendung von JavaScript/AJAX werden untersucht und auf deren Notwendigkeit zur Informationsgewinnung getestet. Zudem wird für einige repräsentative HTML-Seiten – meist die Startseite und einige Seiten mit Artikelinformationen – mittels des Validators des W3-Konsortiums[1] überprüft, ob diese Dokumente der jeweils angegebenen Dokumenttypendefinition entsprechen, um so Rückschlüsse auf die Sorgfalt der Datenpflege ziehen zu können. Details der Navigation werden aufgedeckt, also ob beispielsweise simple Hyperlinks oder aber kompliziertere Navigationstechniken verwendet werden. Auch auf den Zeichensatz und eventuell verwendete Strategien zur Zeichencodierung (z.B. benannte oder numerische Entities) wird eingegangen.

Vorgehensweise bei der Programmierung des Wrappers Dieser letzte Punkt erklärt abschließend, wie der entsprechende Wrapper die Daten erfassen kann. Dabei werden die jeweils zu beachtenden Besonderheiten eines Servers zusammengefasst und aufgezeigt, mit welchen Problemen dort daher zu rechnen ist.

3.2 Beschreibung ausgewählter Websites

Die Auswahl der hier vorgestellten Websites (Abschnitte 3.2.1 bis 3.2.12) ist einerseits durch die Reihenfolge, in welcher Wrapper entwickelt wurden, aber auch durch Besonderheiten der jeweiligen Verlage oder DLs begründet. Einige der hier vorgestellten Sites werden auch im zweiten Teil von besonderem Interesse sein, wenn wir uns mit der Informationsfusion beschäftigen. Eine Liste weiterer Websites, für welche bisher Wrapper erstellt wurden, findet sich im Abschnitt 3.2.13. Anhang D beinhaltet zudem die vollständigen Tabellen, die während der Durchführung dieser Studie entstanden.

[1] `http://validator.w3.org`

3.2.1 ACM

Allgemeine Informationen Die Association for Computing Machinery (ACM) ist weithin als eine der führenden Organisationen für Computerprofis anerkannt. Gegründet wurde sie im September 1947 an der Columbia Universität in New York als erste wissenschaftliche Gesellschaft für Informatik. Heute umfasst die ACM nach eigenen Angaben über 90.000 Mitglieder aus über 140 Ländern, vorwiegend aus der Industrie und dem akademischen Bereich. Intern ist die Organisation in derzeit 34 SIGs (Special Interest Groups) aus den unterschiedlichsten Forschungsrichtungen der Informatik unterteilt.

Für unsere Zwecke bedeutend ist die digitale Bibliothek der ACM, in der der Großteil aller seit ihrer Gründung publizierten Magazine, Journale, Tagungsbände und Newsletter meist lückenlos veröffentlicht sind. Nach Angaben der ACM kommen hier jährlich über 35 Zeitschriften, 30 Newsletter und mehr als 85 Konferenzen hinzu. Die meisten dieser Artikel sind online als PDF-Dokumente verfügbar und stehen Mitgliedern zum Download bereit [ACM09].

Im Datenbestand von DBLP finden sich viele der von ACM publizierten Zeitschriftenreihen wieder. Insgesamt sind 7 von 8 Journals, 6 von 10 Magazinen und etwas mehr als ein Viertel der Newsletter, sowie zahlreiche Konferenzbände dort komplett gelistet – fehlende Publikationsreihen wurden meist nur wegen der begrenzten Kapazität an Arbeitskräften nicht aufgenommen. Wichtige 'Zugpferde' wie beispielsweise das Magazin "Communications of the ACM" sind lückenlos im Datenbestand enthalten und werden regelmäßig aktualisiert.

Seitenstruktur und Datenbestand Ausgangspunkt einer manuellen Suche ist die Startseite des ACM Portals: `http://portal.acm.org/dl.cfm`. Hier kann man die Kategorie (Journals, Magazines, Transactions, Proceedings,...) wählen und gelangt in ein entsprechendes Untermenü, welches alle verfügbaren Publikationsserien enthält. Wählt man hier einen entsprechenden Titel, so gelangt man auf eine Übersichtsseite, in der man entweder das jeweils aktuelle Volume/Issue bzw. den aktuellen Jahrgang, oder aber das Archiv mit einer Liste aller verfügbarer Volumes und Issues bzw. Konferenzjahrgänge auswählen kann. Für unsere Zwecke ist natürlich letzteres Ausgangspunkt der Suche. Im URL kann man erkennen, dass zwei Identifikatoren (`idx` für die Serie, `id` für den ausgewählten Band) verfügbar sind. Eben jene Werte wird auch der Wrapper auslesen und somit sofort die entsprechende Archivseite finden.

Das Archiv selbst besteht aus untereinander angeordneten HTML-Tabellen, die jeweils ein oder mehrere Issues eines Volumes beinhalten (siehe Abb. 3.1, linke Seite). Hier sind allerdings je nach Publikation einige Unterschiede zu beachten. Manche Volumes beinhalten zusammengefasste Nummern (z.B. "Issue 1-4 (Mar 2008)"), andere enthalten überhaupt keine solchen, sofern es pro Volume nur eine einzige Nummer gibt (z.B. "Issue (Mar 2008)"). Hin und wieder kommt es sogar vor, dass ein und dasselbe Volume in mehrere Tabellen aufgespalten ist (d.h. beispielsweise ein Block für Volume 41, Issue 3 und 4, gefolgt von einem Block für Volume 40, der wiederum vor einem Block für Volume 41, Issue 1 und 2 steht). Dies passiert vorwiegend dann, wenn sich ein Volume über mehr als ein Jahr erstreckt. Hin und wieder stehen hinter

den einzelnen Nummern auch spezielle textuelle Informationen (z.B. “Special issue on xxx”), welche ebenfalls bei der Datenerfassung berücksichtigt werden sollen.

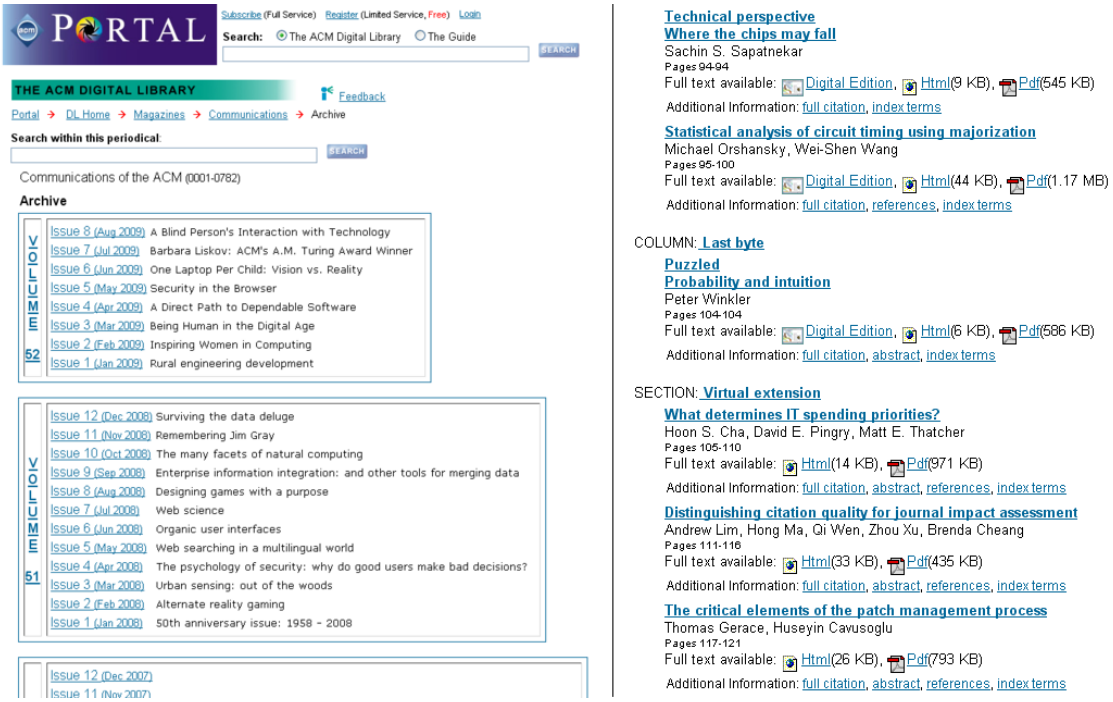


Abb. 3.1: Bibliographische Daten im Portal der ACM: Links eine Übersicht über einzelne Bände der Zeitung “Communications of the ACM”, rechts ein Ausschnitt der bibliographischen Daten eines einzelnen Heftes.
Quellen: `http://portal.acm.org/toc.cfm?id=J79`,
`http://portal.acm.org/toc.cfm?id=1536616`

Der Aufbau der Seiten ist sehr schlicht gehalten und bietet einen guten Überblick über die Vielzahl an Publikationen. Eine im oberen Bereich der Seiten immer vorhandene Menüstruktur zeigt jederzeit deutlich auf, wo sich ein Benutzer gerade befindet und erlaubt eine sehr einfache und intuitive Navigation durch die Inhalte der digitalen Bibliothek.

Alle Datensätze enthalten in der Regel die benötigten Informationen wie Titel und Autorennamen, einige Zwischenkapitel, bei denen eine dieser Angaben fehlt, können getrost ignoriert werden. Meist sind auch Seitenzahlen verfügbar, hin und wieder jedoch leider nur Artikelnummern, die uns nicht weiter helfen können. Oftmals sind die einzelnen Artikel in Blöcke unterteilt, welche mit Zwischenüberschriften versehen sind. Eine klare Struktur ist jederzeit erkennbar, was dem Wrapper die Arbeit erleichtert (siehe Abb. 3.1, rechte Seite). Jahres- und Monatsinformationen können, wie bereits oben gezeigt, aus der Übersicht des jeweiligen Archivs entnommen werden. Zu den meisten der Artikel existieren DOIs, welche zwar nur auf den Abstract-Seiten explizit angegeben sind, sich jedoch aus der jeweiligen `id` der Publikation direkt zusammenbauen lassen. So trägt beispielsweise der Artikel, welcher sich über

`http://portal.acm.org/citation.cfm?id=1452012.1452015`

aufrufen lässt, den DOI

`http://doi.acm.org/10.1145/1452012.1452015`.

Die Qualität der Daten ist ebenfalls sehr hoch, simple Tippfehler o.ä. konnten nicht gefunden werden. Personennamen sind in vorbildlicher Manier stets möglichst vollständig angegeben, Initiale findet man hier höchstens bei mehreren Vornamen.

Technische Details Die digitale Bibliothek der ACM wird von einem Microsoft IIS/6.0 Server in Unicode (UTF-8) unter Angabe der 'content-length' übertragen. Zur Generierung der dynamischen Inhalte wird auf das Softwarepaket ColdFusion, eine für Web-basierte Datenbankanwendungen konzipierte Middleware der Firma Adobe, zurückgegriffen, was an den Dateiendungen "`.cfm`" erkennbar ist.[2] Einzelne kleine JavaScript-Funktionen dienen lediglich dem Benutzerkomfort, sind aber keinesfalls für die korrekte Anzeige der Seiten erheblich. Zum Design der Seiten wird CSS benutzt, welches mittels `<script>`-Tags direkt in die HTML-Codes eingefügt ist. Leider handelt es sich hierbei hauptsächlich um eine Formatierung fester HTML-Elemente, nicht aber um die Definition eigener Klassen, welche uns bei der Aufspürung der Daten helfen könnten (vgl. hierzu Cambridge in Abschnitt 3.2.4). Nach Angabe des DOCTYPE-Headers soll es sich bei der Seitenbeschreibungssprache um 'xhtml 1.0 transitional' handeln, doch der Validator ist auf Grund mehrerer hundert Fehler offensichtlich anderer Meinung.

Zur Navigation werden normale Hyperlinks verwendet, die sich leicht auslesen und weiter verarbeiten lassen. Alle relevanten Parameter werden in jene Links codiert und mittels der GET-Methode übertragen. Viele Parameter sind hierbei für das Laden der für uns relevanten Informationen unerheblich und dienen oftmals nur der optischen Aufbereitung der Ergebnisse, beispielsweise durch Angabe des Typs[3] oder der Benutzerverfolgung durch Setzen verschiedener ColdFusion-interner Variablen.

Vorgehensweise des Wrappers Der Wrapper benötigt einen URL, in welchem zumindest eine Publikations-ID (`idx=...`) erkennbar ist. Diese wird ausgelesen und die entsprechende Publikation identifiziert. Handelt es sich um eine Konferenz, so können die Daten direkt erfasst werden. Bei anderen Publikationstypen muss zunächst das gewünschte Volume ermittelt werden. Durch den konsistenten Aufbau der Links ist es möglich, die Suche sofort auf der jeweiligen Archivseite zu starten. Dort werden nun alle für das jeweilige Volume relevanten Informationen gesammelt und abgespeichert. Hierbei sind die o.g. Besonderheiten (zusammengefasste Nummern, fehlende Nummern, mehrere Blöcke mit Informationen zum gleichen Volume) zu beachten. Anschließend werden alle gewünschten Nummern gescannt. Da jede Nummer bzw. jede Konferenz jeweils nur auf einer einzigen HTML-Seite untergebracht ist, brauchen keine weiteren Unterseiten betrachtet zu werden.

[2] ColdFusion erweitert HTML zur so genannten CFML, der 'ColdFusion Markup Language'.
[3] `type = journal|magazine|proceedings|...`

Einige Metadaten, beispielsweise das Publikationsdatum, spezielle Informationen auf der Archivseite bzw. Untertitel einer Konferenz werden vorab erfasst. Anschließend durchläuft der Wrapper den Block der einzelnen Artikel und liest die entsprechenden Daten. Vorsicht ist hier geboten, falls Zwischenüberschriften vorhanden sind. Diese müssen den jeweiligen Artikeln korrekt zugeordnet werden. Hin und wieder stehen einzelne Artikel zwischen solchen thematischen Blöcken, die nicht direkt zu diesen gehören. In anderen Fällen stehen mehrere dieser Artikel am Schluss der Seite und müssen nach der Extraktion korrekt anhand der Seitenzahlen in die chronologische Datenliste eingefügt werden. Da die technische Qualität – abgesehen von der mangelhaften Umsetzung in xhtml – von Server und Datenbestand äußerst vorbildlich ist, hat der Wrapper hier leichtes Spiel und muss sich keiner besonderen Herausforderung stellen.

3.2.2 ACTA Press

Allgemeine Informationen Bei ACTA Press handelt es sich um einen vergleichsweise kleinen Verlagsserver mit einer überschaubaren Anzahl an Zeitschriften (derzeit sieben) und Konferenzbänden (etwa 200) aus dem Bereich der Informatik.

Der Verlag mit Hauptsitz in Calgary, Kanada, wurde 1972 gegründet und publiziert derzeit Proceedings zu Konferenzen der "International Association for Science and Technology for Development" (IASTED), sowie einige wenige (nach aktuellem Stand sieben, von denen eine nicht weitergeführt wird) Online-Zeitschriften [ACT09]. Da die dort auftretenden Journale eine zunehmend interessanter werdende Rolle spielen, wurde auch für diesen Server ein Wrapper entwickelt. Derzeit finden sich jedoch noch keine Publikationen dieses Verlags im Datenbestand von DBLP.

Seitenstruktur und Datenbestand Die Navigation auf der Homepage des Servers ist sehr benutzerfreundlich gestaltet, die Seite selbst weist ein hübsches und ansprechendes Design auf. Über das Hauptmenü auf der linken Seite kann man sowohl die Startseite der Zeitschriften[4] als auch jene der Konferenzen[5] erreichen, weitere Boxen stellen direkte Links zu den entsprechenden Publikationen bereit. Auf den entsprechenden Unterseiten wird man gleich über die Kosten der jeweiligen Publikation unterrichtet, bevor man etwas über deren Inhalt erfährt.

Jede Zeitschrift oder Konferenz enthält auf ihrer Übersichtsseite eine weitere Menüebene, die über Karteikartenreiter erreicht werden kann. Die Navigation erfolgt mittels JavaScript – erlaubt der Benutzer die Ausführung jenes Programmcodes nicht, so bricht die Navigation zusammen und statt der erwarteten Informationen erscheint lediglich die Nachricht "Loading Information".

Bei aktiviertem JavaScript hingegen kann man die Liste der Publikationen über den Reiter "Papers" betrachten (siehe Abb. 3.2). Hier erhält man einen kompakten Überblick über Titel und

[4] `http://www.actapress.com/journals.aspx`
[5] `http://www.actapress.com/proceedings.aspx`

Autoren, die DOIs lassen sich – soweit vorhanden – über den Klick auf den am unteren Ende der Tabelle platzierten Button "Show Digital Object Identifiers" hinzu blenden. Ein Link zur entsprechenden Abstract-Seite ist ebenfalls verfügbar, auf dieser finden sich jedoch keine weiteren für uns interessanten Informationen. Eine Angabe von Seitennummern sucht man leider vergeblich. Zu jedem Volume einer Zeitschrift bzw. jedem Konferenzband ist das Publikationsjahr angegeben, ein entsprechender Eintrag für den Publikationsmonat fehlt vollständig. Jedes Volume besteht aus einzelnen Issues, die von 1 an durchgängig nummeriert sind. Es existieren keine Doppelnummern oder ähnliche Abweichungen von dieser Norm. Bei den Konferenzen sind die einzelnen Artikel, ähnlich den Nummern der Zeitschriften, in einzelne thematische Blöcke untergliedert, die jeweils mit einer festen Zwischenüberschrift beginnen.

Abb. 3.2: Website der ACTA Press Company: Ansprechendes Design, aber leider nur mäßige Datenqualität durch fehlende Seitennummern, abgekürzte Vornamen und Titel in Großbuchstaben.
Quelle: `http://www.actapress.com/Content_of_Journal.aspx?journalid=111#pages`

Insgesamt machen die Daten einen recht konsistenten Eindruck und scheinen bei Eingabe überprüft zu werden, so dass keine schwerwiegenden Fehler erkennbar sind. Leider stehen jedoch oftmals sowohl Titel als auch Autorennamen nur in reinen Großbuchstaben zur Verfügung. Zur Erfassung in DBLP müssen diese daher entsprechend aufbereitet werden, was vor allem bei den

Titeln zu Fehlern führen kann. Zudem sind bei den Autorennamen meist keine ausgeschriebenen Vornamen vorhanden.

Technische Details Die Seiten des Verlags sind auf einem Microsoft-IIS/6.0 Server untergebracht, welcher bei der Übertragung die entsprechende Länge des Textes (content-length) liefert, und werden mittels ASP generiert. Der übertragene Zeichensatz benutzt die Version UTF-8 des Unicode Standards und macht hiervon auch regen Gebrauch; viele Sonderzeichen stehen direkt im Unicode-Format im HTML-Text.

Der Server nutzt Cookies, um Benutzerinformationen zu speichern. Eine Ablehnung dieser Cookies hat jedoch keinerlei Auswirkungen auf die Erreichbarkeit des Inhaltes. Die übertragenen HTML-Seiten sind nach Angaben des DOCTYPES im Format 'xhtml 1.0 strict', doch eine Validierung zeigt eine große Anzahl an syntaktischen Fehlern auf.

Zur Navigation wird, wie bereits beschrieben, JavaScript genutzt. Viele Webdesigner halten es noch immer für besonders benutzerfreundlich, wenn nach Auswahl eines Menüpunktes in einem Drop-down-Menü (in HTML mittels eines `<select>`-Tags umgesetzt) sogleich der entsprechende Inhalt angezeigt wird, ohne dass ein Submit-Button gedrückt werden muss. Eine Realisierung dessen ist jedoch nur mittels der serverseitig ausgeführten Skriptsprache JavaScript möglich. Durch Verwendung des 'onchange'-Event-Handlers werden bei Auswahl eines anderen Volumes des entsprechenden Journals bzw. eines anderen Jahrgangs eines Konferenzbandes einige versteckte Formularfelder mit entsprechenden Informationen gefüllt, welche anschließend mittels POST-Methode an den Server übertragen werden. Dieser liefert daraufhin umgehend die neue HTML-Seite zurück. Leider scheinen die entsprechenden Routinen recht instabil, so dass es auch bei manueller Navigation hin und wieder zu Fehlermeldungen kommt.

Vorgehensweise des Wrappers Die größten Probleme ergeben sich aus den zuletzt beschriebenen technischen Besonderheiten der einzelnen Volumes einer Zeitschrift. Bei Aufruf muss dem Wrapper ein URL der Form

`http://www.actapress.com/Content_of_Journal.aspx?journalid=xxx`

übermittelt werden, aus welchem sich direkt eine Journal-ID ermitteln lässt. Problematisch ist allerdings, dass es sich hierbei entgegen der mit dem Namen verknüpften Erwartungen nicht um eine wirkliche Journal-ID handelt. Jene `journalid` ändert sich nämlich beim Aufruf eines jeden Volumes. Daher ist es nötig, die entsprechenden Formulardaten (s.o.) von der aktuell bekannten Seite einzulesen und dann die Funktionsweise der JavaScript-Funktionen zu simulieren. Statt ein verstecktes Formular zu füllen und abzuschicken, werden die Daten entsprechend ausgelesen und mittels GET-Methode übertragen, d.h. es wird ein passender URL konstruiert, der dem Server übergeben wird. Hierzu genügt es jedoch nicht, die bandbezogenen Daten abzusenden, sondern es müssen auch spezielle Sessionvariablen mit übertragen werden, die zuvor aus den jeweiligen Formularfeldern ausgelesen und codiert werden müssen – ein Unterfangen, welches ein automatisch generierter Wrapper nicht bzw. nur mit erheblichem Mehraufwand zu leisten in der Lage wäre.

Ein weiteres Problem ergibt sich aus technischen Unzulänglichkeiten des Verlags. Der 'ACTA Press'-Server ist leider sehr oft nicht erreichbar, was den Wrapper oftmals zur vorzeitigen Terminierung führt. Ein Versuch, die entsprechende Seite mittels Browser zu erreichen, scheitert in diesem Fall meist ebenfalls. Die entsprechende Anfrage wird intern in einen Fehler (Code 302) umgewandelt, anschließend wird eine Standard-Fehlerseite geladen.

3.2.3 BMC

Allgemeine Informationen Bei BMC (BioMed Central) handelt es sich um einen kommerziellen, wissenschaftlichen Verlag mit Sitz in London, der hauptsächlich Zeitschriften aus den Bereichen Biologie und Medizin publiziert. Der im Jahre 2000 gegründete Verlag gilt als Pionier der so genannten Open-Access-Publikationen, bei denen alle veröffentlichten Artikel unmittelbar und frei zugänglich sind. Derzeit erscheinen dort über 60 verschiedene Zeitschriften der "BMC Serie", sowie etwa 140 weitere, meist unabhängige Zeitschriften [BMC09]. Im Herbst 2008 wurde BioMed Central vom Springer-Verlag (vgl. 3.2.10) aufgekauft.[6]

Während der Großteil der Publikationen für die Informatik unerheblich ist, wurden in den vergangenen Jahren einige spezielle Serien, allen voran "BMC Bioinformatics" in den Datenbestand von DBLP aufgenommen. In diesem seit Gründung des Verlags bestehenden Journal werden jährlich mehrere hundert Artikel sowie einige Supplements in Form von Konferenzbänden publiziert. Allein die bloße Anzahl der Artikel dieses einen Journals verlieh der Erstellung eines entsprechenden Wrappers ihren Sinn.

Seitenstruktur und Datenbestand Über die Homepage des BMC ist es möglich, die einzelnen Journale auszuwählen. Da unser Interesse jedoch hauptsächlich einem der Bände gewidmet ist, startet unsere Suche im entsprechenden Unterverzeichnis

`http://www.biomedcentral.com/bmcbioinformatics/`.

Dort lässt sich im Menü auf der linken Seite der Punkt "Archive" auswählen, welcher sogleich auf den aktuellen Band verweist. Auf der rechten Seite erscheint ein Menü, über welches sich alte Publikationen abrufen lassen. Wegen der Vielzahl an Artikeln, die lediglich durch deren Veröffentlichungsjahr in Volumes unterteilt sind – eine weitere Aufspaltung in Issues existiert außer bei den Supplements nicht – werden jeweils nur 20 Artikel pro Seite angezeigt. Durch Klick auf den Link "next page >>" kann durch die weiteren Seiten navigiert werden. Ein direkter Sprung auf eine beliebige Unterseite (z.B. an den Anfang der Publikation) ist lediglich durch Hacken der URL-Adresse (d.h. indem man den Wert "`?page=x`" manuell verändert) möglich. Da wir bei der Datenerfassung sequentiell vorgehen werden, wird uns dies allerdings, im Gegenteil zu einem Benutzer, der die Seite manuell betrachtet, nicht weiter stören (siehe Abb. 3.3).

[6] `http://www.fachzeitungen.de/pressemeldungen/springer-erwirbt-biomed-central-group-10610/`

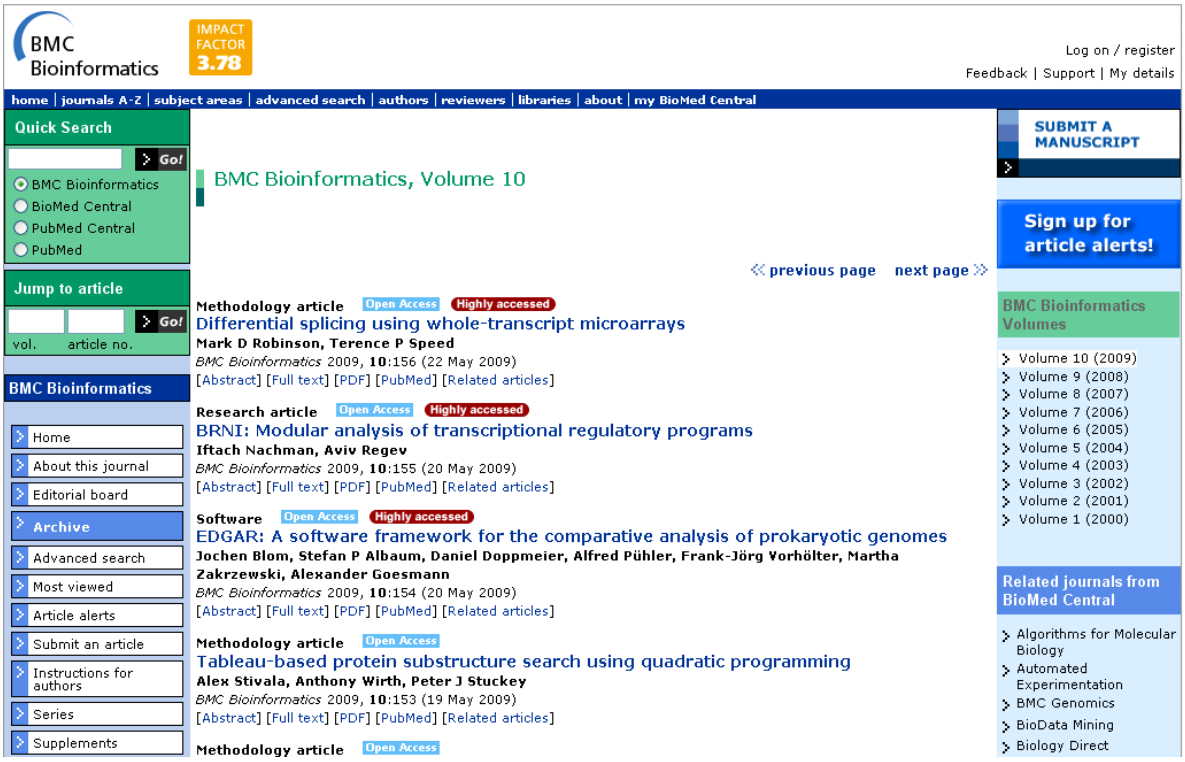


Abb. 3.3: Artikel des Journals "Bioinformatics" des BMC: Ein Volume entspricht einem Jahrgang (hier beispielsweise Volume 10 = 2009), die Artikel werden, nach Erscheinungsdatum sortiert und mit laufenden Nummern versehen, eingefügt.
Quelle: `http://www.biomedcentral.com/bmcbioinformatics/archive/`

Wählt man im Menü auf der linken Seite den Punkt "Supplements", so gelangt man auf eine Übersichtsseite, welche alle verfügbaren Supplements aller Volumes auf einer einzigen Seite auflistet. Verfolgt man einen der Links, mit denen die jeweiligen Titel unterlegt sind, so gelangt man auf die erste Seite der Artikel, welche exakt analog der normalen Volumes aufgebaut ist. Auch wenn es sich hierbei wie eingangs erwähnt um Konferenzbände handelt, werden wir diese bei der Datenerfassung wegen ihrer Erscheinungsform (d.h. der Zuordnung zu einem Volume und der Vergabe einer Nummer) wie Journale behandeln.

Die Qualität der bibliographischen Daten erscheint recht hoch, alle Daten sind vollständig und weisen keine augenscheinlichen Fehler auf. Die einzelnen Artikel enthalten jeweils Informationen zu Titel und Autorennamen, wobei letztere meist recht vollständig zu sein scheinen. An einigen Stellen ist jedoch Vorsicht geboten, wenn Initiale mehrerer Vornamen verschmolzen werden (so findet man beispielsweise den Namen "Otavio JB Brustolini", welchen wir nach DBLP-Konvention als "Otavio J. B. Brustolini" aufnehmen möchten). Weiterhin findet man neben dem Publikationsdatum zwei mittels Doppelpunkt abgetrennte Nummern, von welchen die erste in Fettschrift angezeigt wird und das Volume bezeichnet, die zweite der Artikelnummer entspricht (z.B. **10**:156). Hieraus, und auch durch die Betrachtung des Datums, lässt sich erkennen, dass die Artikel in *umgekehrter* Reihenfolge stehen, d.h. dass jeweils der neueste Artikel an erster Stelle steht, der älteste (des entsprechenden Jahres) an letzter. Dies ist für einen Besucher sicherlich hilfreich, stellt den Wrapper jedoch vor das Problem der korrekten Sortie-

rung, denn in DBLP möchten wir die Artikel in aufsteigender Reihenfolge ihrer Publikation einordnen.

Einen DOI findet man auf der Abstract-Seite, doch lässt sich dieser auch sehr einfach aus den im Inhaltsverzeichnis enthaltenen Informationen konstruieren. So findet man beispielsweise die Seite

`http://www.biomedcentral.com/1471-2105/10/194/abstract`

unter dem DOI

`http://dx.doi.org/10.1186/1471-2105-10-194`,

wobei "1471-2105" dem Schlüssel des entsprechenden Journals (in diesem Fall eben "BMC Bioinformatics") entspricht.

Jeder der Artikel wird zudem zu einer Kategorie (z.B. "Research article", "Software", "Methodology article", ...) zugeordnet, welche bei erster Betrachtung als Zwischenüberschrift dienen könnte. Bei genauerem Hinsehen wird jedoch klar, dass die einzelnen Themen derart oft wechseln, dass hierzu auch eine Sortierung nach jenen Zwischenüberschriften und nicht nach Datum der Veröffentlichung notwendig wäre, was jedoch von DBLP nicht gewünscht ist.

Technische Details Die Seiten von BioMed Central werden von einem Microsoft IIS 5.0-Server übertragen, was auf einen mit Windows 2000 ausgestatteten Rechner schließen lässt. Dementsprechend nutzt der Server ASP zur Generierung der dynamischen Inhalte. Die Zeichencodierung ist durchgängig in Latin-1 gehalten, sämtliche Sonderzeichen werden über numerische Entities übertragen. Das HTTP-Protokoll gibt weder eine 'content-length' noch den 'chunked'-Mode an. Einige Cookies werden übertragen, führen bei Ablehnung aber zu keinerlei sichtbaren Nachteilen.

Die HTML-Seiten weisen keinen DOCTYPE auf, importieren jedoch den XML-Namensraum zu 'MathML', was nicht nur dem Validator recht seltsam vorkommt. Bei einer Validierung als MathML 2.0-Dokument treten mehrere tausend Fehler auf, setzt man HTML 4.01 voraus, was eher glaubwürdig erscheint, reduziert sich die Anzahl der Fehler auf etwa 10 % und ist somit mit mehreren hundert Fehlern noch immer äußerst mangelhaft. JavaScript wird zwar an einigen Stellen für optische Effekte eingesetzt, ist für die Erreichbarkeit der Daten jedoch nicht relevant.

Die interne Struktur der Seite ist sehr intuitiv gehalten, die jeweilige Position ist aus dem entsprechenden URL, meist ohne Verwendung von Parametern ersichtlich. Lediglich zur Navigation wird, wie oben bereits beschrieben, eine Variable "`page`" genutzt.

Vorgehensweise des Wrappers Ausgangspunkt für die Sammlung der Daten ist die o.g. Archiv-Startseite. In der Übersicht auf der rechten Seite wird der Link des entsprechenden

Volumes gesucht, bei Angabe einer Supplement-Issuenummer (wie bereits erwähnt existieren Issues nur für die Supplements) wird die entsprechende Information von der Supplements-Übersichtsseite gewonnen.

Das eigentliche Erfassen der Daten stellt keine größeren Probleme dar, da die Präsentation auf der Seite recht eindeutig ist. Die DOIs werden nach dem beschriebenen Muster aus Volume und Artikelnummer zusammengesetzt. Einzige Schwierigkeit ist die chronologische Sortierung der Artikel, da diese keine Seitennummern aufweisen und der genaue Datumswert nicht abgefragt wird. Auch eine Sortierung nach DOIs, welche ja die entsprechende Artikelnummer beinhalten, schlägt fehl, da die Strings keine Unterscheidung der Wertigkeitsstellen der entsprechenden Zahl vornehmen können und somit lediglich nach dem ASCII-Wert der jeweiligen Stellen geordnet würde, also ergäbe sich bei Annahme einhundert fortlaufend nummerierter Artikel statt der Reihenfolge $[1, 2, 3, ...100]$ die unerwünschte Sortierung $[1, 10, 100, 11, 12, ...99]$.

Aus diesem Grund ist der Wrapper gezwungen, die Artikelnummern direkt als Integerzahlen abzuspeichern, um eine Sortierung zu ermöglichen. Diese Funktion wurde in der Software implementiert und kommt nur beim `BmcWrapper` zum Einsatz.

3.2.4 Cambridge University Press

Allgemeine Informationen Der Verlag der Cambridge Universität feiert in diesem Jahr (2009) sein 475. Bestehen und genießt somit eine lange und weitreichende Tradition. Dem 1534 gegründeten Verlagshaus wurde im gleichen Jahr vom englischen König Henry VIII das 'Letters Patent', d.h. das Recht "alle Arten von Büchern" ("all manner of books") zu drucken, gewährt [Cam09]. Das Druckwesen umfasst eine Vielzahl an Journalen, in welchen über 36.000 aus über 120 Ländern publizieren. Bedeutende Werke von Isaak Newton, Albert Einstein oder Stephen Hawking – um nur einige wenige zu nennen – wurden im Laufe der Geschichte von jenem Verlag veröffentlicht. Zudem handelt es sich um den weltweit ältesten Verlag, der die Bibel publiziert: Deren erstes Exemplar wurde im Jahre 1591 gedruckt.

Auf der Homepage der digitalen Bibliothek sind über 220 Journale der unterschiedlichsten wissenschaftlichen Fachrichtungen verfügbar, von denen derzeit 12 zum Gebiet der Informatik ('Computer Science') gerechnet werden. Im Datenbestand von DBLP werden bisher sieben dieser Journale indexiert, wobei von einigen nicht alle verfügbaren Titel erfasst sind.

Seitenstruktur und Datenbestand Über die Hauptseite der 'Cambridge University Press' führt ein Link direkt zur digitalen Bibliothek der Journale. Dort kann man gezielt nach einzelnen Titeln suchen oder sich alle verfügbaren Zeitschriftenreihen, entweder nach Titel oder Fachrichtung sortiert, anzeigen lassen. Auf der Übersichtsseite einer solchen Zeitschrift ist neben einer Kurzbeschreibung und einigen weiterführenden Informationen auch eine Liste aller verfügbaren Titel enthalten. Direkt sichtbar ist hierbei jeweils nur der aktuelle Band, alle früheren Publikationen lassen sich durch Anklicken des Links "Back Volumes" einblenden und sind ihrerseits wiederum in einzelne Blöcke unterteilt (siehe Abb. 3.4), durch die eine gezielte

Suche in übersichtlicher Weise möglich ist – zumindest sofern man JavaScript aktiviert hat, siehe hierzu den folgenden Abschnitt.

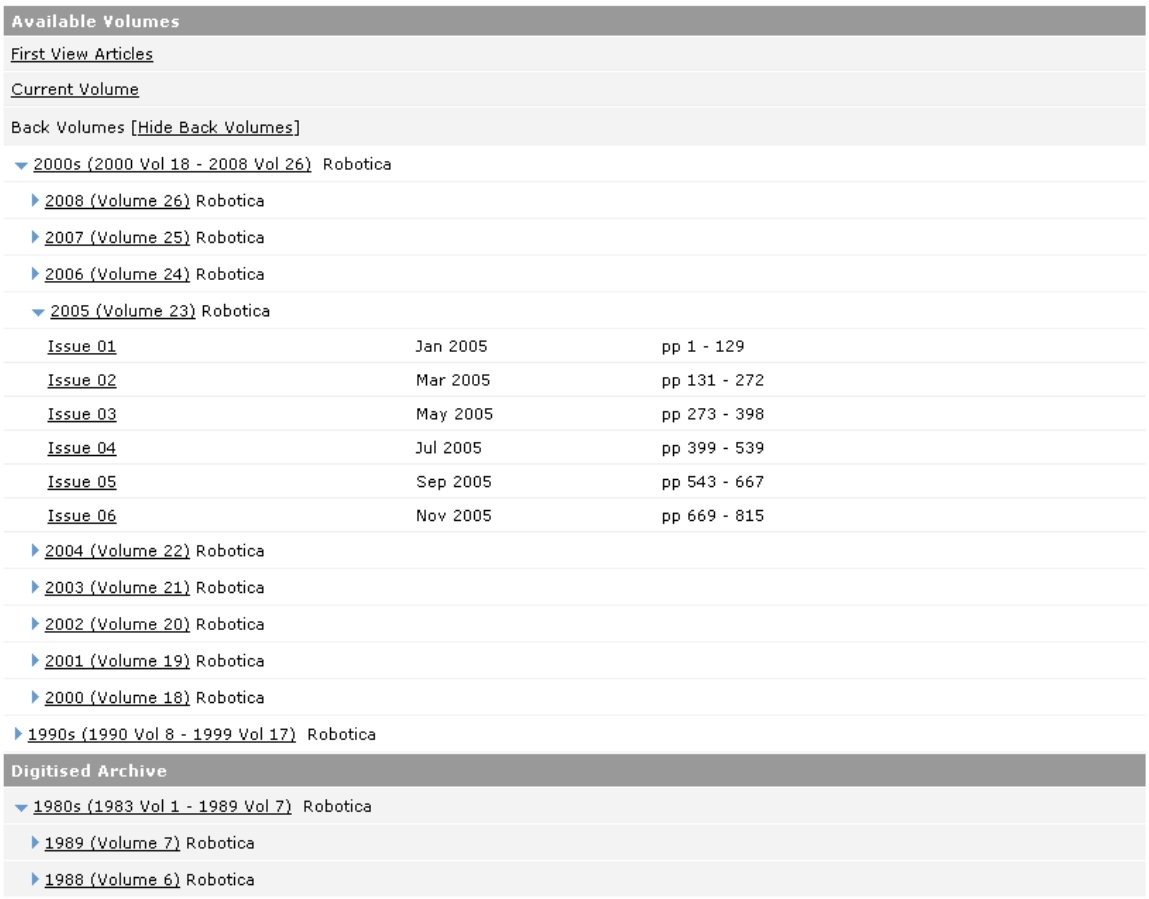


Abb. 3.4: Hierarchische Struktur der Artikel bei 'Cambridge': JavaScript-Navigation, die für den Wrapper unerheblich ist, da sämtliche Informationen im HTML-Text vorhanden sind. *Quelle:* `http://journals.cambridge.org/action/displayJournal?jid=ROB`

Einige Issues werden in Doppelnummern zusammengefasst, andere tragen besondere Namen (z.B. "Special Issue"). Die Nummerierung ist jedoch durchgehend konsistent, spezielle Sonderbände (Supplements) existieren derzeit nicht. Hat man sich für ein bestimmtes Issue eines Volumes entschieden, so gelangt man auf die entsprechende Inhaltsseite, welche alle benötigten Informationen enthält (Titel, Autorennamen, Seitenzahlen). Auch Zwischenüberschriften sind vorhanden, die die einzelnen Artikel in logische Blöcke teilen und somit ebenfalls in DBLP übernommen werden sollten. Zu jedem Artikel sind außerdem Publikationsmonat und -jahr angegeben, DOIs sind ebenfalls oftmals verfügbar, hin und wieder fehlen sie jedoch komplett oder auch nur bei einzelnen Artikeln eines Issues. Da es nicht möglich ist, von anderen Informationen auf die DOI-Nummern zu schließen, und auch keine weiteren Informationen auf den Abstract-Seiten gefunden werden können, müssen so hin und wieder normale URLs in DBLP eingefügt werden. Hieraus resultiert leider bei älteren Publikationen das Problem, dass ein Klick auf den Link "Electronic Edition" in DBLP nicht zum gewünschten Artikel, sondern lediglich zur Hauptseite der Journalsuche führt, da sich die interne Struktur der Seite mittler-

weile geändert hat und alte Variablen- oder Verzeichnisnamen nicht mehr korrekt verarbeitet werden können.

Die Qualität der Daten variiert von Zeitschrift zu Zeitschrift und selbst innerhalb einzelner Publikationen recht stark. Bei vielen Bänden sind sowohl Titel als auch Autorennamen ausschließlich in Großbuchstaben gehalten, was eine fehlerbehaftete Nachbearbeitung nötig macht. Die Vornamen der Autoren sind zudem häufig nicht vollständig, sondern nur in Form von Initialen angegeben.

Technische Details Die 'Cambridge University Press' verwendet die gängige Kombination eines Apache-Servers in Verbindung mit der Skriptsprache PHP zur Generierung und Übertragung ihrer Webseiten. Das HTTP-Protokoll nutzt den 'chunked' Modus zur schnellen Übertragung der dynamischen Inhalte. Cookies werden zwar gesetzt (beispielsweise die bei PHP übliche `SESSION-ID`), eine Ablehnung selbiger führt jedoch zu keinen erkennbaren Nachteilen beim Durchsuchen der Seiten. Die HTML-Seiten selbst geben vor, zur 'xhtml 1.0 Transitional'-DTD konform zu sein, doch einige hundert Fehler bei der Validierung widersprechen dieser Behauptung.

Wie bereits erwähnt nutzen die Seiten JavaScript, um clientseitig dynamisch bestimmte Inhalte der Inhaltsverzeichnisse ein- oder auszublenden. Verbietet ein Benutzer die Ausführung der Skripte, so erkennt der Server dies und gibt in vorbildlicher Weise eine Warnmeldung in Verbindung mit einem Link zur Anzeige aller Inhalte aus. Der Benutzer büßt also lediglich ein wenig Komfort ein, muss jedoch nicht auf die gewünschten Informationen verzichten. Für unseren Wrapper ist dies jedoch gänzlich unerheblich, da die Quelltexte der HTML-Seiten alle benötigten Informationen enthalten, ganz egal ob diese im Browser sichtbar oder ausgeblendet sein sollen. Alle Links wurden mittels normaler HTML-Hyperlinks realisiert, was die Sammlung und Verfolgung eben jener erleichtert. Variablen werden dementsprechend im jeweiligen URL codiert und mittels der GET-Methode übertragen.

Alle Seiten nutzen den UTF-8 Zeichensatz, wobei zur Anzeige der über Latin-1 hinausgehenden Zeichen hin und wieder entsprechende benannte bzw. dezimal numerische Entities genutzt werden. Zudem tauchen an einigen Stellen, vorwiegend in mathematischem Kontext, Image-Tags auf, die kleine Bilder von Sonderzeichen einfügen. Hier ist es in einigen Fällen möglich, auf das entsprechende Zeichen zu schließen; so lässt sich beispielsweise aus dem HTML-Tag

```
<img src=/[...]/xs1D4D4.gif alt=xs1D4D4$>
```
[7]

sowohl über den Dateinamen als auch über den alternativen Text auf ein Unicodezeichen schließen, dessen hexadezimaler Code "`1D4D4`" lautet. Tatsächlich handelt es sich hierbei um ein mathematisches Symbol: Ein großer, mit dem Formattribut 'fett' versehener Buchstabe "E" in Schreibschrift ($\boldsymbol{\mathcal{E}}$)[8]. Inhalte anderer Bilder können leider nicht auf diese Weise herausgefunden

[7] z.B. in `http://dx.doi.org/10.1017/S0960129508007354`
[8] vgl. `http://unicode.org/cldr/utility/character.jsp?a=1D4D4`

werden, da manchmal lediglich eine Artikelnummer angegeben wird. So lässt beispielsweise das Tag

`<img src="/[...]/S0010437X07003405\_inline1.gif"[...]>`[9]

nicht auf den Inhalt des entsprechenden Bildes – in diesem Fall die Zeichenkombination "L_∞" – schließen.

Vorgehensweise des Wrappers Der Wrapper benötigt als Eingabeparameter einen URL, in welchem der entsprechende Identifikationsschlüssel der zu erfassenden Zeitschrift als Parameter enthalten ist (`?jid=...`). Da dies sowohl beim URL der Startseite einer Publikation als auch bei jedem auf eine der Detailseiten verweisenden URL der Fall ist, stellt dies kein größeres Problem dar. Durch die Verwendung regulärer Hyperlinks ist die Navigation zur gewünschten Stelle für den Wrapper denkbar einfach. Auch die Identifikation der relevanten Daten innerhalb des Quelltextes ist problemlos möglich. Schwierigkeiten bereitet lediglich die Nachbereitung der gefundenen Inhalte aus bereits erwähnten Gründen: Die Zeichencodierung wird nicht konsequent durchgehalten, Image-Tags müssen, soweit dies überhaupt möglich ist, in entsprechende hexadezimale Entities umgewandelt und anschließend durch Latin-1 Zeichen ersetzt werden. Zudem müssen Titel und Autorennamen, die in reinen Großbuchstaben angegeben sind, entsprechend umgewandelt werden, um von DBLP akzeptiert zu werden.

3.2.5 Elsevier / ScienceDirect

Allgemeine Informationen Der Name des Verlagshauses 'Elsevier' geht ursprünglich auf Louis Elzevir, einen Drucker, Buchbinder und Buchhändler zurück, der bereits im 16. Jahrhundert in der niederländischen Universitätsstadt Leiden wissenschaftliche Schriften von berühmten Persönlichkeiten wie René Descartes, Galileo Galilei oder Josepf Justus Scaliger verlegte. 1620 entwarf dessen Enkel, Isaac Elzevir, das noch heute vom Verlag publizierte Firmenlogo. Nach dem Tod des letzten Familienmitglieds 1712 blieb das Unternehmen jedoch ohne Nachfolge.

Erst im Jahre 1880 gründete Jacobus George Robbers, ein niederländischer Buchhändler, den Elsevier Verlag, der heute unbestritten als einer der weltweit führenden Wissenschaftsverlage gilt. Er gehört zur 'Reed Elsevier Gruppe', dem fünftgrößten Medienkonzern der Welt und beschäftigt etwa 7000 Mitarbeiter in über 20 Ländern. Während sich der Hauptsitz des Verlags seit 1887 in Amsterdam befindet, wurden in vielen europäischen Ländern regionale Standorte eröffnet; auch in Deutschland gehört die Elsevier GmbH mit Sitz in München "zu den führenden Informationsanbietern in den Bereichen Medizin, Naturwissenschaft und Technik" [Els09].

Größter Stützpfeiler des Unternehmens sind dessen Internetportale "ScienceDirect", "Scorpus", "Scirus", "Embase", "Engineering Village", "Compendex" und "Cell", die 75 % des derzeitigen

[9] `http://dx.doi.org/10.1112/S0010437X07003405`

Jahresumsatzes einbringen. Wichtigstes Produkt ist "ScienceDirect", die digitale Bibliothek wissenschaftlicher Publikationen (ST, d.h. 'Science & Technology'), in deren Datenbank mehr als 9 Millionen – das ist etwa ein Viertel aller wissenschaftlichen Informationen weltweit – gespeichert sind [Els09].

Seitenstruktur und Datenbestand Die Startseite der 'ScienceDirect' Bibliothek bietet diverse Möglichkeiten, nach relevanten Büchern oder Journalen zu suchen. Da der entsprechende Wrapper derzeit nur die Erfassung von Journalen unterstützt, werden wir uns im Folgenden auch auf eben jene beschränken.

Wählt man ein Journal aus, so gelangt man automatisch zum Inhaltsverzeichnis des derzeit aktuellsten Bandes. Ein Menü auf der linken Seite bietet die Möglichkeiten, ältere Publikationen der gleichen Zeitschrift anzusehen. Aus Gründen der Übersichtlichkeit ist dieses Menü meist in mehrere Blöcke unterteilt, die jeweils eine zusammenhängende Teilmenge der verfügbaren Titel (z.B. "Volumes 1-10") beinhalten. Die kleinste verfügbare Menge sind hier die Issues, wobei einige Zeitschriften, wie beispielsweise die "Electronic Notes in Theoretical Computer Science"[10], lediglich in Volumes unterteilt sind, andere wiederum zusammengefasste Nummern (z.B. "Issues 5-6") aufweisen. Zudem enthält das Menü einige weitere Informationen wie Veröffentlichungsdaten, Untertitel oder die im entsprechenden Issue abgedeckten Seitenanzahlen – welche für unsere Zwecke allesamt uninteressant sind, da sie sich einfacher von den jeweiligen Indexseiten auslesen lassen.

Jene Indexseiten listen die Inhalte des gewählten Volumes/Issues nach Seitennummern sortiert auf (siehe Abb. 3.5). Alle für unsere Zwecke relevanten Informationen (Titel, Autorennamen, Seitenzahlen) sind i.d.R. vollständig verfügbar, lediglich nach einer DOI sucht man hier vergebens. Letztere existiert zwar meist, wird jedoch nur auf den Abstract-Seiten der Artikel angezeigt. Kennt man jedoch eine der DOIs, so ist es möglich, alle weiteren durch Inkrementierung des letzten Wertes zu konstruieren; wird also beispielsweise der erste Artikel unter

`http://dx.doi.org/10.1016/j.entcs.2009.06.001`

gefunden, so ist zu vermuten – und im beispielhaften Fall trifft dies auch zu –, dass der folgende Artikel den DOI

`http://dx.doi.org/10.1016/j.entcs.2009.06.002`

trägt. Da dieses Verfahren sich jedoch nicht allgemein validieren lässt, wird der Wrapper von der automatischen Konstruktion absehen und die DOIs jeweils von den Abstract-Seiten lesen – was zwar zu einem erhöhten Zeitaufwand, dafür aber zu eindeutig korrekten Ergebnissen führen wird.

Die Qualität der bibliographischen Daten ist sehr hoch; Vornamen von Autoren sind nur selten durch Initiale abgekürzt, orthographische Fehler konnten nicht gefunden werden.

[10] `http://www.sciencedirect.com/science/journal/15710661`

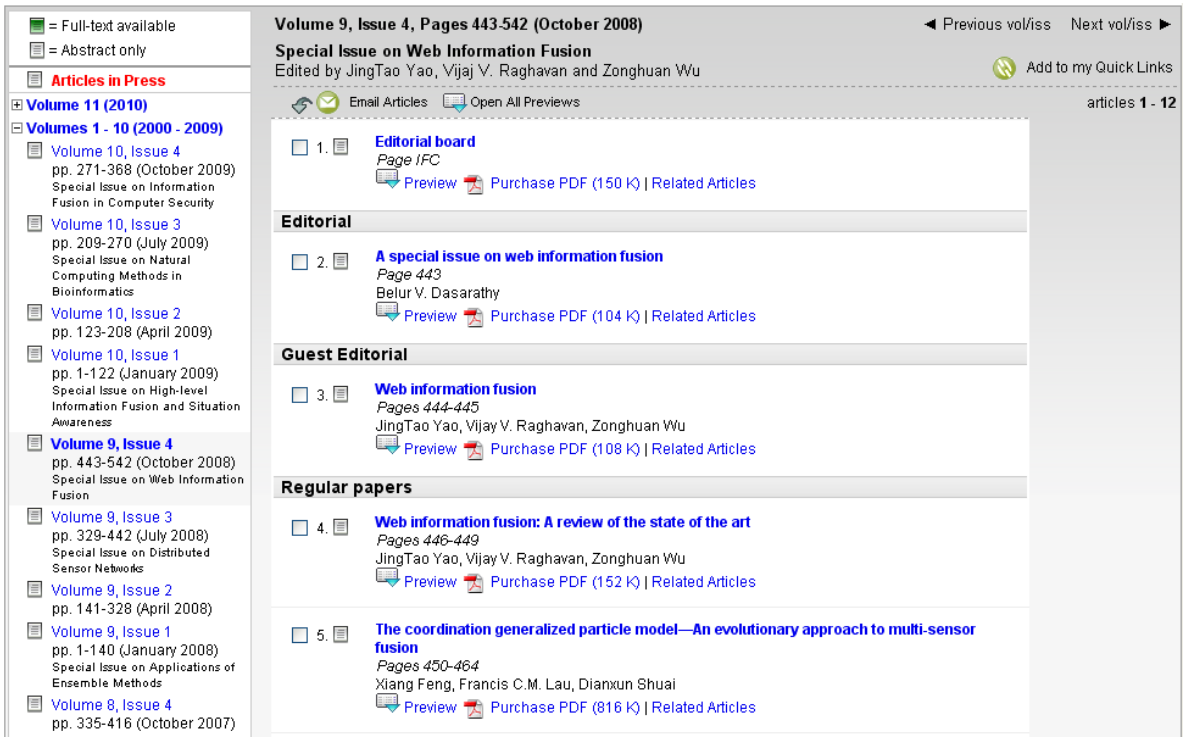


Abb. 3.5: 'ScienceDirect', Internetportal des Elsevier Verlags: Übersichtliche Navigation und hohe Datenqualität.
Quelle: `http://www.sciencedirect.com/science/journal/15662535`

Technische Details Das von 'ScienceDirect' übertragene HTTP-Protokoll lässt weder Rückschlüsse auf den Typ des Servers, noch auf die zur Generierung der Seiten verwendete Skriptsprache zu. Alle Seiten werden UTF-8-codiert und unter Angabe der 'content-length' übertragen. Sowohl Cookies als auch JavaScript werden genutzt, verursachen jedoch bei Deaktivierung keine kritischen Probleme. Den HTML-Seiten fehlt der vorgeschriebene DOCTYPE-Eintrag, bei der Validierung traten unter Annahme von 'HTML 4.01 Transitional' die wenigsten Fehler auf – allerdings lag dieser Wert noch immer im oberen Hunderterbereich. Der HTML-Code enthält in den meisten Fällen, jedoch nicht ausschließlich, reine Latin-1 Zeichen; die darüber hinaus gehenden Unicode-Sonderzeichen werden in Form von Entities übertragen, wobei sowohl benannte als auch numerische, von letzteren sowohl in dezimaler als auch in hexadezimaler Form, Entities benutzt werden.

Alle für das Durchsuchen der Seite benötigten Links entsprechen simplen HTML-Hyperlinks, was den Scanvorgang erleichtert. Die Anzahl der übertragenen Parameter ist oft recht hoch, was die Links äußerst lang werden lässt. Meist genügen jedoch einige wenige Angaben, um die entsprechende Seite mit allen relevanten Informationen dennoch laden zu können, auch wenn dies hin und wieder einige designerische Nachteile mit sich bringt, die für unsere Zwecke getrost vernachlässigt werden können.

Vorgehensweise des Wrappers Da die Seiten intern streng hierarchisch aufgebaut sind und jede Publikation in einem eigenen, mit ihrem jeweiligen Schlüssel benannten Unterverzeichnis abgelegt ist, benötigt der Wrapper als Eingabe lediglich einen URL, der eben jenen Verzeichnisnamen enthält. So verweist beispielsweise der URL

`http://www.sciencedirect.com/science/journal/03784754`

auf das Journal "Mathematics and Computers in Simulation", welches durch den Schlüssel "`03784754`" identifiziert wird. Danach muss der Wrapper eventuell einige Links des beschriebenen Menüs verfolgen, um die Informationen zum gewünschten Volume zu erhalten. Da es sich ausschließlich um Hyperlinks handelt, stellt dies kein Problem dar.

Auch das Sammeln der bibliographischen Daten erweist sich als äußerst einfach, da die Seiten einen einfachen Tabellenaufbau aufweisen und durchweg einheitlich gestaltet wurden. Die hohe Datenqualität sorgt auch bei der Aufbereitung der Daten für einen problemlosen Ablauf.

3.2.6 ICST / EUDL

Allgemeine Informationen Das "Institute for Computer Sciences, Social-Informatics and Telecommunications Enineering" (ICST) ist eine internationale, gemeinnützige Gesellschaft mit Sitz in Gent (Belgien), welche sich mit Themen der Informatik im weitesten Sinne beschäftigt und sich selbst als "an interface between Europe and the world" bezeichnet.

Neben zahlreichen anderen Aktivitäten ist das ICST Ausrichter zahlreicher Konferenzen in allen Teilen der Welt. Entsprechende Konferenzbände werden in Zusammenarbeit mit anderen großen Verlagen und Gesellschaften wie beispielsweise ACM, Cambridge University Press oder IEEE veröffentlicht. Bei Springer erscheint seit Anfang 2009 die Buchserie LNICST (Lecture Notes of the ICST)[11], eine Gemeinschaftsproduktion jener beiden Gesellschaften ([ICS09b]).

Neben diesen Veröffentlichungen auf fremden Servern betreibt das ICST auch eine eigene digitale Bibliothek, die "European Union Digital Library" (EUDL)[12]. In dieser sollen laut eigenen Angaben sämtliche Veröffentlichungen der ICST – also Konferenzbände, Zeitschriftenartikel, Buchkapitel der LNICST etc. – online verfügbar gemacht werden ([ICS09a]). Die EUDL ist derzeit in starkem Aufbau und beinhaltet bereits eine Vielzahl an Konferenzbänden. Dabei stellt die Bibliothek – ähnlich wie DBLP – jedoch nur Verweise zu jenen Artikeln her und beinhaltet nicht die kompletten Texte in Form von PDF-Dateien, wie dies bei anderen DLs oftmals der Fall ist. Die Abstract-Seiten der Artikel liegen i.d.R. auf den Servern der jeweiligen Partner (also z.B. bei IEEE Xplore, dem ACM Portal oder Springerlink).

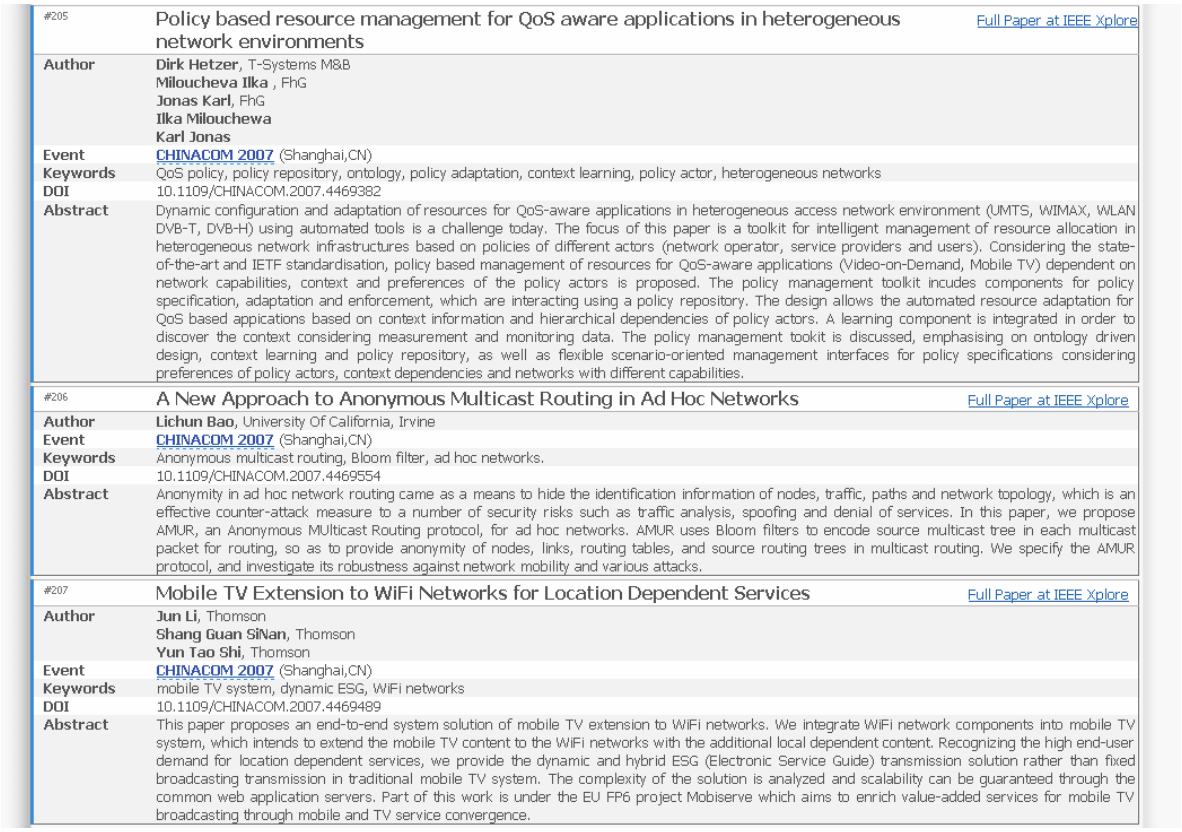
[11] `http://springerlink.metapress.com/content/j0m65u/`
[12] `http://eudl.eu`

Seitenstruktur und Datenbestand Sofern man das Setzen von Cookies gestattet – mehr hierzu im folgenden Abschnitt – gelangt man, ausgehend von der Startseite der EUDL, durch Wahl des Menüpunktes "BROWSE" zu einer Übersichtsseite aller online verfügbarer Konferenzen, welche teilweise ganzheitlich, teilweise durch Auflistung so genannter "sub-events" angezeigt werden. Hat man sich für eine Publikation entschieden, so öffnet ein Klick auf den entsprechenden Link ein neues Fenster (bzw. je nach Browsereinstellung eine neue Registerkarte), in welchem die erste Seite des Inhaltsverzeichnisses erscheint. Da bereits in dieser Ansicht zu jedem Artikel außer den gesuchten bibliographischen Daten auch ein Abstract-Text angezeigt wird, wurde die Anzahl der pro Seite gezeigten Einträge auf zehn begrenzt. Über eine Navigationszeile lässt sich jede weitere Inhaltsverzeichnis-Seite direkt anwählen – bei großen Konferenzen wie z.B. der "ChinaCom 2007" sind dies weit über zwanzig. Die einzelnen Einträge der Artikel enthalten den jeweiligen Titel und die Namen der Autoren, Jahresangaben lassen sich gewöhnlich direkt aus dem Konferenztitel ableiten. In den meisten Fällen sind DOIs verfügbar, die entweder auf Seiten innerhalb der EUDL, oftmals aber aus den eingangs genannten Gründen auf andere digitale Bibliotheken von ACM, IEEE, Springer u.a. verweisen. Ist kein DOI verfügbar, so findet man stattdessen entweder einen Link zur Abstract-Seite des Artikels, oder aber zu einer Suchmaske des jeweiligen Verlags, in welchen der gesuchte Titel eingetragen wurde. Da dies natürlich nicht die Art von Links ist, die wir in DBLP eintragen möchten, ist hier bei der Angabe der URLs stets äußerste Vorsicht geboten.

Ansonsten lassen Vollständigkeit und Qualität der Daten leider noch einiges zu Wünschen übrig. Seitennummern sucht man vergebens; statt dessen trägt jeder Artikel eine Nummer, die rein willkürlich vergeben zu sein scheint – zumindest ist keinerlei Sortierung, weder nach Titel, DOI etc., erkennbar. Äußerst mangelhaft ist auch die Qualität der Autorennamen bzgl. deren Korrektheit. Die gute Absicht, die Datenqualität zu erhöhen, ist klar erkennbar – so wurde meist versucht, möglichst vollständige Namen der Autoren anzugeben, bei asiatischen Namen wurde zudem oftmals der Familienname in Großbuchstaben geschrieben, um diesen eindeutig identifizieren zu können. Doch leider weisen die Autorenangaben zahlreiche unverständliche Fehler, wahrscheinlich durch automatische oder halbautomatische Datenerfassung bedingt, auf, die selbst fachlich unkundige Menschen bei einmaligem Korrekturlesen sofort bemerkt hätten. So findet man in zahlreichen Artikeln Paare von Autoren, die sich nur durch Vertauschung von Vor- und Nachname ergeben und bei denen es sich zweifellos um ein und dieselbe Person handelt (siehe Abb. 3.6: Hier sind sowohl "Jonas Karl" und "Karl Jonas", aber auch und vor allem "Ilka Milouchewa" und "Miloucheva Ilka" dieselben Personen).

Aber auch etwas kompliziertere Fälle sind bei genauem Hinsehen und ein wenig Recherche erkennbar: Gleich im ersten Artikel der Konferenz "MobiMedia 2007"[13] werden beispielsweise die beiden Autoren "Jose Gonzalez Arvelo" und "Jose David González" genannt, bei denen es sich ebenfalls um ein und dieselbe Person handelt, da der gleiche Artikel bereits bei ACM verfügbar ist[14] – auch wenn von EUDL kein entsprechender Link dorthin gesetzt wurde – und ebenfalls in DBLP gefunden werden kann.[15] Dort ist der Autor unter dem Namen "José D. González Arvelo" gelistet, was uns einen Hinweis auf seinen tatsächlichen, vollständigen Namen gibt: "José David González Arvelo". Dieses kleine Beispiel zeigt, dass es uns bei der Suche nach

[13] `http://eudl.eu/?eudlQuery=MOBIMEDIA%202007`
[14] `http://portal.acm.org/citation.cfm?id=1385289.1385305`
[15] `http://dblp.uni-trier.de/db/conf/mobimedia/mobimedia2007.html#ArveloA07`



#205	Policy based resource management for QoS aware applications in heterogeneous network environments — Full Paper at IEEE Xplore
Author	**Dirk Hetzer**, T-Systems M&B **Miloucheva Ilka** , FhG **Jonas Karl**, FhG **Ilka Milouchewa** **Karl Jonas**
Event	CHINACOM 2007 (Shanghai,CN)
Keywords	QoS policy, policy repository, ontology, policy adaptation, context learning, policy actor, heterogeneous networks
DOI	10.1109/CHINACOM.2007.4469382
Abstract	Dynamic configuration and adaptation of resources for QoS-aware applications in heterogeneous access network environment (UMTS, WIMAX, WLAN DVB-T, DVB-H) using automated tools is a challenge today. The focus of this paper is a toolkit for intelligent management of resource allocation in heterogeneous network infrastructures based on policies of different actors (network operator, service providers and users). Considering the state-of-the-art and IETF standardisation, policy based management of resources for QoS-aware applications (Video-on-Demand, Mobile TV) dependent on network capabilities, context and preferences of the policy actors is proposed. The policy management toolkit incudes components for policy specification, adaptation and enforcement, which are interacting using a policy repository. The design allows the automated resource adaptation for QoS based appications based on context information and hierarchical dependencies of policy actors. A learning component is integrated in order to discover the context considering measurement and monitoring data. The policy management tookit is discussed, emphasising on ontology driven design, context learning and policy repository, as well as flexible scenario-oriented management interfaces for policy specifications considering preferences of policy actors, context dependencies and networks with different capabilities.
#206	A New Approach to Anonymous Multicast Routing in Ad Hoc Networks — Full Paper at IEEE Xplore
Author	**Lichun Bao**, University Of California, Irvine
Event	CHINACOM 2007 (Shanghai,CN)
Keywords	Anonymous multicast routing, Bloom filter, ad hoc networks.
DOI	10.1109/CHINACOM.2007.4469554
Abstract	Anonymity in ad hoc network routing came as a means to hide the identification information of nodes, traffic, paths and network topology, which is an effective counter-attack measure to a number of security risks such as traffic analysis, spoofing and denial of services. In this paper, we propose AMUR, an Anonymous MUlticast Routing protocol, for ad hoc networks. AMUR uses Bloom filters to encode source multicast tree in each multicast packet for routing, so as to provide anonymity of nodes, links, routing tables, and source routing trees in multicast routing. We specify the AMUR protocol, and investigate its robustness against network mobility and various attacks.
#207	Mobile TV Extension to WiFi Networks for Location Dependent Services — Full Paper at IEEE Xplore
Author	**Jun Li**, Thomson **Shang Guan SiNan**, Thomson **Yun Tao Shi**, Thomson
Event	CHINACOM 2007 (Shanghai,CN)
Keywords	mobile TV system, dynamic ESG, WiFi networks
DOI	10.1109/CHINACOM.2007.4469489
Abstract	This paper proposes an end-to-end system solution of mobile TV extension to WiFi networks. We integrate WiFi network components into mobile TV system, which intends to extend the mobile TV content to the WiFi networks with the additional local dependent content. Recognizing the high end-user demand for location dependent services, we provide the dynamic and hybrid ESG (Electronic Service Guide) transmission solution rather than fixed broadcasting transmission in traditional mobile TV system. The complexity of the solution is analyzed and scalability can be guaranteed through the common web application servers. Part of this work is under the EU FP6 project Mobiserve which aims to enrich value-added services for mobile TV broadcasting through mobile and TV service convergence.

Abb. 3.6: EUDL, die digitale Bibliothek der ICST: Viele Datensätze weisen erhebliche und völlig offensichtliche Fehler auf (hier beispielsweise offensichtliche Namens-Dubletten im obersten Datensatz).
Quelle: `http://eudl.eu/?eudlQuery=CHINACOM%202007&type=8&page=21`

Fehlern im Datenbestand von EUDL gelungen ist, einen bereits bekannten aber unvollständigen Namen zu entschlüsseln – und eben dies ist das zweite große Ziel der vorliegenden Arbeit.

Technische Details Der Apache Webserver der EUDL läuft unter einem CentOS Linux System und überträgt mittels PHP generierte Webseiten im chunked Modus. Leider ist die Antwortzeit des Servers recht langsam, es kann mehrere Sekunden dauern bis eine Seite geladen wurde. Die Zeichencodierung ist UTF-8, und alle Sonderzeichen werden auch konsequent in Unicode angezeigt, wodurch vollständig (mit Ausnahme einzelner ` `-Entities zur Definition von Leerstellen, an denen kein Textumbruch stattfinden darf) auf Entities, gleich welcher Form, verzichtet wurde – leider jedoch auch auf die von XML vorgeschriebene Codierung der `&`-Zeichen mittels `&` innerhalb von Links, was auch, aber bei weitem nicht ausschließlich, dazu führt, dass die angeblich ‘xhtml 1.1’-konformen Seiten der Validierung nicht Stand halten konnten – wobei bemerkt werden sollte, dass sie deutlich weniger Fehler aufwiesen als der Großteil aller anderen Seiten der untersuchten Verlage.

Absolut verheerend ist jedoch der Gebrauch von Cookies, zu welchem ein Benutzer – und auch unser Wrapper – aufs Schärfste genötigt wird. Akzeptiert ein Browser keine Cookies, so

landet er in einer Endlosschleife, die heutzutage glücklicherweise erkannt und vom Browser vorzeitig beendet wird. Eine simple Anzeige der Seiteninhalte, selbst der Startseite, ist ohne die Annahme diverser Cookies nicht möglich. Weiterhin genügt es nicht, Cookies von Seite zu Seite einmalig zu akzeptieren, sondern diese müssen bei jedem weiteren Seitenaufruf komplett an den Server zurück übermittelt werden. Geschieht dies nicht, so ist eine Navigation durch die Indexseiten nicht möglich und man verweilt, ganz gleich welche Seite man gewählt hat, immer bei den ersten zehn Artikeln eines Bandes.

In Bezug auf andere Technologien wie beispielsweise JavaScript, hier meist in Verbindung zu AJAX, verhält sich EUDL äußerst vorbildlich: Eine Verweigerung der Ausführung führt hier zu keinen Problemen bzgl. der Datenverfügbarkeit. Links sind in Form normaler Hyperlinks realisiert und lassen sich leicht automatisch erkennen und verfolgen. Leider enthalten die URLs jedoch oftmals unschönerweise Leerzeichen, die vor der Weiterverwendung entsprechend codiert werden müssen – eine Unart, die u.a. dazu verpflichtet, den URL-Parameter bei Aufruf des Wrappers in Anführungszeichen zu setzen. Ansonsten kommen die Links mit einer minimalen Anzahl an Parametern – meist nur Konferenzname und Seitenzahl der Indexseite – aus.

Vorgehensweise des Wrappers Aufgrund der aggressiven Cookie-Politik ist der Wrapper gezwungen, diese nach Wunsch des Servers zu handhaben und auch abzuspeichern, da er ansonsten bei Annahme von n Indexseiten genau n-mal die gleichen zehn Datensätze erhielte ohne dies direkt zu bemerken. Durch die klare und einfache Struktur der HTML-Seiten ist die Erfassung der relevanten Daten selbst jedoch sehr einfach. Lediglich bei den URLs ist, wie zuvor beschrieben, Vorsicht geboten. Eine automatische Nachbearbeitung der Daten ist jedoch in vielen Fällen, beispielsweise beim oben geschilderten Problem der falschen Angaben von Autorennamen, von vornherein zum Scheitern verurteilt. Hier ist eine sorgfältige manuelle Überarbeitung des Ergebnisses erforderlich – oder aber der Einsatz der Merge-Software, mit welcher sich der zweite Teil dieser Arbeit beschäftigt. Es bleibt zu hoffen, dass das Team der EUDL in Zukunft mehr Zeit für die gewissenhafte Pflege der Daten aufwenden können wird.

3.2.7 IEEE / Xplore

Allgemeine Informationen Die im Jahr 1884 gegründete, gemeinnützige Organisation IEEE (ausgesprochen als “Eye-triple-E”) ist nach eigenen Angaben die weltweit führende Vereinigung im Bereich technischer Entwicklung. Ursprünglich war der Name ein Akronym für das “Institute of Electrical and Electronics Engineers”, doch da die Interessengebiete der Organisation sich weit darüber hinaus ausgebreitet haben, sollte IEEE – ähnlich wie im Falle von DBLP (vgl. Kapitel 2.1) – heute als Eigenname gesehen werden ([IEE09a]).

Ein Blick auf aktuelle Zahlen der IEEE zeigen deren immense Wichtigkeit im Bereich der Informatik und ähnlicher Forschungsgebiete: Mit mehr als 365.000 Mitglieder in über 160

Ländern fungiert sie als Dachorganisation 45 interner Gesellschaften und technischer Räte, von welchen die 'Computer Society'[16] die größte ist. Neben zahlreichen weiteren Aktivitäten tritt die IEEE als Veranstalter von jährlich etwa 900 Konferenzen auf. Die Publikationen der Organisation stellen fast ein Drittel der weltweit existierenden Fachliteratur der Bereiche Elektrotechnik, Informatik und Elektronik dar; ihre digitalen Bibliotheken umfassen mehr als zwei Millionen Artikel ([IEE09b]).

Xplore[17] ist die DL der Dachorganisation selbst, während die Computer Society eine eigene DL betreibt. Aus diversen Gründen, die in den DBLP-FAQ nachzulesen sind ([DBL09], "Why are many IEEE publications not listed in DBLP?"), werden derzeit keine Publikationen letztgenannter DL in DBLP erfasst. Wir werden uns daher bei unseren Betrachtungen auf die Datengewinnung aus Xplore beschränken und uns im Folgenden, wenn von "Daten bei IEEE" o.ä. die Rede ist, stets auf Xplore beziehen.

Neben einer Vielzahl relevanter Konferenzbände und Journale, deren Extraktion uns zunächst beschäftigen wird, werden wir uns auch im Kontext der Fusion eingehend mit jenen Daten auseinander setzen. Leider lässt deren Qualität oftmals stark zu wünschen übrig, und so werden wir in Kapitel 6 zunächst versuchen, diese durch Fusion mit Daten der EUDL (vgl. Abschnitt 3.2.6) aufzuwerten. Darüber hinaus werden wir in Kapitel 9 die Fusion mit einer unstrukturierten Quelle anhand von Daten aus Xplore durchführen.

Seitenstruktur und Datenbestand IEEE Xplore bietet sowohl eine alphabetisch sortierte Listendarstellung als auch eine Suchfunktion an, mit der nach relevanten Publikationen gesucht werden kann. Jede Publikation (Journal, Magazin, Konferenz, Buch) besitzt eine Übersichtsseite, auf der einige generelle Metadaten verfügbar sind. Bei Journalen oder Zeitungen ist von dieser Übersicht aus die Wahl des gewünschten Volumes/Issues möglich (siehe Abb. 3.7, linke Seite). Dies geschieht mittels zweier Pull-Down-Menüs, von welchen eines die verfügbaren Publikationsjahre, das andere alle im entsprechenden Jahr erschienenen Volumes und Issues anzeigt. Es ist somit nicht möglich, ein Volume direkt auszuwählen, sondern jeweils nur über dessen Erscheinungsjahr. Zwar kann dieses grob berechnet werden, doch es wird für den Wrapper stets nötig sein, mehrere Jahreszahlen durchzutesten, um alle Issues eines Volumes zuverlässig herauszufinden, da es denkbar ist, dass sich ein Volume über mehrere Jahre erstreckt. Einige Issues sind zudem in mehrere Teile aufgespalten. Ändert man die Jahreszahl (oberes Pull-Down-Menü), so wird automatisch mittels JavaScript Event-Handler die entsprechende Seite nachgeladen. Wählt man hingegen ein Volume/Issue aus dem zweiten Menü, so ist ein Klick auf "View Contents" notwendig, um das Inhaltsverzeichnis des entsprechenden Bandes anzeigen zu lassen. Wurde JavaScript im Browser deaktiviert, ist ein Wechsel zwischen mehreren Jahrgängen nicht mehr möglich – eine äußerst ärgerliche Tatsache, die durch den durcheinander geratenen Seiteninhalt (siehe folgenden Abschnitt) noch verstärkt wird. Hat man ein Inhaltsverzeichnis erreicht, so werden dort jeweils bis zu 25 Einträge aufgelistet. Weitere Seiten können über eine Navigationsleiste ausgewählt werden.

[16] `http://www.computer.org`
[17] `http://ieeexplore.ieee.org`

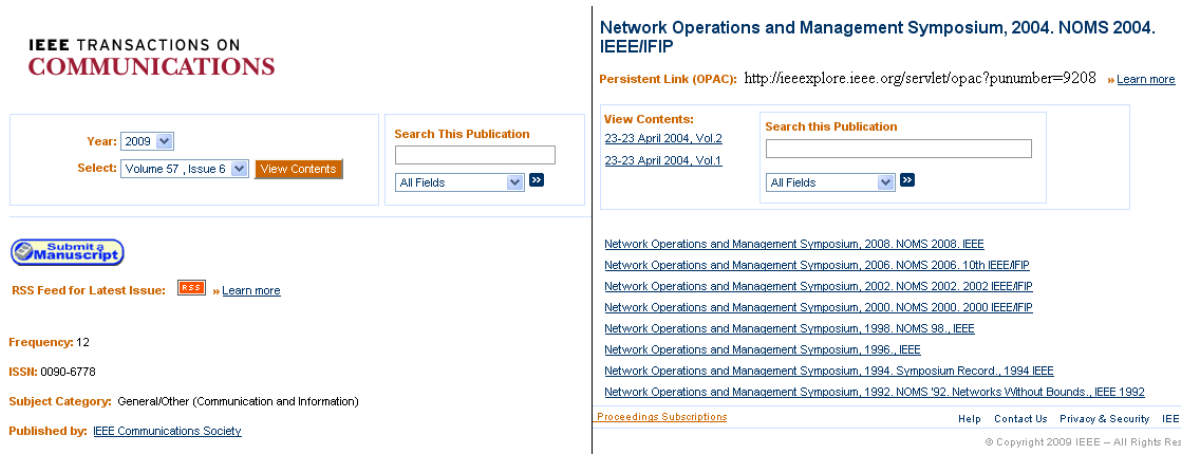


Abb. 3.7: Übersicht über Zeitschriften und Konferenzen bei IEEE Xplore: Links die Wahl eines Jahrgangs und entsprechender Volumes/Issues über Select-Menüs, rechts die mittels Hyperlinks realisierte Übersichtsseite eines Konferenzbandes.
Quellen: `http://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=26`, `http://ieeexplore.ieee.org/xpl/RecentCon.jsp?punumber=9208&conhome=1000491`

Bei den Konferenzen läuft die Wahl ein wenig anders ab. Die entsprechende Übersichtsseite listet alle entsprechenden Veranstaltungen, nach Jahren sortiert, in Form einfacher Links auf, wobei automatisch die Seite der aktuellsten Konferenz geladen wird (siehe Abb. 3.7, rechte Seite). Oftmals sind Konferenzen in mehrere Teile untergliedert, welche separat ausgewählt werden können. Auch bei nur einem Teil ist es notwendig, einen entsprechenden Link anzuklicken, um das Inhaltsverzeichnis zu erreichen.

Bei allen in IEEE Xplore enthaltenen Journalen und Konferenzen können die für DBLP relevanten Daten gewonnen werden. Einzig der Publikationsmonat ist nicht bei allen Einträgen vorhanden. Die syntaktische Qualität jener Daten lässt dagegen, wie bereits eingangs erwähnt, leider zu Wünschen übrig. Häufige Syntaxfehler und inkonsistente Beschreibungen der einzelnen Einträge stellen den Wrapper vor zahlreiche Probleme. Größter Nachteil ist jedoch, dass die Autorennamen häufig nicht vollständig sind, d.h. die Vornamen in vielen Fällen nur als Initiale angegeben sind. Dies erschwert die Verknüpfung der Publikationen erheblich und macht meist eine manuelle Nachbearbeitung unerlässlich. Auf der bereits eingangs zitierten FAQ-Seite der DBLP sind einige Möglichkeiten beschrieben, die Daten entsprechend aufzuarbeiten – die natürlich nicht nur für die Daten aus IEEE Xplore zutreffen, hier allerdings meist von größter Notwendigkeit sind.

Auch wenn die syntaktische Datenqualität oftmals erheblich zu wünschen übrig lässt, so ist und bleibt IEEE Xplore eine äußerst wichtige Datenquelle. Inhaltlich findet man Publikationen bis zurück in die 70'er Jahre. Sehr viele alte Dokumente wurden nachträglich digitalisiert und können bei Bedarf abgefragt werden – auch wenn die Relevanz solch alter Daten für DBLP eher gering ist.

Technische Details Der Webserver liefert die Kennung "IEEE Webserver", was keine Schlüsse auf dessen technische Struktur zulässt. Er überträgt die mittels JSP generierten dynamischen Seiten im 'chunked'-Modus. Außerdem versucht er, diverse Cookies zu setzen, was jedoch für die uneingeschränkte Navigation nicht erforderlich ist. Die übertragenen Seiten, die nach Angaben der DOCTYPE Definition der HTML 4.01 Transitional DTD genügen sollten – was jedoch nicht der Fall ist – enthalten zudem, wie bereits im vorherigen Abschnitt beschrieben, eine mittels JavaScript umgesetzte Navigation. Ist JavaScript deaktiviert, werden die entsprechenden Menüpunkte an den Anfang der Seite gesetzt, was einem menschlichen Besucher als äußerst unansehnlich erscheinen dürfte. Die Auswahl eines Jahrgangs der Journale ist nicht möglich, da das entsprechende HTML-`<select>`-Element mittels eines Event-Handlers bei Änderung einige Variablen eines Formulars mittels POST-Methode an den Server absendet.

In Bezug auf die Zeichencodierung ist der Server ebenfalls inkonsequent und liefert sowohl nach ISO-8859-1 erlaubte Sonderzeichen, als auch benannte oder numerische Entities und an manchen Stellen sogar kleine Passagen mit Codierungen nach Konvention von TEX/LATEX bzw. BIBTEX. All jene Zeichen müssen natürlich geeignet umgewandelt werden. Zudem enthalten vor allem die Titel häufig spezielle HTML-/XML-Tags, die entfernt werden müssen. Zwischen solchen Tags eingeschlossener Text wird dabei grundsätzlich erhalten und mittels Leerzeichen vom ihn umgebenden Text abgegrenzt.

Vorgehensweise des Wrappers Der Wrapper startet seine Arbeit nach Eingabe eines URLs. Er muss nun zunächst identifizieren, ob es sich dabei um ein Journal oder eine Konferenz handelt. Wurde die entsprechende Übersichtsseite angegeben, so ist dies problemlos möglich, da sich hier die URL klar unterscheiden (bei Journalen wird die Seite "RecentIssue.jsp" geladen, bei Konferenzen die Seite "RecentCon.jsp"). Wurde jedoch eine Inhaltsverzeichnisseite übermittelt, so ist eine Unterscheidung anhand des URLs nicht möglich, die geladene Seite trägt in beiden Fällen den Namen "tocresult.jsp", die übergebenen Parameternamen sind jeweils identisch. In diesem Fall muss der Wrapper in der Lage sein, anhand des HTML-Textes der Seite festzustellen, um welche Art der Publikation es sich handelt.

Um die o.g. Auswahl eines Jahrgangs bei den Journalen durchzuführen muss der Wrapper das Verhalten der JavaScript Event-Handler simulieren, indem er die entsprechenden Variablen an den URL anhängt und diese mittels GET überträgt. Da der Server nicht zu prüfen scheint, auf welche Art die Parameter übertragen wurden, stellt dieser Punkt kein Problem dar. Die Auswahl eines Volumes/Issues erfolgt auf gleiche Weise, da es sich hierbei ebenfalls um das Absenden von Formulardaten handelt. Alle übrigen Links wurden mittels einfacher Hyperlinks realisiert und lassen sich direkt aufrufen.

Die Inhaltslisten (sowohl der Bände eines Journals als auch der Artikel innerhalb eines Bandes/einer Nummer) sind oftmals sehr lang und über mehrere Seiten verteilt. Es muss darauf geachtet werden, alle entsprechenden Unterseite zu erfassen. Weiterhin wurden einige Bände – wahrscheinlich wegen deren Länge, oder aber wegen zeitlich versetzter Erscheinungsdaten – in Untergruppen zusammengefasst. So findet man nicht selten Bezeichnungen wie "Volume X, Number Y, Part Z". Es muss demnach darauf geachtet werden, dass bei Bearbeitung des entsprechenden Issues (im obigen Beispiel Y) alle evtl. existierenden Teile erfasst werden.

Auch das Sammeln der eigentlichen Daten ist nicht immer einfach. Wegen der bereits erwähnten, hin und wieder auftretenden Tippfehler und sonstigen syntaktischen Entgleisungen müssen die regulären Ausdrücke entsprechend flexibel gestaltet sein und auf alle möglichen Abweichungen von der Normalform reagieren können. Auch und vor allem die Namenskonventionen sind leider an vielen Stellen nicht einheitlich. So herrscht dort meist die Form 'Nachname, Vorname' vor, wobei einzelne Namen i.d.R. durch Semikola, manchmal – und das sind die kritischen Fälle – aber auch mittels Kommata voneinander abgetrennt werden. Auch kommt es teilweise vor, dass inmitten einer Liste von Namen ein einzelner Name – meist aus dem asiatischen Raum stammend – ohne trennendes Komma auftritt. Hierdurch wird sowohl die Identifizierung aller Namen erheblich erschwert, als auch die Zuordnung, ob es sich dabei um eine Änderung der Reihenfolge (d.h. 'Vorname Nachname') oder einen wie im asiatischen Raum üblich vorangestellten Nachnamen handelt. Zudem kann eine mittels Komma nachgestellte Nummer (z.B. "Hirsebrot, Herbert, III") in manchen Fällen als neuer Nachname aufgefasst werden und wird somit u.U. falsch aufgelöst.

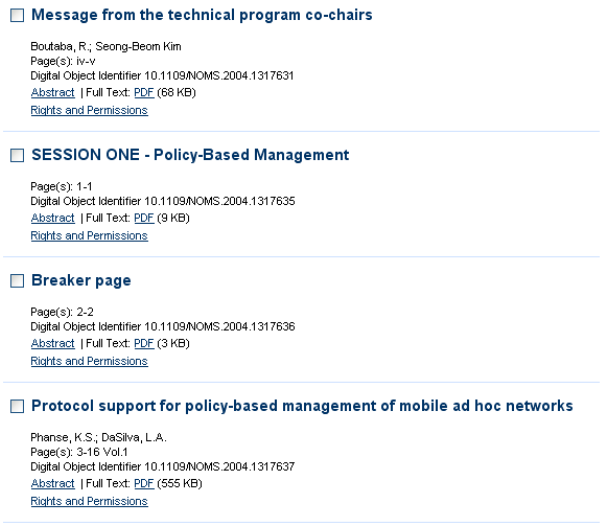


Abb. 3.8: Zwischenüberschriften bei IEEE Xplore: Rein syntaktisch im HTML-Text nicht von normalen Artikeln zu unterscheiden.
Quelle: `http://ieeexplore.ieee.org/xpl/tocresult.jsp?isnumber=29204&isYear=2004`

Abschließend soll hier noch erwähnt werden, dass es bei einigen Konferenzen möglich ist, Zwischenüberschriften zu identifizieren. Diese erscheinen auf den ersten Blick wie normale Artikel, da sowohl HTML-Code als auch Layout unverändert sind (vgl. Abb. 3.8), zeichnen sich jedoch durch einige Besonderheiten aus:

- sie beginnen meist mit einem Schlüsselwort, oftmals in Großbuchstaben, wie beispielsweise "SESSION"
- sie enthalten keine Autoreninformation
- sie erstrecken sich nur über eine Seite

- sie erscheinen meist vor einer leeren Seite, benannt als “Breaker page”, die ebenfalls keine Autorennamen enthält

Treffen mehrere dieser Annahmen zu, so wird der entsprechende Artikel als Zwischenüberschrift deklariert und entsprechend vom Wrapper verarbeitet.

3.2.8 IGI-Global

Allgemeine Informationen Bei IGI-Global[18] handelt es sich um eine 1987 gegründete Verlagsgesellschaft mit Hauptsitz in Hershey, Pennsylvania, die im Bereich der Informatik publiziert ([IGI09]). Das Akronym ‘IGI’ stammt, wie bereits im Kontext der DOIs in Kapitel 2.3.4 beschrieben, vom früheren Namen der Gesellschaft (“Idea Group Incorporated”), der Ende Juni 2009 zu “IGI-Global” geändert wurde. Welche Gründe dies hatte, konnte weder auf dem Server des Verlags noch aus anderen Quellen in Erfahrung gebracht werden.

Sämtliche Veröffentlichungen des Verlags – dies sind nach eigenen Angaben mehrere hundert Bücher jährlich, zahlreiche Konferenzprogramme sowie bis Ende 2009 über hundert Zeitschriftenserien – sind in dessen kommerzieller Volltext-Datenbank InfoSci-Online[19] online verfügbar. ([IGI09]) Bei dieser ist jedoch eine Anmeldung mittels eines kostenpflichtigen Accounts nötig, weshalb diese für unsere Zwecke ungeeignet ist. Die Inhaltsverzeichnisse der meisten Publikationen sind jedoch unter der `igi-global`-Domain verfügbar, wenn auch nicht in allen Fällen lückenlos. Zwar finden sich hier auch zahlreiche Bücher und Konferenzbände, doch wollen wir uns für die Extraktion auf die Journale beschränken, von denen derzeit zehn Serien in DBLP erfasst sind.

Seitenstruktur und Datenbestand Im globalen Verzeichnis aller Journale,[20] welches sich über mehrere Seiten erstreckt, werden alle verfügbaren Titel – zusammen mit einem Bild des Covers und einer kurzen Erläuterung des Forschungsinhalts – aufgeführt. Jedes Journal besitzt eine eigene Übersichtsseite, auf welcher der letzte Menüpunkt des Hauptmenüs auf der linken Seite zur “Table of Contents” führt. Hier findet sich eine chronologisch sortierte Liste aller Volumes und Issues, die derzeit online verfügbar sind. Ein entsprechender weiterer Klick führt zur Indexseite des betreffenden Bandes.

In der Regel enthalten die Einträge alle für unsere Zwecke relevanten Daten. Das Veröffentlichungsjahr, nicht aber der Monat, kann aus der Übersicht gewonnen werden. Eine Angabe der entsprechenden Seiten (von... bis) ist in den meisten Fällen vorhanden. Bei einigen Journals ist jedoch nur eine Angabe der Seitenanzahl (1-x, wenn sich der Artikel über x seiten erstreckt) vorhanden. Alle Artikel werden sequentiell in chronologischer Reihenfolge angezeigt, Zwischenüberschriften können nicht gefunden werden.

[18] `http://www.igi-global.com`
[19] `http://www.infosci-online.com/`
[20] `http://www.igi-global.com/journals/`

IGI-Global verwendet durchgängig keine DOIs, sondern setzt die Links mittels eigener Identifikatoren zusammen:

`http://www.igi-global.com/journals/details.asp?id=XXX`.

Diese `id`s scheinen sehr robust zu sein, und es konnten bisher in keinem Fall fehlerhafte oder nicht mehr gültige Verweise gefunden werden. Entsprechende Links der ehemaligen Domain `igi-pub.com` werden korrekt auf die neue Domain umgeleitet; dennoch zeigt sich hier das in Kapitel 2.3.4 ausführlich beschriebene Problem des Verzichts auf DOIs.

3. Association Analysis of Alumni Giving: A Formal Concept Analysis
Pages: 17-32
Authors: Hashemi, Ray R.; Le Blanc,Louis A.; Bahrami, Azita A.; Bahar, Mahmood; Traywick, Bryan
Affiliations: Armstrong Atlantic State University, USA; Berry College, USA; Armstrong Atlantic State University, USA; Tabiet Moallem University, USA; Armstrong Atlantic State University, USA

5. Assumptions Underlying Agile Software-Development Processes
Pages: 62 - 87
Authors: Turk, Daniel; France. Robert; Rumpe, Bernhard
Affiliations: Colorado State University, USA; Colorado State University, USA; Braunschweig University of Technology, Germany

4. Traffic Responsive Signal Timing Plan Generation Based on Neural Network
Pages: 84-101
Authors: ul-Asar, Azzam; Ullah, M. Sadeeq; Wyne, Mudasser F.; Ahmed,J amal; ul-Hasnain, Riaz
Affiliations: University of Engineering & Technology, Pakistan; University of Peshawar, Pakistan; National University, USA; University of Peshawar, Pakistan; University of Engineering & Technology, Pakistan

4. Agile Workflow Technology and Case-Based Change Reuse for Long-Term Processes
Pages: 80-98
Authors: Minor, Mirjam; Tartakovski, Alexander; Schmalen, Daniel; Bergmann, Ralph
Affiliations: University of Tier, Germany; University of Tier, Germany; University of Tier, Germany

3. Applying Learner-Centered Design Principles to UML Sequence Diagrams
Pages: 25-47
Authors: VanderMeer, Debra; Dutta, kaushik
Affiliations: Florida International University, USA; Florida International University, USA

7. The WASP Framework: Bridging the Gap Between the Web of Systems, the Web of Services, and the Web of Semantics with Agent Technology
Pages: 68 - 82
Authors: Biskup, Thomas; Gomex, Jorge Marx; Rautenstrauch, Claus
Affiliations: Otto-von-Guericke University Magdeburg, Germany; Otto-von-Guericke University Magdeburg, Germany; Otto-von-Guericke University Magdeburg, Germany

Abb. 3.9: Datenfehler und -inkonsistenzen bei IGI-Global: Fehlende Leerstelle nach Komma, Punkt statt Komma, Leerstelle an falscher Position, Rechtschreibfehler: “Tier” statt “Trier”, fehlende Großschreibung, Tippfehler: “Gomex” statt “Gomez” (von oben nach unten).
Quellen: diverse Artikel der Zeitschriften
JDM (`http://www.igi-global.com/journals/details.asp?id=198`) und
IJIIT (`http://www.igi-global.com/journals/details.asp?id=4295`)

Die syntaktische Qualität der Daten ist leider völlig unzureichend und oftmals absolut miserabel. An vielen Stellen erhält man den Eindruck, als seien die Daten unter Zeitdruck in freie Eingabefelder eingetragen und keiner syntaktischen Korrektur unterzogen worden. Vielfach treten simple Tippfehler auf (beispielsweise bei den Autorennamen “Chrisitan” statt “Christian” – ein Fehler, der mittlerweile auf der Website behoben wurde –, aber auch bei Ortsangaben “University of Tier” statt “University of Trier”, vgl. Abb. 3.9), die durch simples Korrekturlesen hätten vermieden werden können. Noch haarsträubender ist die Tatsache, dass das Format der Daten hin und wieder sogar innerhalb ein und desselben Issues wechselt, was vor allem bei den Autorennamen ein großes Problem bei der Datengewinnung darstellt: Manchmal werden die Autorennamen mittels ‘Nachname, Vorname’ angegeben, manchmal auch mittels ‘Vorna-

me Nachname'. Auch bei der Angabe der Seitennummern bzw. Seitenanzahl herrscht vielerorts Inkonsistenz.

Technische Details Der 'IGI-Global'-Server läuft auf einer Windows Plattform, identifiziert sich mittels "Microsoft-IIS/6.0" und übermittelt die Seiten im Latin-1-Standard. Zur Generierung der dynamischen Seiten wird auf ASP zurückgegriffen, einige JavaScript-Funktionen sind in die Seite integriert. Dies alles stellt jedoch kein Problem dar, ebenso wie die Cookies, die man getrost ablehnen kann, ohne Einbußen bei der Erreichbarkeit der Seiten hinnehmen zu müssen.

Die erzeugten HTML-Seiten tragen keinen DOCTYPE-Eintrag, Validierungsversuche mittels unterschiedlicher Typen schlugen allesamt fehl, was bereits bei einem kurzen Blick auf den Quellcode nicht weiter verwundert: Jedes Dokument beginnt grundsätzlich mit einem `<meta>`-Tag, welches an keiner Stelle geschlossen wird. Danach erst folgt das obligatorische `<html>`-Tag. Sämtliche Seiten sind demnach nicht einmal wohlgeformt.

Die meisten relevanten Informationen (Titel, Seitennummern, Autorennamen) stehen innerhalb einfacher `<div>`-Tags, denen nicht einmal eine CSS-Klasse zugeordnet wurde, und müssen somit weitestgehend anhand ihrer Position innerhalb eines Blockes identifiziert werden. Der Zeichensatz enthält sowohl benannte als auch numerische Entities. Sämtliche Links wurden als simple Hyperlinks realisiert und lassen sich somit leicht auffinden.

Vorgehensweise des Wrappers Größtes Problem bei IGI-Global ist die oftmals vorherrschende Inkonsistenz der Daten. An vielen Stellen findet man klare Tippfehler, vor allem bei den Autorennamen. Hier herrscht keine klare Konvention und Namenslisten werden sowohl in der Form 'Vorname Nachname' als auch in der Form 'Nachname, Vorname' gefunden. Als Trennzeichen zwischen mehreren Namen wird entweder ein Semikolon (im zweiten Fall sollte dies ausschließlich so sein) oder ein Komma verwandt, doch gerade im Fall 'Nachname, Vorname' wird hin und wieder statt des trennenden Semikolons fälschlicherweise ein Komma verwendet, was die korrekte Abgrenzung der Namen erschwert. Vor allem, wenn in einem Namen selbst ein Komma vorkommt (z.B. "Müller, Horst, Jr.") ist eine Zuordnung nicht mehr ohne Weiteres möglich. Bei mehr als einem Autoren werden die beiden letzten Namen häufig durch ein kaufmännisches Und-Zeichen (&) voneinander getrennt, wobei oft auch ein zusätzliches Semikolon oder Komma vor diesem Zeichen steht, aber manchmal eben nicht.

Ebenso existieren mindestens vier verschiedene Arten, die Seitenzahlangaben darzustellen ("Pages xx-xx", "# Pages xx-xx", "Pages pp. xx-xx", "# Pages pp xx-xx"), wobei hin und wieder auch nur die Anzahl der Seiten angegeben ist; man muss alle Seiten daher abschließend einem Test unterziehen, ob die vermeintlichen Seitennummern wirklich fortlaufend sind oder es sich lediglich um die jeweilige Anzahl an Seiten handelt – was sich vor allem bei kleinen Issues oft nur schwerlich oder überhaupt nicht feststellen lässt.

Beim Sammeln der Daten muss daher sehr flexibel vorgegangen werden, die verwendeten regulären Ausdrücke müssen kleine, für einen menschlichen Leser sofort offensichtliche Schreib-

fehler (z.B. mehrfache oder fehlende Leerzeichen zwischen einzelnen Bereichen, fehlerhafte Satzzeichen) akzeptieren und dennoch korrekte Ergebnisse liefern. Einige der Fehler, beispielsweise der falsch geschriebene Name in Abb. 3.9 ("Gomex" statt "Gomez") können jedoch nicht automatisch erkannt werden.

3.2.9 Inderscience

Allgemeine Informationen "Inderscience Enterprises Limited" ist ein Verlag mit Sitz in Genf (Schweiz), der seit 1979 eine Reihe qualitativ hochwertiger internationaler Zeitschriften verschiedenster wissenschaftlicher Fachrichtungen – u.a. aus Bereichen der Informatik – verlegt ([Ind09]). DBLP verfügt derzeit über bibliographische Daten aus 40 verschiedenen Zeitschriften dieses Verlags.

Seitenstruktur und Datenbestand Es ist möglich, auf Publikationen jenes Verlags unter einer MetaPress-Domain [21] zuzugreifen (vgl. 3.2.10), jedoch besitzt Inderscience auch einen eigenen Server mit eigenem URL.[22] Unter beiden Adressen, deren Websites sich bzgl. ihrer grafischen Gestaltung ähneln, können die gleichen bibliographischen Daten der Journale betrachtet werden, jedoch verfügt der Inderscience-Server zudem über Buchinformationen und einige wenige Konferenz-Proceedings. Wir werden uns bei der Extraktion lediglich auf die Journale beschränken, wodurch sich die Frage stellt, von welcher der Domains die Daten gewonnen werden sollen. Bei genauer Betrachtung der Datensätze fällt jedoch auf, dass unter der MetaPress-Domain hin und wieder eine schlechtere Datenqualität vorherrscht, wie beispielsweise in Abb. 3.10 erkennbar. Dort werden unter der MetaPress-Domain (links) lediglich drei

Abb. 3.10: Daten bei `inderscience.metapress.com` und `www.inderscience.com`: Unter der MetaPress-Domain (links) werden nicht alle Autoren aufgeführt, zudem fehlt der DOI. Die Inderscience-Domain (rechts) beinhaltet daher vollständigere Informationen. *Quellen:* `http://inderscience.metapress.com/app/home/issue.asp?referrer=parent&backto=journal,1,10;browsepublicationsresults,145,286;`, `http://www.inderscience.com/browse/index.php?journalID=209&year=2009&vol=3&issue=3`

Autorennamen aufgelistet, während im gleichen Datensatz der Inderscience-Domain (rechts) sämtliche Autoren genannt sind.[23] Zudem fehlen bei `metapress.com` die DOIs völlig. Da der strukturelle Aufbau der MetaPress-Domain zudem keine Ähnlichkeit mit dem der anderen

[21] `http://inderscience.metapress.com`

[22] `http://www.inderscience.com`

[23] Auf `http://inderscience.metapress.com` wird durchgängig von dieser Strategie Gebrauch gemacht: Es werden maximal drei Autorennamen ausgeschrieben, weitere Namen werden durch "et al." ersetzt – auch wenn es sich dabei lediglich um einen weiteren Autoren handelt.

unter MetaPress gehosteten Verlagssites (MetaPress, IOS Press und Springer, siehe 3.2.10) aufweist und wir ohnehin einen eigenen Wrapper für Inderscience erstellen müssen, werden wir die MetaPress-Domain ignorieren und unsere Daten ausschließlich von `inderscience.com` beziehen.

Inderscience verfügt über eine übersichtliche Indexseite, auf welcher sämtliche publizierten Journale, deren Code (d.h. die `journalID`), sowie die Anzahl der bisher publizierten Artikel aufgelistet sind. Dies kann der Ausgangspunkt der Suche nach interessanten Zeitschriften sein:

`http://www.inderscience.com/browse/index.php`.

Von dort gelangt man zur Übersichtsseite des jeweiligen Journals, welche für uns vor allem einen wichtigen Bereich enthält: Eine komplette Liste sämtlicher verfügbarer Volumes und Issues. Die einzelnen Indexseiten sind übersichtlich nach einem klaren Muster aufgebaut und enthalten fast alle relevanten und verfügbaren Daten. Auf der rechten Seite werden zudem stets alle Volumes/Issues des gerade gewählten Journals angezeigt, was die Navigation erheblich erleichtert (siehe Abb. 3.11).

Für jedes Journal sind die üblichen relevanten Daten verfügbar. DOIs können entweder direkt im Index oder aber auf der Abstract-Seite gefunden werden, sofern solche verfügbar sind – was vor allem bei älteren Journalen nicht immer der Fall ist. Im Kopf jeder TOC-Seite finden sich die aktuellen Werte zu Volume, Issue und Publikationsjahr. Angaben zum Publikationsmonat sind oftmals, jedoch nicht immer, vorhanden. Von einigen Zeitschriften sind nicht alle Bände ab "Vol. 1" online verfügbar; i.d.R. sind die Bände jedoch ab der Jahrhundertwende (ca. 1999-2000) lückenlos vorhanden.

Technische Details Die Inderscience-Daten stammen von einem Apache/2.0.54-Server, aufgesetzt auf einem Debian Linux System. Zwar verwendet der Server den Latin-1-Zeichensatz, jedoch verzichtet er größtenteils auf solche Zeichen, die den normalen ASCII Zeichensatz überschreiten. Es scheint, als würde Inderscience selbst bereits alle Umlaute und Akzente innerhalb der Titel und Autorennamen eliminieren, denn solche werden innerhalb der Datensätze nicht gefunden.

Die Daten werden mittels 'chunked'-Modus versandt. Der Server selbst ist normalerweise problemlos erreichbar. Inderscience bedient sich zur dynamischen Generierung der Seiten der weit verbreiteten und frei erhältlichen Skriptsprache PHP. Zudem enthalten die Webseiten JavaScript-Blöcke, die jedoch nicht zur Anzeige der Inhalte notwendig sind. Bei der Kommunikation mit dem HTTP-Server setzt dieser diverse Cookies, die aber für unsere Zwecke ebenfalls keinerlei Relevanz besitzen.

Vorgehensweise des Wrappers Die Daten dieses Servers sind von hervorragender Qualität und bereiten dem Wrapper keine größeren Probleme. Es ist lediglich zu beachten, dass

Abb. 3.11: Vorbildliche Qualität und Übersichtlichkeit bei 'Inderscience': Alle relevanten Daten auf einen Blick, rechts zudem stets eine Navigation durch sämtliche Volumes/Issues des gewählten Journals.
Quelle: `http://www.inderscience.com/browse/index.php?journalID=67&year=2009&vol=3&issue=3`

oftmals mehrere Issues zu Gruppen zusammengefasst werden (beispielsweise: "Volume 12, Issue 1/2/3"). In einem solchen Fall sind stets alle enthaltenen Nummern aufgelistet, wodurch der entsprechende reguläre Ausdruck problemlos nach diesen suchen kann.

3.2.10 MetaPress, IOS Press und Springer

Allgemeine Informationen Bei "MetaPress" handelt es sich um den nach eigenen Angaben weltweit größten Content-Management-Service der Firma "EBSCO Industries, Inc."[24], der Verlagen einen Hosting-Service bietet, über welchen diese ihre Print-Produkte auf einfache Weise auch online zur Verfügung stellen und selbst administrieren können ([EBS09]).

[24] `http://www.ebscoind.com/`

Der MetaPress-Server beherbergt eine Reihe von Verlags-Sites, von welchen für die Informatik vor allem Inderscience, IOS Press und Springer interessant sind.[25] Es ist möglich, Ähnlichkeiten zwischen den Websites von MetaPress, Springer und IOS Press zu finden und gleiche Strategien bei allen drei Verlagen anzuwenden. Die Websites von Inderscience (siehe Abschnitt 3.2.9) dagegen weisen eine andere Struktur auf und wurden daher bereits separat beschrieben.

Die DLs o.g. Verlage werden unter der Domain `metapress.com` gehostet, jeweils in entsprechenden Subdomains:

- IOS Press: `http://iospress.metapress.com`
- Springer: `http://springerlink.metapress.com`

Bei IOS Press handelt es sich um ein in den Niederlanden (Amsterdam) ansässiges Verlagshaus, das seit seiner Gründung im Jahre 1987 internationale Zeitschriften und Bücher aus dem Bereich STM (Science, Technology, Medicine) publiziert. Seit 2006 sind alle Publikationen des Verlags online unter der o.g. Subdomain des MetaPress-Servers verfügbar ([IOS09]).

Wichtigster der genannten Verlage ist jedoch zweifellos der 1842 von Julius Springer, einem deutschen Buchhändler, gegründete Springer-Verlag. Das traditionsreiche Unternehmen, dessen Logo bereits seit 1881 den stilisierten Kopf einer Schachfigur – des 'Springers' – darstellt, wuchs im Laufe der Jahre zu einem internationalen Wissenschaftsverlag heran ([Sar92]). Heute gilt Springer nach Elsevier (vgl. Abschnitt 3.2.5) als zweitgrößter Verlag im Bereich STM und ist namensgebender Teil der aus etwa 60 Verlagen bestehenden Verlagsgruppe "Springer Science+Business Media" ([Spr09b]). Das Onlineportal "Springerlink" beinhaltet derzeit (September 2009) weit über 300.000 wissenschaftliche Artikel aus dem Bereich der Informatik.

Eine der bedeutendsten Publikationsserien für die Informatik stellen die "Lecture Notes in Computer Science" (LNCS) dar.[26] Diese seit 1973 beim Springer-Verlag erscheinende Buchserie (siehe Abb. 3.12, linke Seite) umfasst derzeit über 5700 Bände. Der am 12. September 2009 aktuellste Band trägt die Nummer 5813, jedoch werden die Bände oftmals nicht in der Reihenfolge ihrer Nummern veröffentlicht: Der zweit aktuellste Band trägt die Nummer 5806, der nächste die 5797 (vgl. Abb. 4.6 auf Seite 90). Veröffentlichte Bände sind jedoch stets sowohl als Printversion als auch online verfügbar sind. Monatlich werden nach Angaben des Verlags über 150.000 PDF-Volltextversionen einzelner Artikel der LNCS bei 'Springerlink' heruntergeladen ([Spr09a]).

DBLP ist darum bemüht, sämtliche Bände der LNCS-Reihe zu erfassen. Über eine eigens für diese Publikation strukturierte Übersichtsseite lässt sich jeder einzelne Band gezielt anwählen (siehe Abb. 3.12, rechte Seite). Im weiteren Verlauf dieser Arbeit werden wir uns noch mehrmals mit dieser Publikation beschäftigen, vor allem im Bereich der Informationsfusion, wo es darum gehen wird, alten Einträgen neue DOIs zuzuweisen (siehe hierzu Kapitel 6.1.4).

[25] Eine vollständige Übersicht aller durch MetaPress gehosteter Verlage findet sich unter `http://www.metapress.com/clients/`.

[26] `http://springerlink.metapress.com/content/105633`

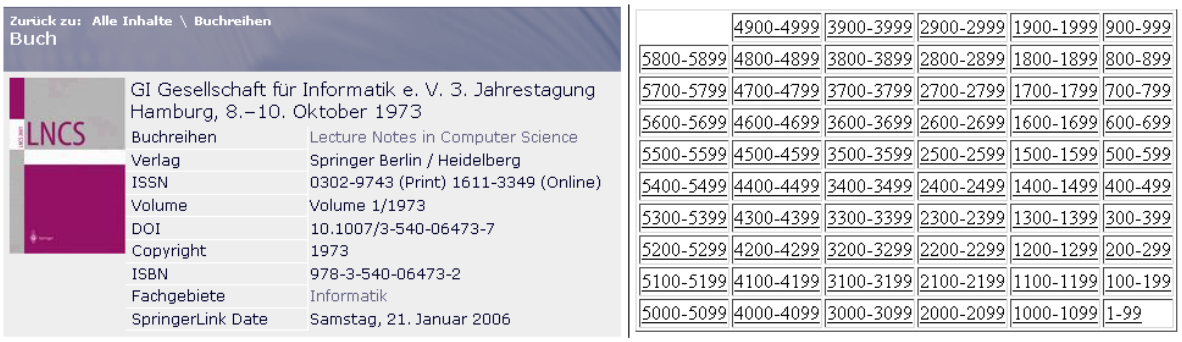


Abb. 3.12: LNCS bei Springer und in DBLP: Links die Daten des ersten LNCS-Bandes von 1973 bei Springer, rechts das Übersichtsmenü aller derzeit verfügbaren Bände in DBLP.
Quellen: `dx.doi.org/10.1007/3-540-06473-7`,
`http://dblp.uni-trier.de/db/journals/lncs.html`

Seitenstruktur und Datenbestand Sämtliche Publikationen der von MetaPress gehosteten Verlage lassen sich über die Hauptdomain (`http://www.metapress.com`) suchen. Die Publikationen einzelner Verlage – in unserem Falle IOS Press und Springer – sind jeweils innerhalb der entsprechenden Subdomains verfügbar. Wir werden daher einen Wrapper konstruieren, der die Informationen sowohl von den Subdomains der beiden genannten wichtigen Verlage, aber auch von der Hauptdomain beziehen kann, um so auch einzelne Publikationen kleinerer Verlage erfassen zu können.

Die Websites von IOS Press, Springer und MetaPress selbst sind nach dem gleichen Muster aufgebaut. Nachdem man, beginnend auf der jeweiligen Startseite, eine gewünschte Publikationsart und hiernach ein Journal oder einen Konferenzband ausgewählt hat, gelangt man zu entsprechenden TOC-Seiten, die sich bei allen drei genannten Verlagssites stark ähneln (siehe Abb. 3.13). Man kann jeweils eine von drei verschiedenen Sichtweisen auf die entsprechenden Daten auswählen:

1. Chronologische Sortierung nach Seitenzahl

2. Listendarstellung, alphabetisch nach Titeln sortiert

3. Listendarstellung wie in 2., allerdings mit weiteren Informationen.

Während bei IOS Press die o.g. Schaltflächen mit den englischen Namen “Editorial View”, “Expanded List View” und “Condensed List View” versehen sind, erscheinen sie bei MetaPress und Springer als “Inhaltsverzeichnis”, “Beitragsliste” und “Erweiterte Beitragsliste”.[27] Funktionsweise und Art der Inhalte sind jedoch identisch. Prinzipiell interessiert uns hierbei nur die erste Sicht, in der jedoch i.d.R. keine DOIs angegeben sind – wohl aber in der erweiterten Beitragsliste.

[27] Es ist zu beachten, dass hier auch die Reihenfolge des zweiten und dritten Reiters vertauscht ist: In der englischen Version zeigt der zweite Reiter die erweiterte Liste, in der deutschen Version der dritte.

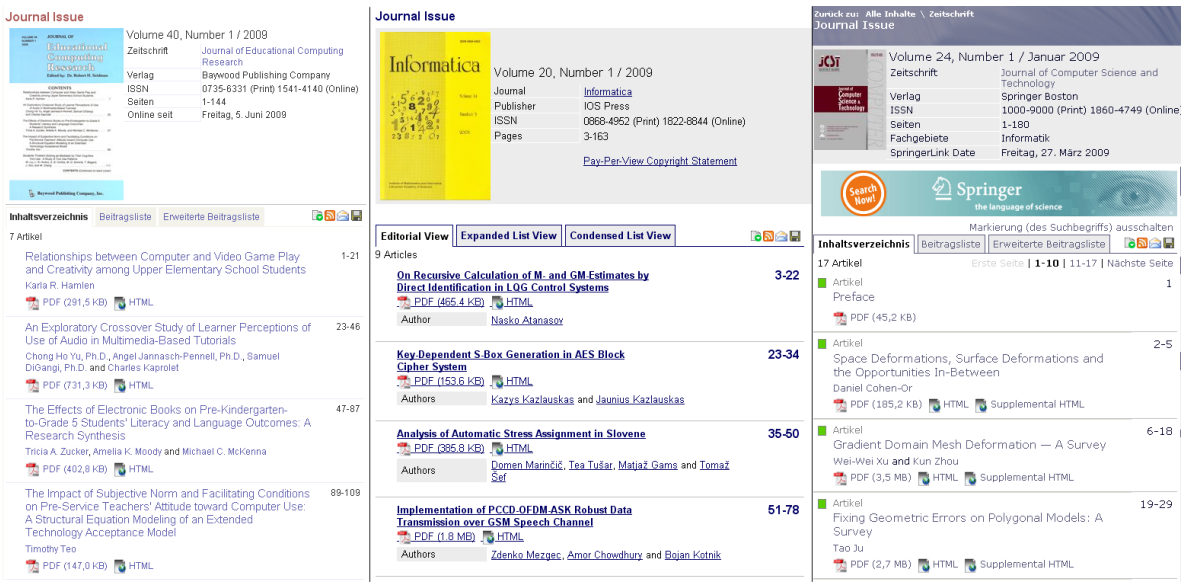


Abb. 3.13: Artikellisten bei MetaPress, IOS Press und Springerlink: Die drei unterschiedlichen Websites, die unter dem gleichen URL gehostet werden, weisen klare Ähnlichkeiten auf und können mit dem gleichen Wrapper bearbeitet werden.
Quellen: `http://www.metapress.com/content/j83476303307`,
`http://iospress.metapress.com/content/g58818260707`,
`http://springerlink.metapress.com/content/qvu817104317`

Da in beiden Listen entsprechende PDF-Dateien als Referenzen angegeben werden, ist eine Verknüpfung über diese möglich, wodurch eine 1:1-Zuordnung 'DOI:Titel' erzeugt werden kann. Allerdings müssen hierfür sämtliche (!) Seiten der erweiterten Sicht gelesen werden, auch wenn bloß ein einzelner Band benötigt wird, was vor allem bei großen Publikationen wie den LNCS einen unsäglichen Mehraufwand darstellt. Daher wurde bei der Umsetzung des Wrappers auf eine solche Vorgehensweise verzichtet; die DOIs werden – soweit vorhanden – von den jeweiligen Abstract-Seiten bezogen.

Jeder einzelne Band eines Journals, Buches oder einer Konferenz ist in einem separaten, mit einer verlagsweit eindeutigen Nummer versehenen Verzeichnis untergebracht. Diese Nummer dient demnach als Identifikator, beispielsweise `101497` für die Zeitschrift "Aequationes Mathematicae", welche unter dem URL `http://springerlink.metapress.com/content/101497/` zu finden ist. Enthält ein Verzeichnis mehr als zehn Einträge, so werden diese auf weiteren Seiten angezeigt, welche über ein Menü in der rechten oberen Ecke des Datenbereichs angewählt werden können.

Grundsätzlich sind bei allen drei untersuchten Websites die benötigten Daten wie Titel, Autorennamen, Seitennummern und Publikationsjahr vorhanden; lediglich der Publikationsmonat ist nicht immer bekannt. Bei Springer können wie bereits erwähnt in den meisten Fällen die DOIs über die Abstract-Seite bezogen werden, während IOS Press i.d.R. keine DOIs vergibt. Die syntaktische Qualität der Daten ist durchweg gut, und so fällt es recht leicht, reguläre Ausdrücke zu erstellen, die in der Lage sind, die benötigten Informationen auszulesen.

Technische Details Alle Websites werden von Servern übertragen, die sich als "Microsoft-IIS/6.0" identifizieren. Die Daten werden in UTF-8 codiert und unter Angabe der 'content-length' übermittelt. Zur Generierung der dynamischen Inhalte wird ASP verwandt; zudem wird JavaScript eingesetzt, dessen Bearbeitung allerdings nicht erforderlich ist. Auch die Cookies, die der/die Server zu setzen versucht/versuchen, können großzügig ignoriert werden, da sich für die uneingeschränkte Navigation und Datenabfrage keine Nachteile durch deren Ablehnung ergeben.

In den erhaltenen HTML-Seiten sind einige Entities (darunter auch sehr seltene Entities, die die Verarbeitung durch eine große Tabelle notwendig machen) enthalten, meist sind die Zeichen jedoch direkt in Unicode enthalten. Eine Umwandlung in die von DBLP unterstützten ASCII-Zeichen incl. die in der DTD definierten Entities ist daher dringend notwendig. Springer nutzt zudem einige dezimal codierte Zeichen (z.B. `&#8821;` für ein typographisches Anführungszeichen), die ebenfalls entsprechend verarbeitet werden. Die Seiten des Verlags IOS Press sind laut Angabe des DOCTYPES von der Form 'HTML 4.01 Transitional', hielten einer entsprechenden Validierung jedoch nicht stand und wiesen wie alle anderen Seiten der im gesamten Kapitel untersuchten Verlage erhebliche syntaktische Fehler auf. Die Seiten des Springer Verlags weisen ihren Typ als 'HTML 4.0 Transitional' aus; eine Validierung mittels `http://validator.w3c.org` war wegen Unkenntnis dieses Typs nicht direkt möglich; Versuche der Validierung mit ähnlichen Typen schlugen fehl.

Vorgehensweise des Wrappers Bei der automatischen Extraktion der Daten ist zu beachten, dass hin und wieder einzelne Issues zu Gruppen zusammengefasst werden. In einem solchen Falle wird ein Rang angegeben, beispielsweise "Volume 7, Issue 1-4". Dies wirft vor allem dann Probleme auf, wenn nur einzelne Issues gescannt werden sollen (z.B. "Volume 7, Issue 3" in obigem Beispiel).

Zudem sind bei einigen Artikeln von IOS Press die Namen der Autoren komplett in Großbuchstaben geschrieben, so dass eine sinnvolle Umwandlung auf Basis einiger einfacher Regeln getroffen werden muss. Die Ergebnisse dieser Umwandlung sind – anders als bei der Umwandlung von Titeln (vgl. hierzu beispielsweise Abschnitt 3.2.2: ACTA Press) – jedoch meist korrekt, da die Angabe allgemeingültiger Regeln (z.B. beginnen Personennamen stets mit einem Großbuchstaben) hier erheblich einfacher ist als bei den Titeln.

3.2.11 SIAM

Allgemeine Informationen Die "Society for Industrial and Applied Mathematics" (SIAM) wurde im Jahre 1952 als gemeinnützige Gesellschaft in Philadelphia gegründet, mit den Zielen der Förderung der mathematischen Forschung im industriellen und wirtschaftlichen Bereich, und als Medium zum Austausch von Informationen und Ideen mit anderen technischen und wissenschaftlichen Mitarbeitern. Bereits in den frühen 80'er Jahren wandte sich das Interesse der Gesellschaft den damals in der Anfangsphase ihrer Entwicklung stehenden Computern zu ([SIA02]).

Heute umfasst SIAM knapp 12.000 Mitglieder und 500 Organisationen weltweit. Derzeit erscheinen vierzehn Journale[28], deren Artikel vollständig und lückenlos in der SIAM-DL verfügbar sind. Vier von diesen sind für den Bereich der Informatik von besonderem Interesse und daher – einige vollständig, andere teilweise – in DBLP verfügbar.[29]

Seitenstruktur und Datenbestand Die Übersichtsseite der SIAM-DL (`http://epubs.siam.org/`) bildet den Einstiegspunkt zur Suche nach interessanten Journalen. Hat man sich für ein Journal entschieden, so lässt sich über den Link "Available volume list" eine große Übersichtsseite erreichen, die sämtliche Volumes und Issues eines Journals komplett darstellt. Jedes Issue eines Volumes ist auf einer einzelnen Seite untergebracht und kann über o.g. Menü leicht erreicht werden. Diese Seite enthält nun eine Auflistung aller enthaltenen Artikel, die jeweils durch eine Reihe von Verweisen ergänzt werden. Einer dieser Verweise gibt die Position einer BibTeX-Datei an, die in einheitlichem Format alle benötigten Informationen enthält. Da diese Informationen i.d.R. klarer strukturiert und einfacher lesbar sind als die Daten auf den HTML-Seiten, verwendet der Wrapper ausschließlich jene Informationen zur Gewinnung der Daten. Durch diese Taktik ist es allerdings notwendig, zu jedem gefundenen Datensatz die entsprechende BibTeX-Seite zu laden. Jedoch sind die BibTeX-Records, vor allem im Vergleich zu den textlastigen HTML-/XML-Dateien, äußerst klein und benötigen nur eine sehr kurze Übertragungszeit, die jedoch natürlich stark von der Antwortgeschwindigkeit des SIAM-Servers abhängt.

```
@article{patra\c{s}cu:730,
author = {Mihai Patra\c{s}cu and Mikkel Thorup},
collaboration = {},
title = {Higher Lower Bounds for Near-Neighbor and Further Rich Problems},
publisher = {SIAM},
year = {2009},
journal = {SIAM Journal on Computing},
volume = {39},
number = {2},
pages = {730-741},
keywords = {lower bounds; data structures; cell-probe complexity; nearest neighbor},
url = {http://link.aip.org/link/?SMJ/39/730/1},
doi = {10.1137/070684859}
}
```

Abb. 3.14: BibTeX-Records aus der DL der SIAM: Sonderzeichen im TeX/LaTeX-Stil müssen korrekt erkannt und evtl. ersetzt werden (in obigem Fall bezeichnet die Kombination "\c{s}" beispielsweise den Buchstaben "ş", welcher jedoch nicht zum ANSI-Zeichensatz gehört und daher in ein einfaches "s" umgewandelt werden muss). *Quelle:* `http://siamdl.aip.org/getabs/servlet/GetCitation?fn=view_bibtex2&source=scitation&SelectCheck=SMJCAT000039000002000730000001`

Die BibTeX-Einträge (vgl. Abb. 3.14) enthalten die meisten der benötigten Daten, sowie einige weitere, die nicht weiter verarbeitet werden. Lediglich ein Publikationsmonat ist dort nicht vorhanden. DOIs können in vielen Fällen gefunden werden; in allen anderen Fällen, in

[28] Ein weiteres, das "SIAM Journal on Financial Matehmatics" wird ab Winter 2009/2010 erscheinen, siehe `http://www.siam.org/journals/sifin.php`

[29] `http://dblp.uni-trier.de/db/journals/publ/siam.html`

denen der BibTeX-Record keine derartige Information enthält, ist diese auch nicht an anderer Stelle der Seite verfügbar und muss durch den URL zur Abstract-Seite ersetzt werden. Manchmal ist es möglich, den fehlenden Publikationsmonat von der Abstract-Seite zu lesen, was die Ausführungsgeschwindigkeit natürlich verlängert, da zu jedem einzelnen Eintrag nun auch noch das Abstract geladen werden muss. Die syntaktische Qualität der Daten ist recht gut, und so fällt es leicht, reguläre Ausdrücke zu erstellen, die die Daten in gewünschter Weise abfragen.

Technische Details Der SIAM-Server identifiziert sich als "Sun-Java-System-Web-Server" und überträgt die Zeichen gemäß der ISO-8859-1 Norm im 'chunked' Mode, so dass die einzelnen hexadezimalen Codierungen innerhalb der Übertragung korrekt gelesen und die nachfolgenden Blöcke korrekt berechnet werden müssen. Die einzelnen BibTeX-Seiten werden dagegen ohne speziellen HTTP-Header einfach in Klartext nach o.g. Norm übermittelt. Wegen der Verarbeitung der BibTeX-Records ist es notwendig, viele der nach TeX/LaTeX-Standard codierten Sonderzeichen korrekt aufzulösen, was ein spezieller `BibtexHandler übernimmt`. Durch die Vielzahl an Möglichkeiten, die TeX/LaTeX bieten, ist es jedoch nicht immer möglich, sämtlichen Code aus den Titeln zu entfernen – was auch auf den Seiten von SIAM selbst nicht geschieht, da die Titel in der Liste oder auf den Abstract-Seiten ebenfalls oftmals TeX-Codes enthalten.

Die HTML-Seiten geben über ihren DOCTYPE vor, vom Typ 'xhtml 1.0 Transitional' zu sein, doch eine entsprechende Validierung scheitert kläglich. SIAM benutzt serverseitig JSP und clientseitig JavaScript zur Generierung dynamischer Seiten. Beim Kontaktieren des Servers besteht dieser darauf, einige Cookies zu setzen, indem er den anfragenden Client mittels HTTP-Code 302 ("temporary moved") auf eine andere Seite umleitet. Dort werden dann die Cookies gesetzt, und eine weitere Umleitung zur Ausgangsseite findet statt. Ignoriert man das Setzen jener Cookies, so verfängt man sich in einer Art Endlosschleife, und wird wie ein Pingpongball immer wieder zwischen diesen beiden Seiten hin und her verwiesen. Man ist also gezwungen, die Cookies anzunehmen und bei der weiteren Kommunikation an den Server zu übermitteln. Erst dann kann auf die gewünschten Inhalte zugegriffen werden.

Vorgehensweise des Wrappers Problematisch, da anders als bei allen anderen untersuchten Servern, ist das Auslesen der BibTeX-Records, da diese zum einen korrekt geparst und die Sonderzeichen entsprechend decodiert werden müssen, zum anderen aber auch von der Software erkannt werden muss, wann ein solcher Record, der über keinen gewöhnlichen HTTP-Header verfügt, vorliegt. Ebenso bereitet die Notwendigkeit, die Cookies zu akzeptieren und zurückzusenden, einige Probleme. Ansonsten sorgen die gute Datenqualität, die für unsere Zwecke optimale Darstellung der Daten auf jeweils nur genau einer Seite und die Aufteilung in jeweils genau ein Volume mit einer festen Anzahl an Issues (ohne Gruppierungen oder Unterteilungen) dafür, dass so gut wie keine unvorhergesehenen Fälle auftreten.

3.2.12 World Scientific

Allgemeine Informationen Die im Jahre 1981 gegründete "World Scientific Publishing Company" hat sich nach eigenen Angaben von einem kleinen, nur fünf Mitarbeiter umfassenden Unternehmen zu einem der führenden wissenschaftlichen Verlage der Welt und dem größten internationalen wissenschaftlichen Verlag in der Asien-Pazifik-Region etabliert [Wor09]. In über 100 regelmäßig erscheinenden Journalen werden jährlich über 400 wissenschaftliche Artikel aus derzeit 19 verschiedenen Bereichen der Wissenschaft – darunter auch Informatik und Mathematik – veröffentlicht.

Seit Sommer 2009 zeigt sich der Webauftritt des Unternehmens in einem neuen, überarbeiteten Gewand. Die Seiten werden nun vollständig mittels der serverseitigen Skriptsprache SSI (Server Side Includes) generiert und stellen keine speziellen Anforderungen mehr an den Wrapper, die über jene der anderen Extraktionsquellen hinaus gehen. Wie man Abbildung 3.15 entnehmen kann änderte sich neben diesen technischen Aspekten und der offensichtlichen grafischen Umstrukturierung auch die logische Struktur der Seiten.[30] Eine Änderung der Daten konnte jedoch nicht festgestellt werden, weder in deren Vorhandensein noch bzgl. deren Qualität.

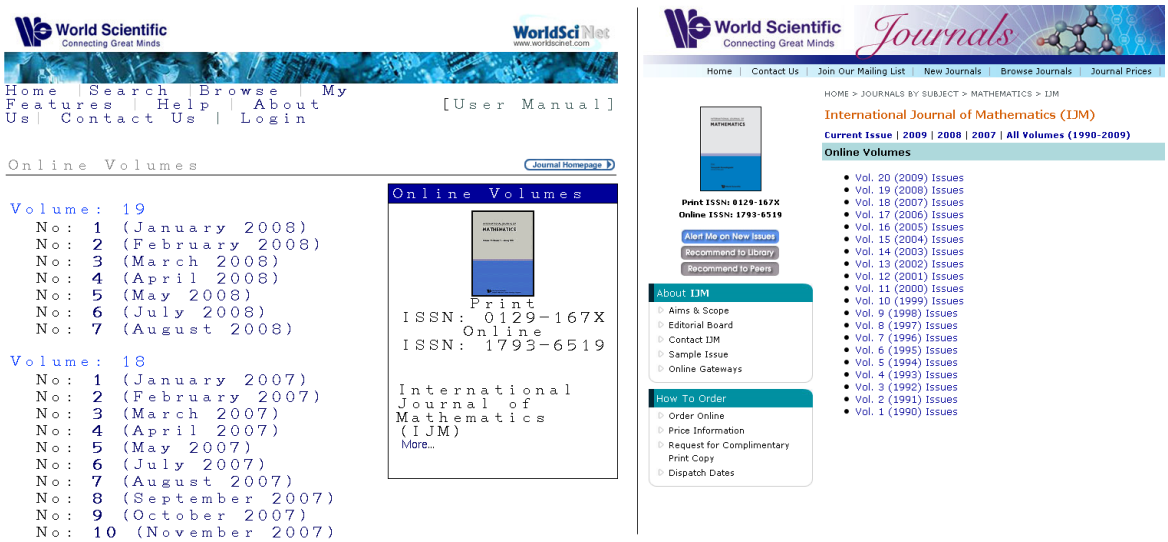


Abb. 3.15: Änderungen im Webauftritt der World Scientific P. C.: Die alte Website (links) stellte für den Wrapper eine große technische Herausforderung dar. Seit Sommer 2009 präsentiert sich die DL jedoch in grafisch, logisch und technisch überarbeiteter Form (rechts).
Quelle: `http://www.worldscinet.com/ijm/mkt/archive.shtml`
Screenshots aus dem Jahre 2008 (links) und September 2009 (rechts)

Da jedoch die alte Struktur wegen ihrer exotisch anmutenden technischen Umsetzung für unsere Zwecke äußerst interessant war, soll in diesem Abschnitt eben jene *alte* Struktur der Seiten beschrieben werden. Die meisten der Link-Angaben sind daher nicht mehr auf dem aktuellen

[30] Beide Abbildungen zeigen eine Übersicht der Volumes des "International Journal of Mathematics", jedoch waren in der alten Version die Issues auf der gleichen Seite aufgelistet, während nun ein weiterer Klick zur Issueliste notwendig ist.

Stand und rufen HTTP-Fehler hervor. Sämtliche Angaben dieses Abschnittes beziehen sich auf Anfang Mai 2009, da hier die alte Version noch existierte. Die dieser Arbeit beiliegende Software enthält jedoch lediglich den neuen, derzeit funktionstüchtigen Wrapper für die Inhalte der 'World Scientific'-DL.

Seitenstruktur und Datenbestand 'World Scientific' stellte – wie bereits erwähnt – technisch gesehen bis zum Sommer 2009 den absoluten Außenseiter dar. Dies machte ein in manchen Teilen völlig anderes Vorgehen bei der Suche nach relevanten Daten erforderlich. Für den menschlichen Besucher präsentierten sich die Seiten recht übersichtlich, wenngleich der ständige Wechsel des darstellenden Formates ein wenig verwirrend erschien. Der Server wies zudem äußerst hohe Antwortzeiten auf, was die Suche nach interessanten Daten u.U. etwas langwierig gestaltete.

Über die alphabetisch sortierte Journal-Indexseite[31] gelangte man mittels einer recht übersichtlichen Menüstruktur schnell zur Übersichtsseite des gesuchten Journals. Der entsprechende URL enthielt die notwendige `journalID` jeweils in zweifacher Form, da sowohl die HTML-Seite selbst, als auch das diese Seite beinhaltende Verzeichnis den gleichen Namen trugen, also beispielsweise (mit journalID = xxx):

`http://www.worldscinet.com/xxx/xxx.shtml`.

Dort musste man nun ein wenig suchen, bis man tief unten, am Ende der Seite (evtl. war es nötig, nach unten zu scrollen) den Link "Online Volumes" fand, der nun zur gesuchten Indexseite führte:

`http://www.worldscinet.com/cgi-bin/details.cgi?id=jsname:xxx&type=all`.

Wie bereits anhand der Links unschwer zu erkennen ist, änderte sich an dieser Stelle die interne Logik der Links, ebenso wie die zur Darstellung verwendete Skriptsprache, doch auch das generelle Layout dieser Übersichtsseite unterschied sich grundlegend von der vorherigen und auch aller von dort erreichbaren weiteren Seiten. Das gesamte Inhaltsverzeichnis des gewählten Journals wurde auf dieser einen, einzigen Seite, sortiert nach einzelnen Volumes und Issues, dargestellt und konnte augenscheinlich leicht abgerufen werden.

Folgte man nun einem der Links, so gelangte man zur Indexseite des gewählten Issues des entsprechenden Volumes. Abgesehen vom Seitenkopf, der nach wie vor die 'World Scientific'-Logos enthielt, änderte sich an dieser Stelle erneut das Layout, was vor allem an Schriftart und Darstellung der Daten auf der Seite auffiel. Der URL aller Seiten war identisch, da alle Daten über versteckte Formulare mittels POST verschickt wurden (siehe hierzu den folgenden Abschnitt):

`http://db0.worldscinet.com/worldsci-staging/toc.nsp`.

[31] `http://www.worldscinet.com/alphabetical.shtml`

An dieser Stelle hatten sich der Server geändert (die Subdomain `db0` lag, wie im folgenden Abschnitt beschrieben, auf einem völlig anderen Rechner), als auch das Dateiformat. Technisch gesehen wurde weiterhin das System der zuvor beschriebenen JavaScript-Funktionen und versteckten Formulare beibehalten. Da man auf der Indexseite nicht alle relevanten Daten fand (hauptsächlich fehlten dort DOIs), musste weiterhin die entsprechende Abstract-Seite geladen werden – nachdem man deren Link aus den JavaScript-Variablen heraus getüftelt hatte:

`http://db0.worldscinet.com/worldsci-staging/tocdetail.nsp`.

Zur Suche der Daten war es erforderlich, die o.g. Funktionalität zu umgehen, indem man die entsprechenden, aus den mittels JavaScript erstellten Formularen auszulesenden Variablen einfach in den URL einflickte und mittels GET übertrug.

Alle benötigten Daten waren – und sind auch in der neuen Version – enthalten, lediglich Publikationsmonat und DOI fehl(t)en in einzelnen Fällen. Die syntaktische Qualität der Daten ist recht gut, es konnten nur wenige Fehler oder Inkonsistenzen gefunden werden. Problematisch war und ist jedoch, dass in den meisten Fällen sowohl die Titel als auch die Autorennamen nur in Großbuchstaben vorhanden sind, was eine fehlerträchtige Nachbearbeitung notwendig macht(e).

Technische Details Wie zuvor beschrieben, nutzte 'World Scientific' (mindestens) zwei völlig verschiedene Server, einen Unix/Apache-Server, der die `www`-Subdomain beherbergte, sowie einen Microsoft IIS-Server, auf welchem die `db0`-Subdomain gehostet war. Ebenso variierten die serverseitigen Skriptsprachen zwischen JSP, CGI/SSI und NSP (s.o.). Nur bei der Nutzung von JavaScript waren die Seiten – wahrscheinlich lediglich in Ermangelung adäquater Alternativen – konsistent.

Technisch ergaben sich große Probleme, da die Site keine gewöhnlichen Links verwendete. Statt dessen waren alle Links mit entsprechenden JavaScript-Funktionen verknüpft, bei deren Aufruf (durch Klick auf den entsprechenden Hyperlink) diverse versteckte Formulare mit den entsprechenden Daten gefüllt und dann mittels JavaScript abgeschickt wurden, wodurch intern der Inhalt der verlinkten Seite berechnet wurde. Doch damit nicht genug: Alle textuellen Inhalte der Seite waren intern als JavaScript-Variablen realisiert, welche mittels entsprechender Funktionen an den korrekten Stellen positioniert wurden. Eine Suche im Quelltext musste demnach innerhalb der JavaScript-Variablen erfolgen; gefundene Daten mussten speziell behandelt werden, um diverse Steuerzeichen oder Escape-Sequenzen zu filtern. Abbildung 3.16 zeigt beispielhaft einen Ausschnitt aus dem vormaligen Quelltext der Webseiten.

Alle generierten Pseudo-HTML-Seiten verzichteten vollständig auf die Angabe eines DOCTYPES, und auch großzügig ausgelegte Validierungsversuche erbrachten – wie erwartet – völlig unzureichende Ergebnisse. Die Server lieferten keine Angabe zum übertragenen Zeichensatz. An einigen Stellen wurden Entities verwendet, es wurden aber auch dezimal oder hexadezimal codierte Zeichen gefunden. Wie bereits zuvor beschrieben, machten die Seiten regen Gebrauch von JavaScript. Der Wrapper konnte dies zwar wie oben erklärt umgehen, für einen Browser

```
<Script>
var vol="3";

document.write("<table border=0 bgcolor=#FFFFFF width=100%><tr><td><b>
document.write("Volume: " +vol + "</font>");

document.write("</b></td></tr></table>");


var issyr="1 (March 2000<br>2 (June 2000<br>3 (September 2000<br>4 (D
var spissyr=issyr.split("<br>");

document.write("<table border=0>");

for (var i=0;i<spissyr.length-1; i++) {
   var iss=spissyr[i].split("(");
document.write("<tr><td width=30> </td><td style='font-size: 11px
document.write("No: <b><a href=\"JavaScript:tocsubmit('" + vol +
document.write("(" + iss[1] + ")</a></font>");
document.write("</td>");
document.write("</tr>");

}

document.write("</table>");

</Script>
<br>
```

Abb. 3.16: Quellcode einer Webseite der World Scientific Publishing Company von Juli 2008: Die gesamte DOM-Struktur der Seite wurde mittels JavaScript aufgebaut, beim Klick auf einen Link wurden unsichtbare Formulare mit Daten gefüllt und mittels JavaScript abgesendet.
Quelle: `http://db0.worldscinet.com/worldsci-staging/direct-arc.nsp`, August 2008

war die Deaktivierung dieser immer noch als sicherheitskritisch betrachteten Sprache jedoch verheerend.

Weiterhin machte 'World Scientific' Gebrauch von der HTML Frame-Technik – wahrscheinlich um den gruseligen Quellcode vor Internetanfängern zu verbergen. Dies stellte für die direkte Programmierung eines maßgeschneiderten Wrappers jedoch kein Hindernis dar.

Vorgehensweise des Wrappers Wie bereits eingangs erwähnt hat die World Scientific Publishing Company ihre Webseiten im Sommer 2009 einer umfangreichen Überarbeitung unterzogen. Daher soll an dieser Stelle auf weitere Einzelheiten verzichtet werden, mit denen der vormalige Wrapper konfrontiert wurde, da diese zudem im vorherigen Text bereits an entsprechender Stelle erklärt wurden. Der derzeit aktuelle 'World Scientific'-Wrapper unterscheidet sich kaum von den meisten der übrigen Wrapper und bringt daher keine neuen Erkenntnisse mehr mit sich. Lediglich in Bezug auf Großschreibung der Autorennamen und Titel weist die DL noch immer Defizite auf.

3.2.13 Weitere Extraktionsquellen

Einige weitere Websites sollen an dieser Stelle nicht ausführlich beschrieben werden, da dort i.d.R. keine neuen Erkenntnisse, die zur Konstruktion der Wrapper-Software notwendig sind, gewonnen werden können. Diese Sites wurden jedoch ebenfalls untersucht, um anschließend

entsprechende Wrapper zu konstruieren. Eine tabellarische Aufstellung der entsprechenden Ergebnisse findet sich in Anhang D.

Für folgende Verlage und DLs wurden ebenfalls Wrapper erstellt:

- IEICE (`http://www.ieice.org`)
- informs (`http://www.informs.org`)[32]
- MIT Press (`http://mitpress.mit.edu`)
- Oxford University Press (`http://www.oxfordjournals.org/`)
- Revues online (`http://www.revuesonline.com/`)
- Sage (`http://www.sagepub.com`)
- Taylor & Francis (`http://www.informaworld.com`)
- Wiley (`http://www.interscience.wiley.com`)

[32]Die Konstruktion des Wrappers für die Domain `informs.org` wird in einem Tutorial in Anhang C ausführlich beschrieben.

Kapitel 4

Praktische Umsetzung der Wrapper-Software

Nachdem wir uns nun eingehend der Theorie gewidmet haben, soll in diesem Kapitel die praktische Umsetzung in Java dokumentiert werden. Zunächst werden in Abschnitt 4.1 zwei Anwendungsszenarien definiert, zu welchen wir die Software nutzen möchten. Anschließend wird in Abschnitt 4.2 die hierzu gewählte Methode der Informationsextraktion erläutert und begründet. Danach erfolgt eine detaillierte Beschreibung des Ablaufes der Extraktion bei der dieser Arbeit beiliegenden Software, zunächst in allgemeiner Beschreibung (Abschnitt 4.3.1) und anschließend anhand eines konkreten Beispiels (Abschnitt 4.3.2). Abschnitt 4.4 bietet einen Ausblick auf eine sinnvolle Ergänzung der bestehenden Software, um den Prozess der Gewinnung bibliographischer Informationen noch weiter zu automatisieren.

4.1 Anwendungsszenarien

In der Praxis ergeben sich zwei konkrete Szenarien, die sich ein wenig unterscheiden, jedoch mit Hilfe ein und derselben Strategie gelöst werden können.

4.1.1 Szenario E-1: Extraktion eines 'conference'-Bandes

Beschreibung Von einer der in Kapitel 3 vorgestellten Extraktionsquellen sollen Daten des Typs 'conference' extrahiert und in einer BHT_c-Datei ausgegeben werden (vgl. Abb 4.1).

Erläuterung Wie in Kapitel 2.2.2 definiert, bezieht sich der Typ 'conference' auf das resultierende Ausgabeformat. Es kann sich sowohl um einen Konferenzband als auch um ein Buch o.ä. handeln, bei welchem keine Unterteilung nach Volume und Issue erfolgen soll.

Als Datenquelle stehen uns eine oder mehrere HTML-Seiten eines Verlags bzw. einer DL im WWW zur Verfügung. Diese Seiten müssen gelesen und die relevanten Informationen von ihnen extrahiert und in BHT_c-Format konvertiert werden.

4.1.2 Szenario E-2: Extraktion eines oder mehrerer 'journal'-Bände

Beschreibung Von einer der Extraktionsquellen sollen Daten des Typs 'journal' extrahiert und in einer oder mehreren BHT_j-Dateien ausgegeben werden (vgl. Abb 4.2).

Erläuterung Grundsätzlich gleicht dieses Szenario stark dem zuvor beschriebenen. Dennoch bestehen zwei wesentliche Unterschiede: Zum einen im Typ der Ausgabe (BHT_j statt BHT_c), zum anderen – und dies ist der wesentliche Unterschied – wird das Ergebnis u.U. in mehreren Dateien ausgegeben. Der Wrapper wird in diesem Szenario iterativ jedes Volume einzeln extrahieren und pro Volume eine eigene Datei erstellen. Auf diese Weise wird es möglich sein, vollständige Heftreihen mit nur einem einzigen Befehl zu erfassen.[1]

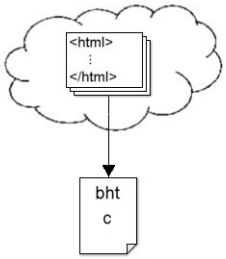


Abb. 4.1: Szenario E-1: Extraktion der Daten eines Konferenzbandes

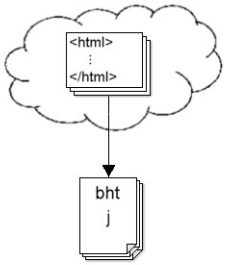


Abb. 4.2: Szenario E-2: Extraktion der Daten einer Zeitschrift

4.2 Wahl der Methode

Zur Extraktion der bibliographischen Informationen wurde eine entsprechende Software in Java implementiert. Diese lässt sich mittels des Kommandos "`get`" (siehe Anhang A.2.1) aufrufen. Über das Kommando `test` (siehe Anhang A.4.1) lässt sie sich umfangreichen Tests bzgl. aller in Kapitel 3 dokumentierter Extraktionsquellen unterziehen.

[1] Oftmals tragen auch Konferenzen Bandnummern, doch da i.d.R. jede einzelne Konferenz einen unterschiedlichen Konferenztitel besitzt, der meist zusätzliche Angaben wie Tagungsort oder inhaltliche Schwerpunkte beinhaltet, wird gemäß Szenario E-1 von einer automatischen Erfassung mehrerer Konferenzbände abgesehen, auch wenn dies in Einzelfällen durchaus sinnvoll sein könnte. Im Normalfall erscheinen neue Konferenzbände allerdings deutlich seltener als einzelne Hefte einer Zeitschrift.

Die Software arbeitet stets regelbasiert und wird daher gemäß der in Kapitel 1.6 erläuterten Definition als *Wrapper* bezeichnet. Der Wrapper arbeitet ausschließlich mit manuell codierten Regeln, denn trotz der klaren Nachteile, die dies mit sich bringt, scheint er für unsere Anforderungen dennoch geeignet, da hier ein äußerst spezifisches Problem in einer klar eingegrenzten Domäne gelöst werden soll. Zudem stellen die Extraktionsquellen die Software vor verschiedenste Anforderungen, deren allgemeingültige Lösung äußerst komplex wäre und dem Nutzen einer solchen Software in keiner Weise gerecht würde. Außerdem besteht das Ziel von DBLP gerade *nicht* darin, völlig automatisierte Datenerfassung zu betreiben, weshalb ein manuell codierter Wrapper einen idealen Kompromiss aus Aufwand und Nutzen bietet (vgl. Kapitel 1.6.1). Die Extraktionsregeln werden stets mittels regulärer Ausdrücke, wie sie die Programmiersprache Java zur Verfügung stellt, gebildet.

Die Software beinhaltet eine abstrakte Klasse mit Namen `BaseWrapper`, welche die prinzipielle Funktionalität zur Verfügung stellt, die bei allen Extraktionsquellen benötigt wird, wie beispielsweise die Steuerung der im folgenden Abschnitt 4.3 detailliert beschriebenen Vorgehensweise bei der Extraktion. Von dieser Basisklasse leiten sich spezielle Wrapper ab, wobei ein solcher für jeweils eine der genannten Quellen zuständig ist (`AcmWrapper`, `ActapressWrapper`, `BmcWrapper`, ⋯ , `WorldscientificWrapper`).

Jeder dieser speziellen Wrapper implementiert dabei die abstrakten Methoden der Elternklasse entsprechend den konkreten Anforderungen, die die jeweilige Quelle stellt. Dabei herrscht an manchen Stellen ein wenig Redundanz in Form gleicher oder ähnlicher Methoden, die in vielen der speziellen Wrapper vorkommen. Diese Redundanz ist jedoch beabsichtigt, da sie bei der Konstruktion eines Wrappers einen nötigen Freiraum zur Bewältigung 'ungewöhnlicher' Formate (vgl. WorldScientific in Kapitel 3.2.12) lässt.

Die Wrapper-Klassen kümmern sich hierbei um zwei prinzipielle Aufgaben: Die Suche relevanter HTML-Seiten, von welchen Daten extrahiert werden können, sowie die eigentliche Extraktion jener Daten. Sämtliche bibliographischen Daten werden dabei intern als Strings gespeichert, auch solche Werte, die meist numerisch sind, wie beispielsweise *volume* oder *issue*. Hin und wieder bestehen, wie in Kapitel 3 beschrieben, jene Angaben auch aus weiteren Werten, beispielsweise "`1a`", "`2/3`" oder "`S1`".

Wurden bibliographische Daten eines Artikels extrahiert, so werden diese in einem eigenen Objekt der Klasse (`DblpRecord`) abgespeichert. Alle diese Record-Objekte werden einem Objekt der Klasse `DblpList` hinzugefügt. Diese beiden Klassen implementieren das in Kapitel 2.4 definierte Datenmodell: Ein Objekt der Klasse `DblpList` beinhaltet also eine Liste von Objekten der Klasse `DblpRecord`, während jedes solche Record einzelne Attribute gemäß obigem Datenmodell enthält (`title`, `pages`, ⋯).

Die Normalisierung (vgl. Kapitel 2.2) der extrahierten Strings erfolgt *nicht* innerhalb der Wrapper-Klassen, sondern beim Eintrag in ein `DblpRecord`-Objekt. Der Wrapper kann also nahezu beliebige Daten an das Record-Objekt übergeben und muss sich nicht um syntaktische Eigenheiten der Server kümmern. Die Klasse `DblpRecord` dagegen stellt eine reichhaltige 'Toolbox' zur Konvertierung jener Daten in das gewünschte Format zur Verfügung, die jedoch gemäß den Grundlagen der objektorientierten Programmierung vollständig gekapselt ist. Dies

hat den immensen Vorteil, dass wir die gleichen Klassen auch bei der anschließenden Fusion (ab Kapitel 6) benutzen können und uns auch dort keinerlei Gedanken über Zeichencodierung oder Einhaltung bestimmter Konventionen machen müssen.

Intern werden die Strings, welche an ein `DblpRecord` übergeben werden, zunächst – unter Zuhilfenahme einer umfangreichen Ersetzungstabelle in XML-Format – vollständig nach UTF-8 konvertiert. Damit sind sämtliche zuvor beschriebenen Besonderheiten der Zeichencodierung verschiedener Extraktionsquellen hinfällig. Anschließend werden diese UTF-8-codierten Strings in ein zur DBLP-DTD konformes Zeichenformat transformiert. Zudem werden die Strings an dieser Stelle an die von DBLP geforderten Normen angepasst – teilweise während sie in UTF-8 codiert sind, teilweise auch erst später.

Ist die Extraktion einer HTML-Seite abgeschlossen, so enthält das `DblpList`-Objekt also sämtliche Daten in Form von `DblpRecord`-Objekten. Der Wrapper fordert diese an, indem er die öffentliche Methode `get()` des Listen-Objekts aufruft. Dieses liefert daraufhin einen String, der sämtliche Record-Daten bereits in gewünschtem $\mathrm{BHT}_{\mathrm{c/j}}$-Format enthält. Diese befinden sich normalerweise in der gleichen Reihenfolge, in welcher die Daten dem `DblpRecord`-Objekt hinzugefügt wurden. Falls die Seitenangaben der Objekte jedoch linear und unsortiert sind, so werden sie zuvor in Reihenfolge aufsteigender Seitennummern gebracht. Auch dies ist für den Wrapper also völlig transparent und wird uns ebenfalls bei der Fusion dienlich sein.

Eine ausführlichere Beschreibung der Software liefern Anhang B sowie die Dokumentation auf der beiliegenden CD-ROM.

4.3 Vorgehensweise der Wrapper

Nachdem die Anforderungen an die Software klar definiert sind, und wir wissen, in welcher Art diese umgesetzt wurde, soll nun deren Arbeitsweise bei der Informationsextraktion detailliert beschrieben werden. In Abschnitt 4.3.1 wird zunächst der genaue Ablauf dieses Vorgangs skizziert, Abschnitt 4.3.2 verdeutlicht jene Vorgänge anschließend anhand eines konkreten Beispiels.

4.3.1 Ablauf der Extraktion

Die Wrapper-Software wurde im Hinblick auf die eingangs definierten Szenarien E-1 und E-2, sowie die sich aus den von DBLP definierten Vorgaben (Kapitel 2) gestellten Anforderungen kreiert. Die Studie der Extraktionsquellen (Kapitel 3) lieferte eingehende Informationen zur Extraktion und Verarbeitung der Daten; die Ausgabe erfolgt in $\mathrm{BHT}_{\mathrm{c/j}}$-Format.

Um eine möglichst komfortable Handhabung von Seiten des Benutzers zu ermöglichen, soll der Programmaufruf recht einfach gehalten werden. Die Wrapper-Software (`get`, siehe Anhang

A.2) erwartet daher lediglich einen URL einer der zu bearbeitenden Webseiten. Anhand des Domainparts dieses URLs wird nun zunächst die Extraktionsquelle identifiziert, damit die jeweils passende Wrapper-Klasse instanziiert werden kann. Aus dem Localpart des URLs muss ein eindeutiger Identifikator, den wir *Publikationsschlüssel* (kurz: *publKey*) nennen werden, hervorgehen, der den jeweiligen Band eindeutig identifiziert. Wie dieser publKey aussieht, ist je nach Quelle äußerst verschieden; bei einigen Verlagen handelt es sich um einen Parameter, der im URL übertragen wird (z.B. "`http://...?id=12345&...`"), bei anderen wiederum werden die Daten durch das Dateisystem identifiziert, indem jeder Band (und evtl. auch jedes einzelne Heft) in einem speziellen Unterverzeichnis liegt (z.B. "`http://.../15/2/''` für "Volume 15, Issue 2"). Bei Journalen kann es demnach vorkommen, dass wir im Laufe der Extraktion mehrerer Volumes gemäß Szenario E-2 verschiedene Publikationsschlüssel ermitteln müssen, bzw. nicht den Schlüssel nehmen können, der direkt vom Benutzer eingegeben wurde.

Bei 'conferences' genügt uns die Angabe eines solchen URLs, da jede Konferenz einen eigenen *publKey* besitzt und damit eindeutig bestimmt werden kann. Möchten wir dagegen Hefte oder Bände eines 'journals' extrahieren, so benötigen wir weitere Angaben. Oftmals ist es gewünscht, nur einzelne Hefte zu erfassen – nämlich immer dann, wenn andere bereits schon erfasst sind. Sind die Issues 1 bis 6 eines Volumes bereits in DBLP eingetragen, so wäre es unsinnig, diese bei Erscheinen des siebten Heftes erneut zu extrahieren. Zudem müssten die Daten der Issues 1 bis 6 in der BHT_{j}-Datei manuell gelöscht werden, da eine doppelte Erfassung zu ernsthaften Problemen bei der Vergabe der Primärschlüssel innerhalb von DBLP führen könnte. Daher ist bei den Journals ein weiterer Eingabeparameter nötig, der das zu bearbeitende Volume/Issue beschreibt. Da es jedoch möglich sein soll, mehrere Hefte – oder sogar mehrere Bände – einer Zeitschrift gleichzeitig zu erfassen, ist auch die Angabe zweier Volume/Issue-Informationen möglich: eines Start- und eines Endwerts. Der Wrapper wird in diesem Fall alle möglichen Nummern einschließlich der Eingabewerte ermitteln und die entsprechenden Daten extrahieren. Wird als Start- bzw. Endwert nur ein Volume angegeben, so werden alle Issues dieses entsprechenden Volumes erfasst.

Abbildung 4.3 zeigt ein UML-Aktivitätsdiagramm, das den Ablauf des Extraktionsvorgangs beschreibt. Wir gehen dabei davon aus, dass die Software alle Eingabeparameter entsprechend der obigen Erklärung bei Programmaufruf erhalten hat. Natürlich muss in der Praxis zuerst eine Überprüfung auf deren Korrektheit erfolgen, die an dieser Stelle jedoch vorausgesetzt werden soll.

Insgesamt lassen sich die verschiedenen Aktivitäten in einzelne Gruppen unterteilen. Der große, hellgrau unterlegte Bereich stellt hier den *eigentlichen* Wrapper im Sinne der in Kapitel 1 beschriebenen Informationsextraktion dar. Die übrigen Aktivitäten sorgen lediglich für eine benutzerfreundliche Handhabung, die Erzeugung der Ergebnisdateien (in $\text{BHT}_{\text{c/j}}$-Format) sowie die automatische Extraktion mehrerer Volumes einer Zeitschrift gemäß Szenario E-2.

Die eigentlichen Wrapper-Aktivitäten lassen sich ebenfalls in spezielle Kategorien untergliedern. Die beiden dunkelgrauen Bereiche zeigen die Verarbeitung zweier verschiedener Arten von Daten: *globale* und *spezielle*. Die beiden Kommentarboxen zeigen, um welche Daten es sich hier jeweils handelt – eine detaillierte Erklärung erfolgt auf den Seiten 81 (globale Daten) bzw. 82 (spezielle Daten). Doch auch horizontal lassen sich, bedingt durch die Anordnung der Akti-

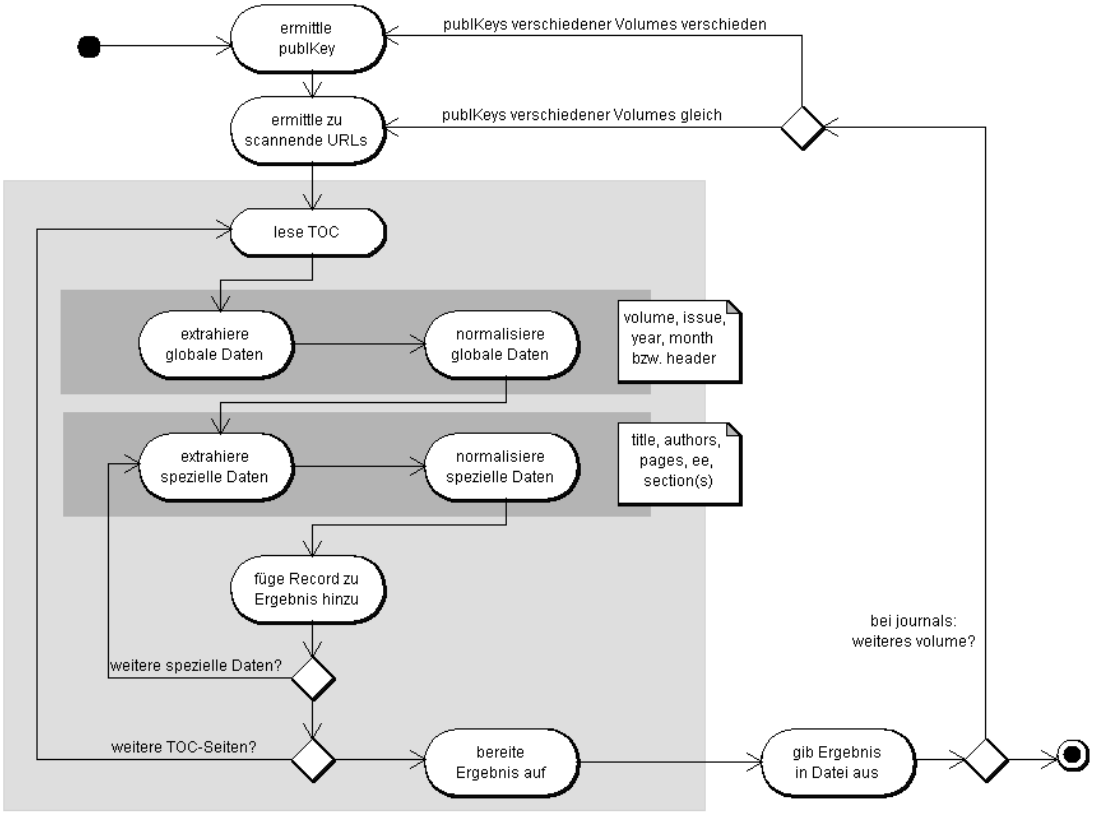


Abb. 4.3: Ablauf der Extraktion (UML Aktivitätsdiagramm)
Quelle: eigene Erstellung

vitäten, Kategorien feststellen. So dienen die beiden Aktivitäten auf der linken Seite der reinen Datenerfassung, während die Stationen auf der rechten Seite (Normalisierung und Aufbereitung der Daten) der Einhaltung der entsprechenden, in Kapitel 2.3 erläuterten, Konventionen und Qualitätsansprüche dienen.

Im Folgenden werden die einzelnen Stationen des Diagramms genauer erläutert. Für detailliertere Informationen zur Umsetzung innerhalb der Software sei auf Anhang B, bzw. die Softwaredokumentation auf der dieser Arbeit beiliegenden CD-ROM verwiesen.

ermittle publKey Zunächst muss der Publikationsschlüssel aus dem URL, den der Wrapper als Eingabe erhält, ermittelt werden. Dies erfolgt, wie bereits oben erwähnt, je nach Website, von der die Daten gelesen werden sollen, auf verschiedene Weise. Anschließend erfolgt eine Überprüfung (Validierung) dieses Schlüssels, indem ein URL mit Hilfe des *publKeys* generiert und jener über HTTP von der Quelle abgerufen wird. Ist der Schlüssel valide, so erhalten wir eine korrekte Webseite, andernfalls reagiert der Wrapper mit einer Fehlermeldung.

Scannen wir ein Journal, so kann es sein, dass wir an dieser Stelle den Schlüssel eines bestimmten Volumes herausfinden müssen, wenn dieser vom eingegebenen Wert abweicht. Dies kann

immer dann der Fall, wenn zur Identifikation eines Volumes die Verzeichnisstruktur genutzt wird (wie beispielsweise bei MetaPress, vgl. Kapitel 3.2.10), kann jedoch auch bei anderen Verlagen vorkommen, wenn diese jedem Band einen eigenen Schlüssel zuordnen (wie beispielsweise ACTA Press, vgl. Kapitel 3.2.2).

Um dem Benutzer eine direkte Möglichkeit der Kontrolle zu geben, wird nach Validierung des publKeys zudem der Titel der jeweiligen Publikation aus obiger Testseite ausgelesen (beispielsweise "International Journal of..." oder "International Conference on...") und als Meldung ausgegeben.

Es ist zu beachten, dass wir in diesem Schritt jeweils nur einen Konferenz- oder Zeitschriftenband (also ein *volume*) bearbeiten, d.h. wir gehen immer wie in Szenario E-1 vor. Sollen gemäß Szenario E-2 mehrere Volumes bearbeitet werden, so verfahren wir in einer Schleife iterativ vom kleinsten zum größten zu scannenden Band. Diese Schleife entspricht der letzten Entscheidung vor Erreichung des Endzustandes in Abbildung 4.3 ("*bei journals: weiteres volume?*").

ermittle zu scannende URLs Mit Hilfe des validierten *publKeys* ist die Software nun in der Lage, die Verbindung zu einer Webseite der gewünschten Publikation aufzubauen. Im einfachsten Fall handelt es sich dabei bereits um die einzige Seite mit relevanten Informationen, d.h. alle Daten des gewünschten Issues befinden sich auf der gleichen Seite. Manche Verlage/DLs wie z.B. BMC (vgl. Kapitel 3.2.3) oder Springer (vgl. Kapitel 3.2.10) stellen jedoch aufgrund der Fülle an Informationen nur einen Teil der Artikel auf einer Seite dar, d.h. es müssen mehrere Seiten bearbeitet werden, um alle Daten zu erfassen.

Ziel dieser Aktivität ist es nun, *alle* URLs, die im weiteren Verlauf des Extraktionsvorgangs gelesen werden sollen, in einer Liste zu sammeln. Dies birgt den entscheidenden Vorteil, dass wir im darauf folgenden, eigentlichen Extraktionsprozess nur noch eine Liste mit URLs abarbeiten müssen.

lese TOC Während die vorausgegangenen Aktivitäten eher organisatorischer Art waren, beginnt nun der eigentliche Extraktionsvorgang. Gemäß der zuvor erstellten URL-Liste wird der nächste zu bearbeitende URL via HTTP vom Server der Extraktionsquelle gelesen. Durch unsere Vorarbeit wissen wir, dass es sich hierbei um eine TOC-Seite handelt, welche zu extrahierende bibliographische Informationen enthält.

extrahiere globale Daten Zunächst werden wir versuchen, solche Daten zu lesen, die für alle Artikel der Seite von Belang sind. Dies sind bei 'journals' die Monats-, Jahres-, Volume- und evtl. Issueangaben; bei 'conferences' ist dies der (Konferenz-)Header. Diese Informationen sind meist im Kopfbereich der entsprechenden Seite zu finden.

Natürlich bestätigen auch hier Ausnahmen die Regel. Bei 'ACTA Press' (vgl Kapitel 3.2.2) werden beispielsweise alle Issues eines Volumes auf ein und derselben Seite angezeigt. Hier kann der Issue-Wert demnach nicht global für die gesamte Seite festgelegt werden. Bei anderen

Servern befinden sich einige der o.g. Angaben nicht auf der TOC-Seite eines Heftes, sondern müssen von anderer Stelle, beispielsweise von den entsprechenden Abstract-Seiten der einzelnen Artikel (z.B. die Angabe des Publikationsmonats bei SIAM, vgl Kapitel 3.2.11) gewonnen werden.

normalisiere globale Daten Wie wir in der Studie der Extraktionsquellen in Kapitel 3 gesehen haben, können die bibliographischen Daten in völlig unterschiedlicher Form und Zeichencodierung vorliegen. In diesem Schritt nun werden alle zuvor extrahierten Strings in eine einheitliche Form gebracht, indem sämtliche Sonderzeichen gemäß dem UTF-8-Standard umgewandelt werden. Demnach werden beispielsweise die Zeichenkombinationen "`&auml;`" (benanntes Entity), "`&#228;`" (numerisches Entity), "`&#xE4;`" (hexadezimales Entity) sowie "`\"{a}`" (BibTeX-codierung) allesamt in das Unicode-Zeichen "ä" verwandelt. Manche Server stellen Sonderzeichen jedoch in Form von Bildern dar, die in den HTML-Text eingefügt sind (siehe z.B. 'Cambridge' in Kapitel 3.2.4). Solche werden in diesem Schritt entweder – sofern möglich – gegen valide Zeichen, oder aber gegen eine Markierung, dass hier eine manuelle Nachbearbeitung notwendig ist, ersetzt.

Enthält ein String BibTeX-Elemente, so wird versucht, diese nach Möglichkeit durch alternative Elemente zu ersetzen. Die Zeichenkombination "`$a^b$`" beispielsweise wird in TeX/LaTeX dazu verwendet, eine Potenz (a^b) darzustellen. Diese wird in die zur DBLP-DTD konforme Form "`a<sup>b</sup>`" umgewandelt, was im Browser ebenfalls zu einem hochgestellten kleinen `b` führt.

Ebenso werden in diesem Schritt feste syntaktische Vorgaben bearbeitet. Beispielsweise wird an dieser Stelle das Datumsformat überprüft und ggf. korrigiert: Eine Jahreszahl sollte vierstellig sein; Monatsnamen sollten in englischer Sprache und vollständig ausgeschrieben werden, also beispielsweise werden "`Oct.`", "`Oktober`", "`10`" u.ä. Angaben allesamt in "`October`" verwandelt.

Wurden alle diesbezüglichen Änderungen durchgeführt, so wird der fertige, syntaktisch korrekte Wert in ASCII-Zeichen codiert, wobei sämtliche Nicht-ASCII-Zeichen (mit Zeichencodes ≥ 128) ersetzt werden müssen, um den in Kapitel 2.2 definierten Vorgaben zu genügen. Dies erfolgt gemäß einer komplexen Übersetzungstabelle entweder in DTD-konforme benannte Entities oder in möglichst *ähnliche* Zeichen. Ersteres ist bei einigen Zeichen, die in Latin-1 erlaubt sind, der Fall; so wird beispielsweise das Zeichen "ä" (Zeichencode 228) in die Zeichenfolge "`&auml;`", also das entsprechende, auch in HTML definierte benannte Entity, verwandelt. Ein "ă" (Zeichencode 462) dagegen zählt zur zweiten Kategorie und wird in ein normales kleines "a" verwandelt, ein griechischer Buchstabe wie "γ" in eine entsprechende Zeichenkombination – in diesem Falle "`gamma`".

extrahiere spezielle Daten Nun werden diejenigen Attribute, welche einen Artikel eindeutig beschreiben (also 'title', 'authors', 'pages' und soweit verfügbar 'ee') extrahiert und mit den obigen globalen Informationen zu einem *Record* zusammengefasst. Ein solches Record beinhaltet demnach alle relevanten Informationen zu einem einzelnen Artikel.

Zwischenüberschriften nehmen eine Sonderstellung zwischen globalen und speziellen Daten ein, da diese normalerweise für einige, nicht aber alle Artikel einer Seite gültig sind. Sie werden jedoch bei der sequentiellen Abarbeitung einer Seite zusammen mit den speziellen Daten erfasst, behalten dann aber – ähnlich den globalen Daten – bis zur nächsten gefundenen Überschrift ihren Wert.

Besitzt ein Abschnitt keinen Titel oder keine Angabe von Autoren, so vermuten wir, dass dieser Artikel nicht relevant ist und überspringen ihn, geben aber im Log eine entsprechende Meldung aus.

normalisiere spezielle Daten Bei der Normalisierung findet eine Zeichenersetzung analog der oben beschriebenen Ersetzung bei den globalen Daten statt. Die einzelnen Strings werden zunächst nach UTF-8 codiert, dann syntaktisch bearbeitet und schließlich in eine der DBLP-DTD entsprechende Form umgewandelt.

Bei der syntaktischen Aufbereitung werden die Autorennamen in eine korrekte Reihenfolge gebracht – Vorname(n) vor Nachnamen, kein Komma innerhalb eines Namens etc. (siehe Kapitel 2.3.2) – und die Syntax der Seitenzahlen überprüft, so dass diese einer der in Kapitel 2.3.3 beschriebenen Formen entsprechen.

Zudem werden Titel, Autorennamen und/oder Zwischenüberschriften, welche vollständig in Großbuchstaben geschrieben sind, an dieser Stelle in Groß- und Kleinschrift umgewandelt. Bei den Autorennamen geschieht dies mit Hilfe einfacher Regeln (Erster Buchstabe groß, restliche Buchstaben klein) und einiger weniger Ausnahmen ("MCDONALD" wird zu "McDonald", "O'BRIAN" wird zu "O'Brian" etc.). Bei Titeln und Zwischenüberschriften ist eine Angabe solcher Regeln nicht möglich. Stattdessen wird eine einfache Heuristik angewandt, mittels derer die jeweilige Schreibweise festgelegt wird: Aus den Titeln aller in DBLP eingetragener Datensätze wird die jeweils am häufigsten auftretende Schreibweise bzgl. Groß- und Kleinschreibung eines Wortes ermittelt. Wird das Wort jedoch nicht gefunden, so wird es wie im Fall der Autorennamen mit großem Anfangsbuchstaben geschrieben.[2] Diese Vorgehensweise führt zwar nicht in jedem Fall zu einem völlig zufrieden stellenden Ergebnis, doch werden durch die Anwendung obiger Heuristik zumindest häufig auftretende Wörter korrekt geschrieben ("AND" wird zu "and"), und gebräuchliche Akronyme werden korrekt in reiner Großschrift belassen ("DBLP" bleibt bestehen und wird nicht nach zweiter Regel in 'Dblp" verwandelt).

füge Record zu Ergebnis hinzu Unser Ergebnis ist eine zunächst leere Liste, in welche nun ein fertiger, in korrekter Syntax vorliegender Datensatz (das o.g. *Record*) eingefügt wird. Im Laufe des Extraktionsvorgangs erhalten wir so eine Liste von Records (R^L), die alle gefundenen Datensätze beinhaltet.

[2]Dies gilt selbstverständlich auch für Wörter, die am Satzanfang stehen; jene werden in jedem Fall mit großem Initial geschrieben.

Verzweigung: weitere spezielle Daten? Liegen weitere spezielle Daten vor (d.h. haben wir die Seite noch nicht komplett abgearbeitet und es sind weitere Artikelinformationen verfügbar), so fahren wir mit der Extraktion des nächsten Datensatzes fort.

Verzweigung: weitere TOC-Seiten? Nachdem alle Daten der Seite extrahiert wurden, müssen wir überprüfen, ob die Verarbeitung weiterer TOC-Seiten notwendig ist. Hierzu nutzen wir die Liste von URLs, die wir zu Beginn erstellt haben; enthält diese noch bisher ungenutzte URLs, so verzweigen wir entsprechend und bearbeiten nun die nächste Seite.

bereite Ergebnis auf Ist der komplette Extraktionsvorgang eines Konferenz- oder Zeitschriftenbandes abgeschlossen, so beinhaltet unsere Ergebnisliste eine Reihe von Records mit entsprechenden bibliographischen Daten.[3] Möglicherweise sind diese Daten jedoch nicht in der von uns gewünschten Sortierung (aufsteigend nach Seitennummern), was wir in diesem Schritt überprüfen und falls möglich korrigieren werden.

Die Daten liegen bereits in korrekter Zeichencodierung vor und müssen nun lediglich in das gewünschte Ausgabeformat (derzeit BHT, aber eine Ausgabe als XML-Dateien wurde ebenfalls vorbereitet) gebracht werden. Hierzu werden sie nun entsprechend der Formatregeln des Ausgabeformats angeordnet. Bei den Journals werden die Überschriften aus den Werten für 'monat', 'jahr', 'issue' und 'volume' nach festen Regeln zusammengesetzt. Überschriften und Zwischenüberschriften werden im BHT-Format nur an jenen Stellen ausgegeben, an denen ein Unterschied zum vorherigen Datensatz besteht.

gib Ergebnis in Datei aus An dieser Stelle liegt das fertige Ergebnis in Form eines einfachen Strings vor, der lediglich in eine Datei geschrieben wird. Dabei ist jedoch die Wahl des Dateinamens zu beachten, die sich über Optionen (vgl. die Softwaredokumentation in Anhang A.2.1) einstellen lässt. Wurde keine Einstellung seitens des Benutzers getroffen, so wird der Dateiname entsprechend des extrahierten Inhalts von der Software bestimmt. Der Name enthält bei Journalen i.d.R. die Bandnummer sowie, falls nicht bei Heft 1 begonnen wurde, die Heftnummer der bearbeiteten Publikation. Bei Konferenzen wird keine Nummer angefügt.

Existiert bereits eine Datei mit dem auf diese Weise konstruierten Titel, so wird jener entsprechend durch Ergänzung einer vierstelligen, fortlaufenden Nummer mit führenden Nullen modifiziert, so dass keine alten Dateien überschrieben werden können.

Verzweigung: bei journals: weiteres volume? Wie bereits oben erwähnt, wird jedes Volume einzeln bearbeitet, da auch für jedes Volume eine eigene Datei erstellt werden soll.

[3] Selbstverständlich ist es an dieser Stelle auch möglich, dass die Liste leer ist und wir im obigen Vorgang keine Daten finden konnten, beispielsweise wenn keiner der Artikel eine Titel- und/oder Autorenangabe besaß. In diesem Fall ist das Ergebnis natürlich entsprechend ebenfalls leer.

Wurden Start- und Endwerte verschiedener Volumes angegeben, so wird an dieser Stelle die Bandnummer inkrementiert und mit dem nächsten Band fortgefahren.

Verzweigung: publKeys zwischen Volumes verschieden/gleich Je nach Verlag kann es sein, dass für ein weiteres Volume ein neuer *publKey* benötigt wird oder eben nicht. Entsprechend beginnt die Verarbeitung eines neuen Bandes in einer der beiden Aktionen.

4.3.2 Beispiel

Nun, da der Ablauf der Extraktion theoretisch geschildert wurde, soll er noch einmal anhand eines konkreten Beispiels nachvollzogen werden. Betrachten wir hierzu Volume 44, Issue 7 der "ACM SIGPLAN Notices", ein Newsletter der ACM, der unter dem URL `http://portal.acm.org/toc.cfm?id=1543136` zu finden ist. Abbildung 4.4 zeigt einen Ausschnitt jener HTML-Seite (links) sowie des zugehörigen Quelltextes (rechts). Die Bedeutung der beiden grau markierten Bereiche wird an entsprechender Stelle erläutert.

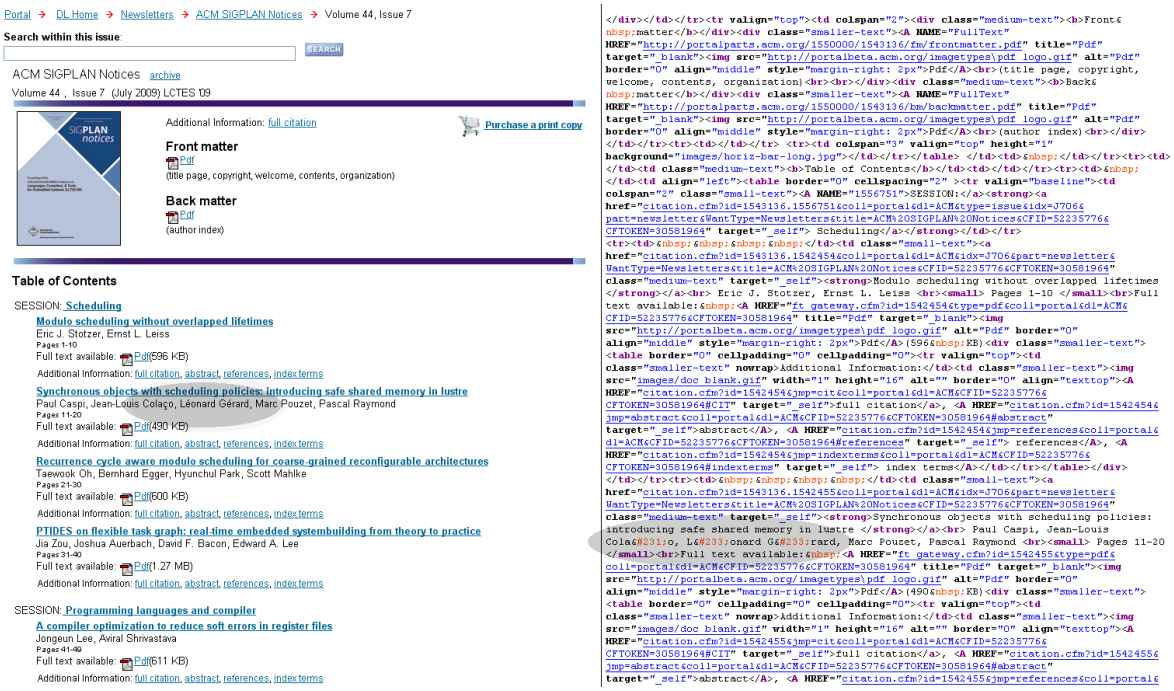


Abb. 4.4: Beispielhafte Seite zur Beschreibung des Wrappers: Links ein Ausschnitt der Ansicht im Browser, rechts des zugehörigen Quelltextes
Quelle: `http://portal.acm.org/toc.cfm?id=1543136&idx=J706`

Zur Erfassung dieser Daten muss die Wrapper-Software mittels des Kommandos

```
> java get "http://portal.acm.org/toc.cfm?id=1543136&idx=J706" 44.7
```

gestartet werden. Da es sich bei dem Newsletter um eine Publikation handelt, deren Daten in einer BHT_j-Datei ausgegeben werden sollen, ist die Angabe von Volume und Issue hier obligatorisch. Eine umfassende Beschreibung des `get`-Kommandos ist in Anhang A.2.1 zu finden.

Im ersten Schritt des Extraktionsprozesses (*ermittle publicKey*) wird nun aus dem übergebenen URL der Publikationsschlüssel des Journals entnommen – in unserem Beispiel lautet er `J706`, die andere Nummer (`1543136`) ist die ID der betrachteten Seite – und validiert, indem die Seite mit dem URL `http://portal.acm.org/toc.cfm?id=J706` geladen wird. Bei dieser handelt es sich um die TOC-Seite der "ACM SIGPLAN Notices", und eben jene benötigt der Wrapper, um seine Arbeit zu verrichten. Von dieser Seite liest er nun sämtliche Issues des 44. Volumes, findet in diesen das von uns gewünschte Issue 7 und speichert dessen URL in der Liste ab (*ermittle zu scannende URLs*).

Hier stellt sich natürlich die Frage, warum der Wrapper nicht gleich den von uns übergebenen URL benutzt – da dieser ja sofort zur gewünschten Seite führt – und den Umweg über die TOC-Seite des Journals wählt. Dies hat den Grund, dass es auf diese Weise keine Rolle spielt, welcher URL konkret übergeben wird. Der Wrapper nutzt lediglich den Publikationsschlüssel und ist danach in der Lage, beliebige Volumes und Issues der Publikation zu lesen; er muss sich lediglich nach den beim Aufruf übergebenen Parametern richten und nicht nach den Parametern des URLs. Diese Tatsache bietet vor allem im Hinblick auf die Automatisierung der Extraktion, welche in Abschnitt 4.4 erörtert wird, einen immensen Vorteil.

Nun beginnt der eigentliche Extraktionsprozess. Zunächst wird der Quelltext des zuvor ermittelten URLs über HTTP abgerufen (*lese TOC*). Die globalen Daten (volume, issue, month, year) befinden sich, wie man Abbildung 4.3.2 entnehmen kann, allesamt im Kopf der Seite und können von dort auf einfachste Weise extrahiert werden (*extrahiere globale Daten*). Da alle Werte in diesem Fall bereits in gewünschtem Format vorliegen, entfällt die Normalisierung (*normalisiere globale Daten*). Wäre jedoch beispielsweise das Publikationsdatum im Format "`Jun. 09`" angegeben, so müsste hier eine entsprechende Transformation des Monatsnamens stattfinden, sowie eine Ergänzung der Jahreszahl zu einem vierstelligen Wert. Diese globalen Daten werden abgespeichert und während des gesamten weiteren Extraktionsvorgangs sämtlichen Ergebnisrecords hinzugefügt. Neben jenen Angaben finden wir auch noch die Information "LCTES '09", ein Hinweis darauf, dass die Artikel dieses Issues in Zusammenhang mit jener Konferenz[4] stehen. Auch diese Information speichern wir global und werden sie den einzelnen Records als Zwischenüberschrift (*section*) übergeben.

Anschließend beginnt der Wrapper mit der Extraktion der einzelnen Artikelinformationen (*extrahiere spezielle Daten*). Zunächst wird eine weitere Zwischenüberschrift gefunden: "Scheduling". Diese ist jedoch im Gegensatz zu obiger Konferenzinformation ("LCTES '09") nicht global, sondern nur für eine bestimmte Anzahl an Artikeln gültig. Da der *section*-Wert bereits vergeben ist, werden wir diese Information als *subsection* übergeben – solange, bis wir eine andere Information erhalten, was nach dem vierten Artikel der Fall sein wird ("Programming languages and compiler"). Man erkennt hier deutlich die zuvor genannte 'Zwischenstellung' zwischen globalen und speziellen Daten, welche die Zwischenüberschriften einnehmen.

[4] `http://www.cse.psu.edu/lctes09/`

Nun werden die Daten des ersten Artikels erfasst: Der Titel lautet "Modulo scheduling without overlapped lifetimes", dessen Autoren sind "Eric J. Stotzer" und "Ernst L. Leiss", und er befindet sich auf den Seiten "1-10". Ein DOI ist nicht aufzufinden, doch wie in Kapitel 3.2.1 erläutert lässt sich dieser aus dem URL der Abstract-Seite, welchen wir im Quellcode finden, konstruieren.[5] Diese speziellen Daten werden nun zu einem Record gemäß unserem Datenmodell (vgl. Kapitel 2.4) vereint und bei diesem Vorgang ebenfalls normalisiert (*normalisiere spezielle Daten*), was jedoch in diesem Falle wiederum nicht nötig ist, da alle Daten in gewünschter Form vorliegen. Abschließend wird dieses Record dem Ergebnis – einer bis zu diesem Zeitpunkt noch leeren Record-Liste – hinzugefügt (*füge Record zu Ergebnis hinzu*).

Da die Seite noch nicht fertig bearbeitet ist und wir die Frage "*weitere spezielle Daten?*" mit "ja" beantworten können, fährt der Wrapper mit der Extraktion der nächsten speziellen Daten fort. Titel, Seitennummern und DOI verursachen auch hier keine Probleme, doch die Autorennamen enthalten Sonderzeichen, welche im Quelltext mittels numerischer Entities codiert sind (`&#231;` und `&#233;`). Diese müssen bei der Normalisierung in DTD-konforme Zeichensequenzen umgewandelt werden. Hierzu werden sie zunächst intern nach UTF-8 transformiert (ç und é) und anschließend, bevor sie dem Ergebnis hinzugefügt werden, gemäß der DTD in benannte Entities verwandelt (`&ccedil;` und `&eacute;`).

So fährt der Wrapper nun fort, bis alle Artikel der Seite erfasst wurden. Dann verzweigt er zur Frage "*weitere TOC-Seiten?*", welche er mit "nein" beantworten kann, da wir zu Anfang nur den URL dieser einen Seite erfasst hatten. Hätte der entsprechende Parameter bei Aufruf des `get`-Kommandos anders gelautet (beispielsweise `44` statt `44.7`, was die Extraktion des gesamten Volumes veranlasst hätte), so würde nun der Quellcode der nächsten Seite eingelesen und mit dieser analog zur obigen Seite verfahren.

Wir gelangen jedoch nun zur Station "*bereite Ergebnis auf*". Hier werden nicht wie zuvor die Daten der einzelnen Records bearbeitet, sondern die gesamte Record-Liste. Die *pages*-Attribute werden auf Linearität untersucht, und im Falle dass sie zwar linear, aber unsortiert sind, wird eine Sortierung der Artikel vorgenommen. Anschließend ist der Extraktionsprozess im Engeren Sinne abgeschlossen, und wir müssen lediglich das gesamte Ergebnis in eine Datei ausgeben (*gib Ergebnis in Datei aus*). Hierzu werden sämtliche Records der Liste der Reihe nach ins BHT_j-Format transformiert.

Da kein weiteres Volume bearbeitet werden soll (*bei journals: weiteres volume?*) ist der Extraktionsprozess an dieser Stelle beendet. Andernfalls würde der Wrapper, da bei ACM die Publikationsschlüssel innerhalb eines Journals gleich bleiben (*publKeys verschiedener Volumes gleich*), wieder zur TOC-Seite des Journals zurückkehren und die URLs der Issues des nächsten zu erfassenden Volumes auslesen (*ermittle zu scannende URLs*).

Ergebnis dieses Extraktionsvorgangs ist die Datei `acm44-7.bht`, welche in Abbildung 4.5 zu sehen ist.

[5] Das *href*-Attribut des den Titel umschließenden `<a>`-Tags beginnt mit "`citation.cfm?id=1543136.1542454&coll=GUIDE&dl=`", woraus sich der DOI `http://doi.acm.org/10.1145/1543136.1542454` ergibt

```
<h2>Volume 44, Number 7, July 2009</h2>
LCTES '09
Scheduling
<ul>
<li>Eric J. Stotzer, Ernst L. Leiss:
Modulo scheduling without overlapped lifetimes.
1-10
<ee>http://doi.acm.org/10.1145/1543136.1542454</ee>
<li>Paul Caspi, Jean-Louis Cola&ccedil;o, L&eacute;onard G&eacute;rard, Marc Pouzet, Pascal Raymond:
Synchronous objects with scheduling policies: introducing safe shared memory in lustre.
11-20
<ee>http://doi.acm.org/10.1145/1543136.1542455</ee>
<li>Taewook Oh, Bernhard Egger, Hyunchul Park, Scott Mahlke:
Recurrence cycle aware modulo scheduling for coarse-grained reconfigurable architectures.
21-30
<ee>http://doi.acm.org/10.1145/1543136.1542456</ee>
<li>Jia Zou, Joshua Auerbach, David F. Bacon, Edward A. Lee:
PTIDES on flexible task graph: real-time embedded systembuilding from theory to practice.
31-40
<ee>http://doi.acm.org/10.1145/1543136.1542457</ee>
</ul>
Programming languages and compiler
<ul>
<li>Jongeun Lee, Aviral Shrivastava:
A compiler optimization to reduce soft errors in register files.
41-49
<ee>http://doi.acm.org/10.1145/1543136.1542459</ee>
<li>Hugh Leather, Michael O'Boyle, Bruce Worton:
Raced profiles: efficient selection of competing compiler optimizations.
50-59
<ee>http://doi.acm.org/10.1145/1543136.1542460</ee>
<li>Xuejun Yang, Nathan Cooprider, John Regehr:
Eliminating the call stack to save RAM.
60-69
<ee>http://doi.acm.org/10.1145/1543136.1542461</ee>
  ⋮
```

Abb. 4.5: Ergebnis des Beispiels: Die Datei `acm44-7.bht` enthält alle gewünschten bibliographischen Daten in BHT_j-Format
Quelle: eigene Erstellung

4.4 Ausblick: Automatisierung der Extraktionsvorgänge

Zum Abschluss der Überlegungen bzgl. der Extraktion soll ein Ausblick auf weitere mögliche Erweiterungen der vorhandenen Software gegeben werden. Zum einen wäre natürlich eine Überarbeitung der Wrapper möglich, um von der manuellen Codierung Abstand zu nehmen und lernende Algorithmen zu implementieren. Es ist jedoch fraglich, ob der Nutzen einer solchen Umstrukturierung tatsächlich die damit verbundene Arbeit aufwiegen würde.

Eine andere, in jedem Fall lohnenswerte Ergänzung zur bestehenden Software wäre jedoch eine Automatisierung der Extraktionsvorgänge. Gerade bei Zeitschriften ist die immer wieder erforderliche manuelle Überprüfung, ob neue Hefte erschienen sind, auf Dauer äußerst mühsam. Zudem kann es natürlich leicht geschehen, dass eine solche Prüfung vergessen wird und entsprechende Daten somit über eine längere Zeitspanne hin in DBLP fehlen, obwohl sie auf dem Server des entsprechenden Verlags bzw. der DL verfügbar sind. Hier bietet es sich in höchstem Maße an, eine Software zu konstruieren, die in festen periodischen Abständen eigenständig nach neuen Veröffentlichungen sucht und im Erfolgsfall die entsprechende Extraktion vornimmt.

Eine solche Software wurde im Rahmen dieser Arbeit nicht implementiert. Im Folgenden sollen jedoch sämtliche theoretischen Überlegungen beschrieben werden, die in Bezug auf eine solche Software angestellt wurden.

Die Software sollte lediglich durch ihren Namen aufgerufen werden und keine Parameter erfordern. Dies ermöglicht die Einrichtung eines Cronjobs[6], so dass die Software in regelmäßigen Abständen automatisch gestartet wird. Sämtliche Konfigurationsparameter sollten in geeigneter Weise, beispielsweise in einer XML-Datei, ausgelagert sein.

Zur Verrichtung der Aufgabe sind die folgenden Parameter notwendig:[7]

`publisher` und `publKey`: Eindeutige Identifizierung der Extraktionsquelle und des zu bearbeitenden Journals. Da die Wrapper-Software ohnehin eine solche Identifizierung anhand des Domainparts des übergebenen URLs vornimmt und der *publKey* wie zuvor beschrieben aus dessen Localpart gewonnen wird, bietet es sich an, an dieser Stelle einfach einen beliebigen URL zu übergeben, der auf Daten des gewünschten Journals verweist.

`frequency`: Je nach Erscheinungszyklus der Hefte kann es sinnvoll sein, wöchentlich, monatlich oder halbjährlich nach neuen Publikationen Ausschau zu halten. In keinem Fall sollten alle Seiten bei jedem Aufruf des entsprechenden Cronjobs (hier wäre eine tägliche Ausführung empfehlenswert) durchsucht werden – dies würde lediglich unnötigen Traffic auf eigenen und fremden Servern verursachen. Daher sollte ein entsprechender Wert konfiguriert werden können, damit die Software im 'richtigen' zeitlichen Rhythmus nach Neuerscheinungen sucht. Als Werte wären fest definierte Konstanten denkbar (`weekly`, `biweekly`, `monthly`, ···), aber auch die Eingabe eines Integerwertes, der die Tage bis zum nächsten Seitenbesuch angibt. Bei Fehlen dieser Angabe sollte ein geeigneter Defaultwert (beispielsweise 'monatlich') angenommen werden.

`last_visited`: Hier sollte ein standardisierter Datumswert, beispielsweise nach 'ISO 8601'-Norm[8], eingetragen sein, der angibt, wann die Quelle zuletzt nach neuen Daten durchsucht wurde.

Mit Hilfe der beiden zuletzt genannten Parameter ist die Software in der Lage zu entscheiden, ob die Seite besucht werden soll oder nicht: Sie soll nämlich genau dann besucht werden, wenn das Datum des letzten Besuchs (`last_visited`) zuzüglich der Besuchsfrequenz (`frequency`) kleiner ist als das aktuelle Datum. Nach dem Besuch muss die Software dann den Wert des Parameters `last_visited` entsprechend auf das aktuelle Datum setzen.

Damit die Software feststellen kann, ob tatsächlich neue Daten verfügbar sind, werden weitere Parameter benötigt. Dabei ist die Entscheidung, wie die entsprechende Information codiert werden soll, weniger trivial als sie anfänglich erscheinen mag. Wurde beispielsweise zuletzt Heft 7 des dritten Bandes gefunden, so liegt es zunächst nahe, diesen Wert (z.B. in der auch von der Wrapper-Software geforderten Form '`3.7`') als einzigen Parameter abzuspeichern. Die Software könnte dann entsprechend nach 'Band 3, Heft 8' sowie 'Band 4, Heft 1' suchen – da wir nicht wissen, ob Heft 7 das letzte des Bandes ist – und anschließend den Wert entsprechend auf den des zuletzt gefundenen Heftes setzen. Dies mag bei vielen Publikationen Erfolg versprechen, jedoch nicht in jedem Fall.

[6] `http://de.wikipedia.org/wiki/Cron`
[7] Die Namen der Parameter sind selbstverständlich nicht verpflichtend, erscheinen jedoch geeignet.
[8] `http://www.iso.org/iso/en/prods-services/popstds/datesandtime.html`

Die Springer LNCS (vgl. Kapitel 3.2.10) werden nicht immer in der Reihenfolge ihrer Nummerierung veröffentlicht. Am 12. September 2009 trugen die vier aktuellsten Bände die Nummern 5813, 5806, 5797 und 5793 (siehe Abb. 4.6). Es ist zu erwarten, dass die fehlenden Nummern

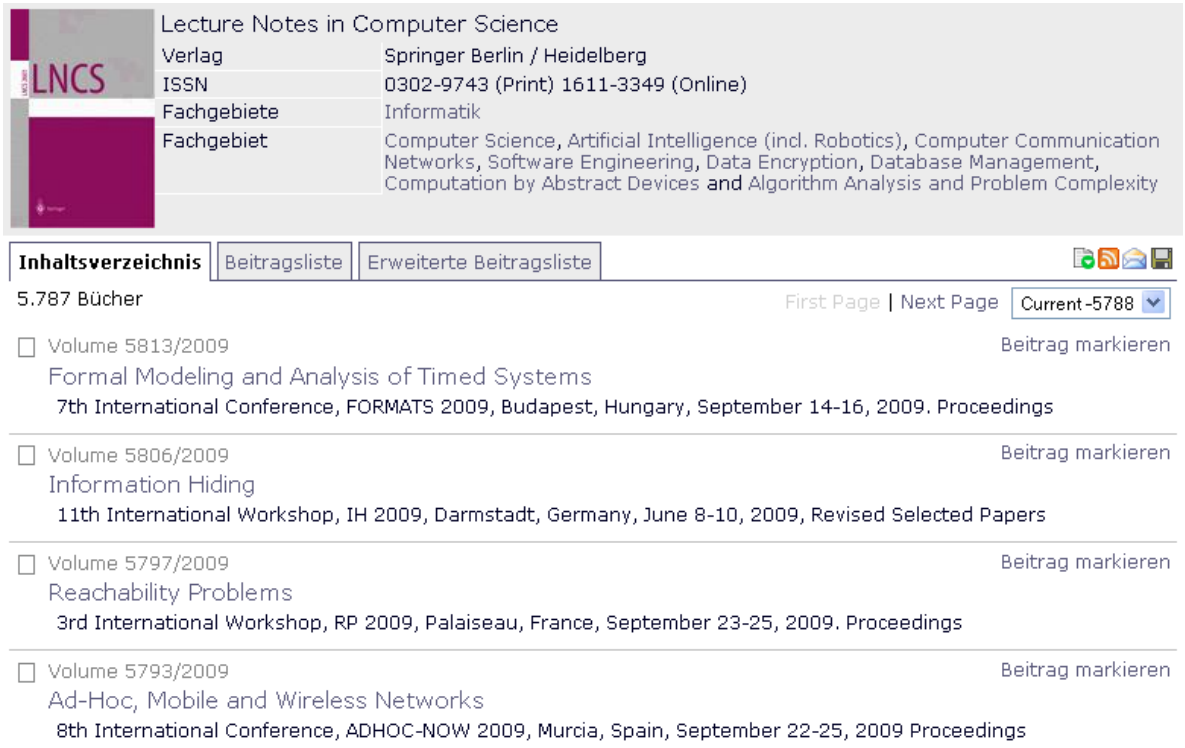


Abb. 4.6: Aktuelle Bände der Springer LNCS: Die Veröffentlichung erfolgt oftmals nicht in der Reihenfolge der Nummerierung
Quelle: `http://www.springerlink.com/content/105633/`,
Stand: 12. September 2009

im Laufe der Zeit ergänzt werden. Abgesehen davon, dass obige Strategie ("Suche nach dem nächsten Band bzw. Heft") an dieser Stelle versagt – was leicht durch die Regel "Suche nach neueren Bänden bzw. Heften als..." behoben werden könnte – wäre es in eben diesem Fall äußerst wünschenswert, dass die Software bei jedem Besuch der Seite nicht bloß nach neueren Nummern suchen, sondern auch vormals fehlende Bände finden würde.

Eine andere Strategie könnte darin bestehen, alle bereits erfassten Bände/Hefte abzuspeichern. Dies würde jedoch gerade im o.g. Fall der LNCS mit mehreren tausend Publikationen eine entsprechend große Liste bewirken, die ebenso ineffizient wie unnötig wäre. Sinnvoll dagegen erscheint die Speicherung jener Bände, die bei einer vorherigen Extraktion nicht gefunden werden konnten. Im Beispiel der LNCS würde diese daher am 12. September 2009 die Nummern 5812-5807, 5805-5798, 5796-5794,··· in absteigender Reihenfolge beinhalten.

Dennoch ergeben sich auch hier bei konsequenter Anwendung dieser Strategie Probleme. Nehmen wir an, das zuletzt gefundene Heft einer Zeitschrift sei 'Volume 5, Issue 8'. Finden wir nun ein Heft mit 'Volume 6, Issue 1', so kann nicht mit Sicherheit entschieden werden, ob 'Volume 5,

Issue 9', 'Volume 5, Issue 10' etc. als Lücke aufgefasst oder ignoriert werden sollen. Hier ist also ebenfalls eine durchdachte Strategie notwendig (beispielsweise in Form der Speicherung des jeweils letzten erfassten Issues eines Volumes), evtl. auch ein manuell festzulegender bool'scher Wert, der bestimmt, ob die jeweilige Publikation Lücken aufweisen kann oder nicht.

Es wird empfohlen, obige Konfigurationsparamter in Form einer XML-Datei anzulegen. Dabei ist jedoch zu beachten, dass die automatisch ablaufende Software an dieser Datei Änderungen vornehmen muss, um das Datum des letzten Besuchs und die Informationen der zuletzt gefundenen Publikation zu aktualisieren sowie entsprechende Lücken zu vermerken oder eben jene Vermerke zu beseitigen. Zudem muss es jederzeit möglich sein, die Datei manuell zu bearbeiten, vor allem, um neue Zeitschriften zur Liste hinzuzufügen. Hier ist äußerste Vorsicht geboten, da es bei gleichzeitigem Zugriff von Software und Benutzer zu Inkonsistenzen und Datenverlusten kommen kann.

Eine Abhilfe könnte die Verwendung *zweier* XML-Dateien an Stelle von nur einer schaffen. Die erste Datei sollte dann lediglich die statischen Informationen `publisher`/`publKey` und `frequency` enthalten und könnte von Benutzern zu beliebigen Zeitpunkten manuell bearbeitet werden. Die zweite Datei dürfte dagegen lediglich von der Software aktualisiert werden und würde die übrigen, dynamischen Parameter enthalten. Problematisch ist jedoch, dass der Benutzer auch in der Lage sein sollte, einen 'Startwert' einzugeben, da es offensichtlich völlig unsinnig wäre, die gesamte LNCS-Serie beim ersten Aufruf des Programms in Ermangelung entsprechender Parameter komplett einzulesen. Würde jener Startwert in der ersten XML-Datei gespeichert, so würde diese nach kurzer Zeit veraltete Werte enthalten, die zumindest für Verwirrung sorgen könnten.

Eine andere, weniger elegante aber evtl. Erfolg versprechende Lösung wäre es, den Cronjob ausschließlich nachts oder am Wochenende ausführen zu lassen, um eine Überschneidung mit manuellen Änderungen zu vermeiden.

Sind all diese Überlegungen bzgl. der Modellierung geeigneter Parameter zufriedenstellend abgeschlossen, so ist die Programmierung der entsprechenden Software trivial. Diese muss lediglich obige Liste der zu überprüfenden Publikationen abarbeiten. Falls eine der Seiten besucht werden soll (was wie oben beschrieben abhängig von `frequency` und `last_visited` ist), müssen Veröffentlichungen, deren Volume-Issue-Werte über der zuletzt erfassten liegen bzw. als 'Lücken' vermerkt sind, gefunden werden. Hierzu kann die Wrapper-Software mittels der entsprechenden Parameter (der zur Identifikation der Publikation bekannte URL sowie die zu erfassenden Volumes/Issues) aufgerufen werden. Liefert sie kein bzw. ein leeres Ergebnis, so wurden keine entsprechenden Publikationen gefunden. Ansonsten sollte das Ergebnis in ein festes Verzeichnis kopiert und evtl. eine Benachrichtigung per E-Mail abgesandt werden. Abschließend müssen die dynamischen Daten (in jedem Fall das Datum des letzten Besuchs, im Erfolgsfall auch die zuletzt erfassten Publikationen) aktualisiert werden.

Natürlich sollten auch die auf diese Weise erzeugten Dateien vor der Aufnahme in DBLP manuell überprüft werden. Die beschriebene Software garantiert bei sinnvoller Konfiguration (insbesondere der Besuchsfrequenz der Seiten) eine automatische, zeitnahe und lückenlose Erfassung von Zeitschriften und Buchserien.

Kapitel 5

Informationsfusion

Nachdem wir uns bislang mit der Informationsextraktion beschäftigt haben, wenden wir uns nun dem zweiten Teil dieser Arbeit zu, der Fusion bibliographischer Informationen. Hierzu werden wir zunächst einen kurzen Blick auf verschiedenste Anwendungsgebiete werfen (Abschnitt 5.1), und anschließend die eng verwandten Begriffe der *Datenfusion* bzw. *Datenintegration* erläutern (Abschnitt 5.2). Hiernach wenden wir uns in Abschnitt 5.3 einem geeigneten Modell zu, mit dessen Unterstützung wir in der Lage sein werden, in den Kapiteln 7 und 8 eigene Theorien und Algorithmen bzgl. der Fusion der bibliographischen Daten unseres aus Kapitel 2.4 bekannten Datenmodells zu entwickeln.

Der Begriff "Informationsfusion" wird in der Literatur zumeist im Kontext der Verknüpfung von Messdaten verschiedener Sensoren verwandt. Ruser und León beispielsweise bezeichnen Informationsfusion als einen *Prozess*, bei welchem "Daten aus unterschiedlichen Sensoren oder Informationsquellen mit dem Ziel [verknüpft werden], neues oder präziseres Wissen über physikalische Größen, Ereignisse und Situationen zu gewinnen" ([RL07]).

Im weiteren Sinne kann man Informationsfusion als einen allgemeinen Prozess auffassen, bei welchem gleich- oder verschiedenartige Informationen zu einem Gesamtbild zusammengefügt werden. Wolfgang Koch, Leiter der Abteilung 'Sensordaten- und Informationsfusion' im 'Forschungsinstitut für Kommunikation, Informationsverarbeitung und Ergonomie' (FKIE) stellt heraus, dass prinzipiell alle Lebewesen Informationsfusion betreiben, indem sie "Eindrücke unterschiedlicher Sinnesorgane mit zuvor erlerntem Erfahrungswissen und Mitteilungen anderer Lebewesen" fusionieren und sich auf diese Weise ein 'Bild' ihrer Umwelt verschaffen ([VDE08]). Weiterhin bezeichnet er die Informationsfusion als "interdisziplinäres Fachgebiet", d.h. sie ist in der Lage, unterschiedlichste Daten miteinander in Verbindung zu bringen.

5.1 Anwendungsgebiete der Informationsfusion

Die Anwendungsgebiete der Informationsfusion sind daher vielfältig. Die folgende Liste stellt lediglich eine kleine Auswahl unterschiedlicher Anwendungsgebiete dar, die einen Eindruck jener Vielfalt vermitteln sollen.

Wettervorhersage Systeme zur Wettervorhersage beruhen auf Hinweisen verschiedener Informationsquellen wie Satelliten, Wetterstationen, Radargeräten etc., die in geeigneter Weise fusioniert werden müssen ([FZ09]).

Roboter-Steuerung Intelligente Roboter-Steuerung bedarf der Fusion verschiedenartiger Informationen, die über Sensoren der Roboter eingefangen werden oder in einer Datenbank gespeichert sind. Die so genannte "Multi-sensor information fusion technology" beschäftigt sich mit der Verarbeitung jener Daten in Echtzeit und nutzt Algorithmen und Techniken aus Mathematik, Ingenieurwissenschaften und Bionik.[1] Einen guten Überblick über derartige Systeme bietet beispielsweise [ZLH08].

Biometrie Im Bereich der Biometrie werden Informationen verschiedener Quellen zur Personenerkennung eingesetzt. Hier werden Merkmale wie beispielsweise Aussehen, Sprache oder Handschrift zur eindeutigen Identifizierung einer Person eingesetzt, indem die entsprechenden Daten geeignet fusioniert werden. Eine Übersicht über verschiedene 'Level' dieser Art der Fusion bietet [FZ09]. Auch der ePA (elektronischer Personalausweis), welcher nach einem Beschluss des Deutschen Bundestags ab 1. November 2010 den bisherigen Personalausweis ablösen soll, werden biometrische Daten (Foto, Unterschrift, zwei Fingerabdrücke auf freiwilliger Basis) gespeichert, mittels deren Fusion eine eindeutige Identifikation des Benutzers ermöglicht wird[2].

Führungsinformationssysteme Derartige Systeme kommen vor allem im militärischen Bereich zum Einsatz und sollen dort behilflich sein, schnelle strategische Entscheidungen zu treffen. Hier dienen Methoden der Informationsfusion zur "situationsadaptiven Entscheidungsunterstützung mit allen damit verbundenen Konsequenzen" ([WG09]).

Jede dieser Anwendungsgebiete definiert eigene Verfahren zur Fusion der Daten, stets abhängig von der Domäne, der die betrachteten Daten entstammen. Entsprechende Literatur zu Fusionsverfahren bzgl. der in der vorliegenden Arbeit untersuchten Domäne der bibliographischen Daten konnte jedoch nicht gefunden werden. Im Bereich der Datenbanken kommen aber sehr ähnliche Konzepte zur Anwendung. Hier spricht man meist von Datenfusion.

5.2 Datenfusion / Datenintegration

Die Begriffe der *Datenfusion* und *Datenintegration* sind eng mit dem der Informationsfusion verwandt, wobei hier in der Literatur keine einheitliche Verwendung zu finden ist. Conrad et al. beschreiben die Datenintegration als einen von fünf Funktionsbereichen eines Softwaresystems, der eine Basis für eine Informationsfusion bildet ([CSS99]). Dagegen stellt die Datenfusion

[1] `http://de.wikipedia.org/wiki/Bionik`

[2] Gesetzesbeschluss des Deutschen Bundestages: Gesetz über Personalausweise und den elektronischen Identitätsnachweis sowie zur Änderung weiterer Vorschriften, Drucksache 32/09 vom 23. Januar 2009, `http://dip21.bundestag.de/dip21/brd/2009/0032-09.pdf`

im Datenintegrationsmodell von Bleiholder und Naumann eine Phase des Datenintegrationsprozesses dar (vgl. Abschnitt 5.3). Den "Prozess des Zusammenführens [von] Daten aus einzelnen Quellen unter Auflösung von auftretenden Konflikten" bezeichnet Bleiholder dagegen als "Data Merging" ([Ble04]) und beschreibt damit ebenfalls einen Fusionsprozess. Im Kontext des WWW, in welchem sich, bedingt durch dessen wachsende Popularität und Bedeutung, die dringende Notwendigkeit zeigt, Systeme zur Integration und Fusion zu konstruieren, benutzen Yao et al. die Begriffe "Informationsfusion", "Informationsintegration" und "Datenintegration" völlig synonym: "In the context of the Web, we do not differentiate between the terms Web information fusion, Web information integration and Web data integration, and use them interchangeably to refer to the task of combining information on the Web" ([YRW08]).

Da die von uns betrachteten bibliographischen Daten dem Web entnommen sind, und wir in den Kapiteln 9 und 10 weitere Web-Ressourcen zur Fusion mit bestehenden Daten nutzen werden, betrachten wir dementsprechend in den meisten Fällen eine derartige Web-Informationsfusion. Wir werden daher ebenfalls auf eine exakte Unterscheidung jener Begriffe verzichten und immer dann von einer Fusion sprechen, wenn mehrere Daten zu einem einzigen Ergebnis zusammen gefasst werden.

Prinzipiell unterscheidet man hier zwischen der Fusion heterogener und homogener Informationen. Bei der *heterogenen* Datenfusion werden unterschiedliche Informationen zu einem Gesamtbild zusammengefasst. Bei 'Google Maps'[3] beispielsweise erhält der Benutzer durch Eingabe textueller Daten (i.d.R. eine Postanschrift) Informationen über die topologische Beschaffenheit des Geländes, Satellitenaufnahmen des Gebietes, Fotos etc. ([YRW08]). Auch bei den zuvor in Abschnitt 5.1 genannten Anwendungen handelt es sich durchweg um eine heterogene Fusion.

Von *homogener* Informationsfusion spricht man dagegen, wenn die Daten in einer einheitlicheren Form – beispielsweise in Form von Textdateien oder Sensoren mit einheitlichen Sensorprinzipien – vorliegen und mehrere Datensätze zu einem Gesamtbild zusammengefasst werden sollen. Im Falle unserer bibliographischen Daten, die allesamt in Textform vorliegen, handelt es sich daher stets um eine homogene Fusion. Gerade im Gebiet der Datenbanksysteme beschäftigt sich die aktuelle Forschung laut Bleiholder und Naumann überwiegend mit Systemen zur Integration heterogener Daten, während vor allem die kommerziellen Systeme vor der Integration homogener, vorhandener Daten "zurückschrecken" ([BN08]).

5.3 Das Datenintegrationsmodell nach Bleiholder und Naumann

In [BN08] wird Datenfusion im engeren Sinne lediglich als der letzte Schritt eines komplexeren Vorgangs, der 'Datenintegration' (data integration), bezeichnet. Abbildung 5.1 zeigt die drei Phasen des Datenintegrationsprozesses nach Bleiholder und Naumann, welche als Grund-

[3] `http://maps.google.com`

lage für die vorliegende Arbeit genommen wurde. Jener Prozess untergliedert sich demnach in die drei Phasen ‘Schema Mapping’, ‘Duplicate Detection’ und ‘Data Fusion’. [BN08].

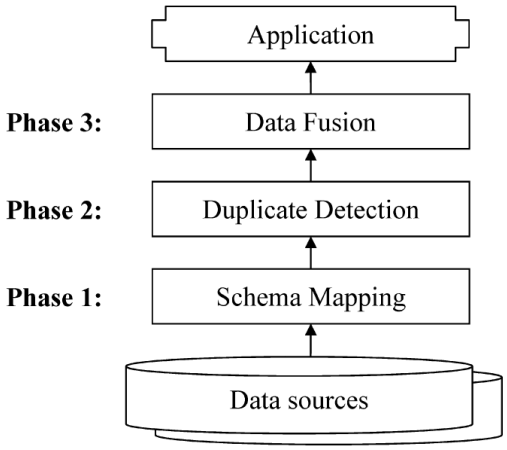


Abb. 5.1: Die drei Phasen der Datenintegrationsprozesses nach Bleiholder und Naumann
Quelle: [BN08], Seite 2

In den Kapiteln 6 bis 8 werden wir uns eingehend mit dem Spezialfall der Fusion bibliographischer Daten beschäftigen. Hier wird uns jenes Modell von Naumann und Bleiholder helfen, die einzelnen Phasen der Fusion bzw. Integration zu bearbeiten und – mit einem Blick auf die Implementierung – theoretisch zu beschreiben. Die folgenden Erläuterungen jener drei Phasen entstammen [NBBW06] und [BN08].

5.3.1 Phase 1: Schema Mapping

Diese erste Phase des Datenintegrationsprozesses ist in wiederum zwei Schritte unterteilt: Die Festlegung eines einheitlichen Schemas und die anschließende Transformation der Daten in eben jenes Schema.

Geht man davon aus, dass die zu fusionierenden Daten beliebigen und unterschiedlichen Quellen entstammen, so muss ebenso davon ausgegangen werden, dass deren Schemata unterschiedlich sind. Da das Fusionsergebnis jedoch in einem einheitlichen Datenschema vorliegen soll, ist es notwendig, sämtliche Daten in ein solches zu überführen; ein Vorgang, der als “Data Transformation” bezeichnet wird. Zuvor muss jedoch ein solches einheitliches Schema gefunden werden, was prinzipiell auf zwei Arten geschehen kann:

Schema integration
Bei dieser Vorgehensweise wird versucht, aus allen im Datensatz bestehenden Schemata ein neues Datenschema zu konstruieren, das komplett und korrekt in Bezug auf die Ausgangsschemata ist, zudem minimal und verständlich.

Schema mapping
Das 'Schema mapping' dagegen setzt ein vorgegebenes Schema voraus und transformiert sämtliche Datensätze in eben jenes Schema. Dieser Ansatz wird vor allem dann verwendet, wenn ein festes Datenformat existiert, in welchem die Ergebnisse vorliegen sollen. In unserem Falle liegt ein derartiges Schema vor: das Datenmodell aus Kapitel 2.4, welches wir bereits zur Informationsextraktion verwendet haben. Wir werden daher stets 'schema mapping' betreiben und die Daten aller Quellen in das bestehende Schema überführen.

Hat man sich auf ein Schema geeinigt, so müssen im zweiten Schritt sämtliche Daten in eben jenes Schema transformiert werden. Hierzu ist eine Normalisierung, wie wir sie im Zusammenhang mit den bibliographischen Daten bereits bei der Extraktion aus Quellen des WWW kennen gelernt haben, notwendig.

5.3.2 Phase 2: Duplicate Detection

Wurden alle Datensätze in ein einheitliches Schema überführt, so kann nun nach Duplikaten gesucht werden, d.h. nach mehreren Repräsentationen des gleichen Objektes der realen Welt ("multiple representations of the same real-world object", [BN08]). Prinzipiell müssen hier sämtliche Datenobjekte mittels einer geeigneten Ähnlichkeitsfunktion paarweise miteinander verglichen werden, doch dabei ergeben sich in der Praxis erhebliche Probleme bzgl. der Effektivität und Effizienz.

Effektivität
Der Vorgang der Duplikatserkennung wird als *effektiv* bezeichnet, wenn möglichst wenige bzw. keine 'false positives' im Ergebnis auftauchen, und keine 'true negatives' zurückbleiben.[4] Dieser Wert ist im Wesentlichen von der Güte der verwendeten Ähnlichkeitsfunktion sowie einem zu definierenden Schwellenwert, der angibt, ab welchem Ergebnis der Ähnlichkeitsfunktion zwei Datensätze noch als 'ähnlich' angesehen werden sollen, abhängig.

Effizienz
Die *Effizienz* hängt dagegen von der Speicherplatz- und Laufzeitkomplexität des Algorithmus' zur Duplikaterkennung ab. Ein paarweiser Vergleich aller Datensätze kann gerade bei umfangreichen Datenbeständen äußerst große Mengen an entsprechenden Ressourcen verbrauchen, zumal die Ähnlichkeitsfunktion oftmals äußerst teuer in Bezug auf obige Ressourcen sein kann.

Man unterscheidet zudem die Art der Ähnlichkeitsfunktion bzgl. ihrer Domainabhängigkeit. Bei einer *domainabhängigen* Funktion fließt stets ein gewisser Grad an Semantik mit in die Ähnlichkeitsfunktion ein; sie hängt also von der jeweiligen Domäne, welcher die Daten entstammen, ab. In unserem Fall bildet jedes zu fusionierende Attribut prinzipiell eine eigene Domäne, denn

[4] Als 'false positive' gelten hier solche Daten, die in der realen Welt verschiedene Objekte bezeichnen, jedoch als Duplikate identifiziert werden. Entsprechend sind mit 'true negatives' derartige Daten gemeint, die tatsächlich ein und dasselbe Objekt der realen Welt beschreiben, jedoch nicht als Duplikate erkannt werden.

bei der Fusion zweier Autorennamen muss gänzlich anders vorgegangen werden als bei der Fusion zweier Seiteninformationen, obwohl es sich in beiden Fällen syntaktisch gesehen um Strings handelt. Als *domainunabhängig* bezeichnet man dagegen solche Funktionen, die rein syntaktisch auf die Daten angewandt werden können. So werden sämtliche Stringmatching-Algorithmen, wie beispielsweise die Levenshtein-Distanz [Lev66], als domainunabhängig angesehen.

5.3.3 Phase 3: Data Fusion

In dieser letzten Phase werden nun die zuvor identifizierten Duplikate miteinander fusioniert. Es werden somit n Datensätze, die den gleichen realen Sachverhalt zu beschreiben scheinen, zu einem einzigen Datensatz zusammengefasst. Enthalten hierbei einzelne Daten unterschiedliche Werte, so spricht man von einem *Konflikt*. Wollen wir beispielsweise zwei Records unseres Datenmodells miteinander fusionieren, die über unterschiedliche Werte in einem ihrer Attribute (z.B. unterschiedliche EE-Informationen) verfügen, so stellt dies einen eben solchen Konflikt dar.

Prinzipiell lassen sich zwei Arten von Konflikten unterscheiden: 'Unsicherheiten' und 'Widersprüche'. Eine Unsicherheit liegt dann vor, wenn ein Datensatz einen Wert enthält, der andere jedoch nicht. Liegen jedoch zwei verschiedene Informationen ungleich des Nullwertes vor, so handelt es sich um einen Widerspruch ([Ble04]).

Bleiholder und Naumann liefern eine Zusammenstellung diverser Konfliktlösungsstrategien, von welchen im Folgenden eine Auswahl vorgestellt werden soll. Eine ausführlichere Übersicht jener Strategien liefert [BN06].

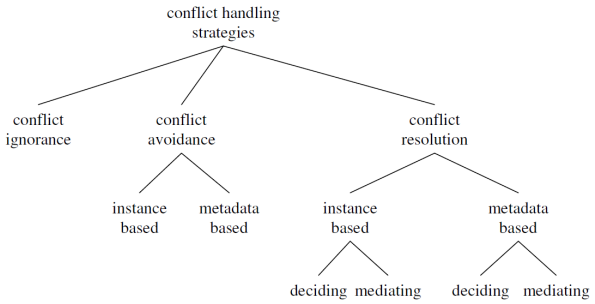


Abb. 5.2: Klassifikation der Strategien zur Konfliktbehandlung nach Bleiholder und Naumann
Quelle: [BN06], Seite 2

Wie Abbildung 5.2 zeigt, lassen sich die Strategien zur Konfliktbehandlung grundsätzlich in drei Kategorien einteilen: conflict ignorance, conflict avoidance und conflict resolution.

Conflict ignorance
Bei Strategien dieser Kategorie werden Konflikte schlichtweg ignoriert; die Software verfügt über keinerlei Algorithmen zur Lösung auftretender Konflikte. Derartige Strategien sind natürlich

leicht zu implementieren und bedürfen stets einer externen Konfliktlösung. Ein Vertreter dieser Kategorie ist die Strategie PASS IT ON, bei welcher alle auftretenden Konflikte an den Benutzer oder eine andere Software weitergeleitet werden und dort entschieden werden müssen.

Conflict avoidance
Auch die Strategien dieser Kategorie lösen die Konflikte nicht direkt, doch sie eliminieren zumindest inkonsistente Datenbestände. Dies birgt den Vorteil, dass eine Entscheidung i.d.R. schneller getroffen werden kann als bei den Konflikt lösenden Strategien (siehe 'conflict resolution'), die Software jedoch meist anhand vorher festgelegter Regeln entscheidet und somit oftmals gar nicht um eventuell aufgetretene Konflikte weiß. Problematisch ist dagegen, dass die Software wegen der fehlenden Betrachtung der Daten oftmals nicht alle verfügbaren Informationen nutzen kann.

Prinzipiell können die Konflikt vermeidenden Strategien wiederum in zwei Sektionen unterteilt werden: Jene, die ihre Entscheidung anhand von Metadaten fällen (metadata based), und solche, die dies nicht tun (instance based). Eine Strategie der ersten Sektion ist TRUST YOUR FRIENDS. Bei dieser wird im Falle eines Konfliktes anhand der Quelle, welcher die Informationen entstammen, entschieden, welche Information vorzuziehen ist. Diese Strategie wird auch bei der Fusion der bibliographischen Daten häufig zum Einsatz kommen: Immer dann, wenn ein Konflikt zweier Attribute nicht nach einer anderen Strategie aufgelöst werden kann, wird grundsätzlich die Information der ersten Quelle ins Ergebnis übernommen (siehe hierzu die Fusion zweier Records in Kapitel 8.2).

Die Strategie TAKE THE INFORMATION dagegen zählt zur zweiten Sektion und nutzt keine Metadaten zur Entscheidung. Dabei besagt sie lediglich, dass im Falle eines Konfliktes zwischen einem beliebigen Wert und einem Nullwert, der entsprechend keinen Informationsgehalt trägt, stets jener Wert ins Ergebnis einfließt. Konflikte zweier Werte ungleich dem Nullwert können mittels dieser Strategie jedoch nicht aufgelöst werden; sie ist demnach geeignet, um Unsicherheiten aufzulösen, nicht aber Widersprüche. Auch diese Strategie wird in Bezug auf die bibliographischen Daten zur Anwendung kommen, wenn ein Datensatz Attribute enthält, die im anderen Datensatz nicht vorhanden sind – was wir nach der Definition in Kapitel 2.4 mit einem Wert von *null* gleichgesetzt haben. Auf diese Weise können wir beispielsweise Daten, die bereits in DBLP erfasst sind, jedoch nicht über ein *ee*-Attribut verfügen, durch Fusion mit einer zweiten Quelle ein solches Attribut hinzufügen.

Conflict resolution
Strategien dieser Kategorie lösen – im Gegensatz zu allen zuvor beschriebenen – Konflikte tatsächlich automatisch auf. Dabei kann wie zuvor eine Unterscheidung zwischen 'instance based' und 'metadata based' getroffen werden. Zudem lassen sich jene Strategien in die Sektionen 'deciding' (entscheidend) und 'mediating' (vermittelnd) unterteilen. Entscheidende Strategien wählen stets einen der vorhandenen Werte aus, während vermittelnde Strategien u.U. einen neuen Wert, der zuvor in keiner der Datenquellen vorhanden war, aus den zur Verfügung stehenden Daten generieren.

Entscheidende Strategien sind beispielsweise CRY WITH THE WOLVES, bei welcher ein Wert dann ins Ergebnis aufgenommen wird, wenn er in der Mehrzahl der zur fusionierenden Da-

tensätze auftritt, ROLL THE DICE, bei welcher ein zufälliges Ergebnis ausgewählt wird, sowie KEEP UP TO DATE, bei welcher das Ergebnis des zeitlich aktuellsten Datensatzes übernommen wird. Die beiden erstgenannten Strategien gehen hierbei 'instance based' vor, während letzterer eine Metainformation zur Entscheidung verwendet. All diese Strategien sind für unsere Zwecke jedoch prinzipiell ungeeignet. Da wir jeweils nur genau zwei Daten zur Verfügung haben, ist ein mehrheitliches Vorkommen eines Wertes nicht möglich, und wir möchten das Ergebnis mit Sicherheit auch nicht dem Zufall überlassen. Eine Bevorzugung der neusten Information würde im Falle einer Aktualisierung bestehender Daten stets die alten Informationen überschreiben, was jedoch auch nicht in jedem Falle gewollt ist, da die bestehenden Daten u.U. bereits mehrfach manuell korrigiert oder ergänzt wurden und wir diese Informationen unter keinen Umständen einfach überschreiben möchten.

Beispiel einer vermittelnden Strategie ist MEET IN THE MIDDLE, bei welcher ein Ergebnis gewählt wird, das allen Daten möglichst ähnlich ist. Diese Strategie werden wir beispielsweise bei der Fusion zweier Namensteile (vgl. Kapitel 8.3.5) anwenden, wenn beide Namen Informationen enthalten, die im jeweils anderen Namen nicht vorhanden sind.

5.3.4 Abschließende Bemerkungen

Bei der Betrachtung unseres konkreten Problems der Fusion bibliographischer Daten werden wir stets mit dem Sonderfall konfrontiert sein, dass wir *genau zwei* fest definierte Quellen besitzen, deren Daten miteinander fusioniert werden sollen. Dennoch lässt sich das in diesem Abschnitt vorgestellte Integrationsmodell nach BLEIHOLDER UND NAUMANN, im Folgenden kurz als (Daten-)Integrationsmodell bezeichnet, in geeigneter Weise auf diese Problemstellung anwenden.

In der ersten Phase, dem 'Schema Mapping', werden wir die Daten aus unterschiedlichen Quellen (BHT-Dateien, DBLP-Records, aber später auch Webseiten oder PDF-Dokumenten) einlesen und in unser einheitliches, internes Datenformat überführen, welches wir bereits zur Extraktion genutzt haben. Hierzu liefert das folgende Kapitel 6 einen praxisnahen Einstieg.

Phase 2 widmen wir uns in Kapitel 7: Die dort beschriebene Fusion komplexer Objekte entspricht der Duplikatserkennung ('Duplicate Detection') des Integrationsmodells. In unserem speziellen Fall sind jedoch die Quellen, welchen die Duplikate entstammen sollen, im Voraus festgelegt, weshalb wir dort von einer *Partnersuche* sprechen werden (Kapitel 7.2). Wir suchen also zunächst keine Duplikate im gleichen Datenbestand, sondern ausschließlich zwischen jenen beiden Quellen. Nur wenn bei diesem Vorgang Datensätze übrig bleiben, werden wir überprüfen, ob es sich dabei um Duplikate innerhalb der gleichen Datenquelle handelt – welche wir aus Gründen der Klarheit stets als *Dubletten* bezeichnen werden – und diese ggf. eliminieren (Kapitel 7.3 und 7.4). In Kapitel 7.1 werden je nach der Art der zu fusionierenden Daten entsprechende Ähnlichkeitsfunktionen definiert, die zwar teilweise domainunabhängige Stringmatching-Algorithmen nutzen, insgesamt jedoch stets domainabhängig sind, da sie sich jeweils auf eine ganz spezielle Art bibliographischer Objekte beziehen.

Die letzte Phase, d.h. die Datenfusion im engeren Sinne, werden wir anschließend in Kapitel 8 exakt beschreiben. Auch dieser Prozess ist nach obiger Definition stets domainabhängig; wir müssen also in jedem Fall die Semantik der zu fusionierenden Datensätze berücksichtigen. Dabei werden wir je nach Typ der zu fusionierenden Objekte und abhängig von der Konfiguration durch den Benutzer unterschiedliche Strategien der Konfliktbehandlung, die im vorherigen Abschnitt vorgestellt wurden, nutzen, um das Ergebnis zu generieren.

Kapitel 6

Fusion zweier strukturierter Quellen

Nachdem nun einige allgemeine Grundlagen beschrieben wurden, wollen wir uns wieder unserem speziellen Fall der bibliographischen Daten widmen. Prinzipiell unterscheidet sich die Fusion von der Extraktion vor allem darin, dass nicht mehr nur eine Quelle betrachtet wird (beispielsweise die Website eines Verlegers oder einer DL), sondern stets zwei Quellen, deren Informationen in sinnvoller und geeigneter Weise miteinander 'verschmolzen' werden sollen. Doch was bedeuten die Worte 'sinnvoll' und 'geeignet' in unserem Kontext?

Natürlich ist es theoretisch denkbar, beliebige Daten miteinander zu kombinieren und ein mehr oder weniger sinnvolles Ergebnis daraus zu enthalten. Unser Ziel ist es jedoch, die Datenqualität der bibliographischen Daten in DBLP zu erhöhen und hierzu eine automatische Vorarbeit zu leisten, um den Anteil an manueller Nachbearbeitung zu verringern. Wir werden daher nur solche Fälle diskutieren, in denen es von der *Semantik* der Daten her sinnvoll ist, diese zu fusionieren. Es ergibt keinen Sinn, einen Autorennamen mit einer Seitennummer zu verschmelzen, ebenso wenig wie wir in der Realität darüber nachdenken werden, einen Elefanten mit einer Ananas zu kreuzen. Aber auch der Nutzen der Fusion eines Artikels über Stringmatchingalgorithmen aus dem Jahre 2005 mit einem Artikel über Prozessortechniken aus dem Jahre 2008 ist völlig indiskutabel. Solche Fälle werden wir daher erst gar nicht behandeln und bei zwei Objekten, deren Fusion keinen Sinn ergibt, davon sprechen, dass diese "nicht fusioniert werden können".

Eine Fusion ist also nur dann *sinnvoll*, wenn beide zu fusionierenden Informationen den gleichen realen Sachverhalt beschreiben, beispielsweise den gleichen Konferenzband, den gleichen Zeitschriftenartikel oder die gleiche reale Person. Nur in einem solchen Fall können wir von der Fusion ein sinnvolles Ergebnis erwarten, welches uns die nachfolgende Arbeit erleichtern kann.

Die Art und Weise, in der die Fusion stattfindet, soll zudem *geeignet* sein, d.h. je nach betrachtetem Typ der Information wird die Fusion stets anders aussehen. Die Fusion zweier Autorennamen unterscheidet sich offensichtlich stark von der Fusion zweier Seitennummern, und bei der Fusion zweier Artikel müssen wir selbstverständlich anders vorgehen als bei der Fusion zweier Konferenz- oder Zeitschriftenbände. Einige der Informationen, die wir verschmelzen möchten, sind atomar, d.h. sie bestehen aus semantischer Sicht nur aus einem einzigen Wert (beispielsweise *ein* Titel, *eine* Seitenangabe oder auch *ein* kompletter Artikel), andere wieder-

um bestehen aus einer Reihe gleichartiger Werte (beispielsweise eine Liste von Autoren, eine Liste von Artikeln innerhalb eines Bandes oder Heftes).

Im folgenden Abschnitt werden wir nun zunächst – analog zur Informationsextraktion – einige praktische Anwendungsszenarien definieren, in welchen wir die Fusion bibliographischer Daten konkret einsetzen möchten. Anschließend werden wir darauf aufbauend in Abschnitt 6.2 einige Definitionen treffen, sowie eine formale Notation einführen, die uns zur Beschreibung der bei der Fusion auftretenden Sachverhalte in den folgenden Kapiteln behilflich sein wird. Insbesondere unterscheiden wir hier zwischen trivialer und nicht-trivialer Fusion bzgl. der Umsetzung in der Software. In Abschnitt 6.2.3 werden die so genannten "Fusions-Modi" definiert, mit deren Hilfe die zuvor beschriebenen Arten der Fusion umgesetzt werden können.

6.1 Anwendungsszenarien

Bevor wir uns der Theorie der Fusion der einzelnen bibliographischen Objekte zuwenden, wollen wir zunächst die Ziele, die wir mit der Fusion verfolgen, klar herausstellen. Dies wird es uns erleichtern festzulegen, wie das jeweilige Fusionsergebnis der Objekte aussehen soll.

Prinzipiell lassen sich zwei Fälle unterscheiden, in denen die Fusion zweier Datenbestände denkbar ist, nämlich

1. bei der Erfassung neuer Datensätze, und

2. bei der Aufbereitung bestehender Datensätze.

Analog zu Abschnitt 4.1 beschreiben wir dies anhand der folgenden Szenarien.

6.1.1 Szenario F-1: Fusion zweier BHT-Dateien

Beschreibung Zwei $\mathrm{BHT_{c/j}}$-Dateien des gleichen Typs (also entweder zweier $\mathrm{BHT_c}$-, oder aber zweier $\mathrm{BHT_j}$-Dateien) sollen fusioniert werden. Das Ergebnis sei wieder eine Datei gleichen Typs, in welcher die Informationen aus der primären Quelle gezielt durch einzelne Informationen der sekundären Quelle angereichert wurden.

Erläuterung Dieses Szenario erscheint am einfachsten, da wir lediglich zwei der $\mathrm{BHT_{c/j}}$-Dateien, die wir durch die zuvor durchgeführte Extraktion gewonnen haben, als Eingabe erwarten. Abbildung 6.1 verdeutlicht diesen Vorgang. Beide Dateien müssen lediglich in unser internes Datenformat (vgl. Kapitel 2.4), welches wir bereits bei der Extraktion der Daten benutzt haben, umgewandelt werden. Die Fusion zweier $\mathrm{BHT_{c/j}}$-Dateien kann auf diese Weise auf die Fusion zweier Record-Listen ($L^R{}_1$ und $L^R{}_2$) zurückgeführt werden.

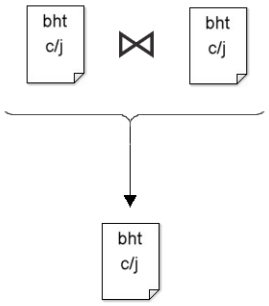


Abb. 6.1: Szenario F-1: Fusion zweier BHT-Dateien

6.1.2 Szenario F-2: Fusion bestehender Records mit einer BHT-Datei

Wollen wir bestehende Datensätze durch Informationen einer zusätzlichen Quelle anreichern, so müssen wir uns dem internen Format von DBLP annehmen. Wie in Kapitel 2 beschrieben, liegen die einzelnen Einträge in simplen Textdateien, den so genannten *records* vor. Es läge also nahe, ein Verzeichnis mit derartigen *records* als primäre Quelle zu definieren und diese mit einer $\mathrm{BHT}_{\mathrm{c/j}}$-Datei, in welcher die neu gewonnenen Informationen vorliegen, zu fusionieren.

Beschreibung Ein Verzeichnis mit *records* soll mit einer $\mathrm{BHT}_{\mathrm{c/j}}$-Datei fusioniert werden. Einzelne Attribute der records sollen hierbei ersetzt werden; als Ergebnis erhalten wir wiederum ein Verzeichnis mit *records*.

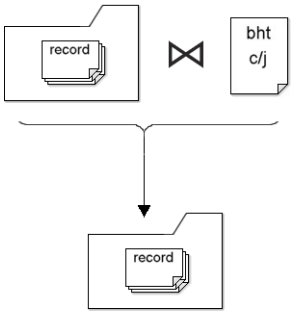


Abb. 6.2: Szenario F-2: Ergänzung bestehender DBLP-Records

Erläuterung In diesem Szenario (vgl. Abbildung 6.2) werden erstmals Daten betrachtet, die bereits in DBLP erfasst sind. Die Software muss nun also auch in der Lage sein, *records* einzulesen und ebenfalls intern in eine Record-Liste ($L^R{}_1$) umzuwandeln. Diese kann dann mit der zweiten Liste ($L^R{}_2$), die wir wie in Szenario F-1 aus einer $\mathrm{BHT}_{\mathrm{c/j}}$-Datei erzeugen, fusioniert

werden. Bei der direkten Fusion spielt es also keine Rolle, ob die erste Quelle eine $BHT_{c/j}$-Datei oder ein Verzeichnis mit *records* ist – es werden in jedem Fall zwei Record-Listen fusioniert.

Es liegt nun zunächst nahe, als Ergebnis ebenfalls eine $BHT_{c/j}$-Datei auszugeben, denn auf diese Weise könnten wir ebenso wie in Szenario F-1 verfahren und würden keine zusätzlichen Algorithmen zur Transformation einer Record-Liste in ein Verzeichnis mit *records* benötigen. Jene $BHT_{c/j}$-Datei könnte anschließend wieder in DBLP eingetragen werden als handele es sich um neue Daten, und hierbei könnten die alten Werte entsprechend überschrieben werden. Dies birgt jedoch ein hohes Risiko. Wie in Kapitel 2.2.3 erläutert, wird bei der Erfassung der Daten eine $BHT_{c/j}$ in eine BHT_{cite}-Datei und *records* aufgespalten. Jedes *record* erhält hier von der verarbeitenden Software automatisch einen festen und eindeutigen Schlüssel, der sich wie in Kapitel 2.2.1 beschrieben im Wesentlichen aus den Autorennamen und dem Jahr der Veröffentlichung zusammen setzt. Würde sich durch die Fusion der Name des Autors ändern (beispielsweise "Schultze" statt "Schulze"), so würde bei einem Neueintrag nach obiger Idee ein Schlüssel ungleich dem alten generiert werden (beispielsweise "/conf/abcde/Schultze09" statt "/conf/abcde/Schulze09"). Das entsprechende alte *record* würde nicht überschrieben werden und es käme zu Inkonsistenzen im Datenbestand von DBLP.

Daher müssen wir fordern, dass die Software die Ergebnisse auch wieder direkt in Form von *records* ausgibt. Indem wir den jeweiligen Schlüssel im Attribut *key* unseres Datenmodells speichern, können wir stets eine Zuordnung zum entsprechenden *record* der Quelle herstellen. Damit ist sicher gestellt, dass die Schlüssel in jedem Fall erhalten bleiben, auch wenn sich die Namen der Autoren ändern.

Zusätzlich soll die Möglichkeit bestehen, *records* entweder direkt zu überschreiben, oder aber Kopien der veränderten *records* in einem anderen Verzeichnis anzulegen. Im ersten Fall sollten die Original-Dateien in jedem Fall in einem Backup-Verzeichnis abgelegt werden, falls ein Fehler auftrat und die alten Daten wiederhergestellt werden sollen. In beiden Fällen ist zu beachten, dass *records* nur dann verändert werden sollten, wenn bei der Fusion auch tatsächlich Unterschiede gefunden wurden. Die Software muss daher in der Lage sein, aus der Record-Liste einzelne *records* zu generieren, diese dann mit den *records* der Quelle zu vergleichen, und im Fall der Gleichheit keine weiteren Aktionen durchzuführen. Dies bewirkt, dass das Datum der letzten Änderung (`mdate`) auch nur bei jenen Dateien auf den aktuellen Zeitpunkt gesetzt wird, in denen auch wirklich etwas verändert wurde. Dies ist beispielsweise für die Erstellung der DBLP-History-Datei (`dblph.xml`), welche neben den aktuellen DBLP-Records auch Änderungen (mit dem entsprechenden Änderungsdatum) enthält, notwendig.

Bei der Konstruktion des Ergebnis-Records sind einige weitere Vorsichtsmaßnahmen einzuhalten. Die Reihenfolge der Attribute innerhalb eines *records* ist nicht fest definiert und sollte im Ergebnis die gleiche sein wie in der Quelle. Nicht sichtbare Sonderzeichen wie Zeilenumbrüche sollen ebenfalls identisch bleiben. Zudem sind einige Attribute, die in BibTeX existieren und somit auch innerhalb der *records* auftreten können (wie beispielsweise 'editor' oder 'url'), in unserem Datenmodell nicht definiert. Daher soll bei der Erstellung der Ergebnis-Records wie folgt vorgegangen werden:

Zunächst besteht das Ergebnis aus dem ursprünglichen *record*, welches aus der ersten Quelle eingelesen wird und in XML-Syntax vorliegt. Daraufhin werden lediglich die Inhalte all jener XML-Elemente ersetzt, die sich durch die Fusion geändert haben; Die Tags selbst bleiben unverändert. Somit müssen bei einer Ersetzung keine Tags verändert werden, sondern stets nur deren Inhalte (vgl. Abb. 6.3). Wurde ein Attribut gefunden, welches zuvor im *record* nicht vorhanden war, so wird dieses am Ende, unmittelbar vor dem abschließenden Tag, eingefügt.

```
<inproceedings key="conf/aose/ParunakO01">
<author>H. Van Dyke Parunak</author>
<author>James Odell</author>
<title>Representing Social Structures in UML.</title>
<pages>1-16</pages>
<year>2001</year>
<crossref>conf/aose/2001</crossref>
<booktitle>AOSE</booktitle>
<ee>http://link.springer.de/link/service/series/0558/bibs/2222/22220001.htm</ee>
<url>db/conf/aose/aose2001.html#ParunakO01</url>
</inproceedings>
```

```
<inproceedings key="conf/aose/ParunakO01">
<author>H. Van Dyke Parunak</author>
<author>James J. Odell</author>
<title>Representing Social Structures in UML.</title>
<pages>1-16</pages>
<year>2001</year>
<crossref>conf/aose/2001</crossref>
<booktitle>AOSE</booktitle>
<ee>http://dx.doi.org/10.1007/3-540-70657-7_1</ee>
<url>db/conf/aose/aose2001.html#ParunakO01</url>
</inproceedings>
```

Abb. 6.3: Beispiel der Ersetzung von Daten in DBLP-Records: Nur die grau markierten Bereiche wurden modifiziert, der Rest der Datei wurde aus dem ursprünglichen *record* übernommen.
Quelle: Internes DBLP-Record `/dblp/publ/conf/aose/aose2001` (LNCS Volume 2222, Artikel 1)

Trotz all jener Überlegungen birgt Szenario F-2 dennoch zwei gewaltige Nachteile:

1. In einem Verzeichnis befinden sich i.d.R. *alle* Records einer Konferenz bzw. einer Zeitschrift. Eine $\text{BHT}_{\text{c/j}}$-Datei enthält dagegen lediglich ein einzelnes *volume*.

2. Die Records beinhalten, wie aus Kapitel 2 bekannt, keine Zwischenüberschriften, denn jene befinden sich in der entsprechenden BHT_{cite}-Datei.

Es liegt also nahe, Szenario F-2 nicht in dieser o.g. Form zu implementieren, sondern in einer etwas modifizierten Weise:

6.1.3 Szenario F-2': Fusion einer BHT_{cite}-Datei mit einer $\text{BHT}_{\text{c/j}}$-Datei

Beschreibung Eine BHT_{cite}-Datei soll mit einer $\text{BHT}_{\text{c/j}}$-Datei fusioniert werden. Die Software, welche die Fusion durchführt (im Folgenden als "Merger" bezeichnet) lädt automatisch die innerhalb der BHT_{cite}-Datei angegebenen *records* und fusioniert diese intern mit den Daten der $\text{BHT}_{\text{c/j}}$-Datei. Anschließend werden die modifizierten *records* in einem Verzeichnis ausgegeben. Zudem erstellt der Merger bei Bedarf (d.h. falls das Quellverzeichnis dem Zielverzeichnis entspricht) eine Sicherheitskopie der Originale.

Erläuterung Dieses Szenario mutet bereits etwas komplexer an. Abbildung 6.4 zeigt eine schematische Darstellung dieser Vorgänge. Die Merger-Software wurde hier als Box dargestellt, um die verschiedenen Ein- und Ausgabedaten besser darstellen zu können. Die Ausgabe der

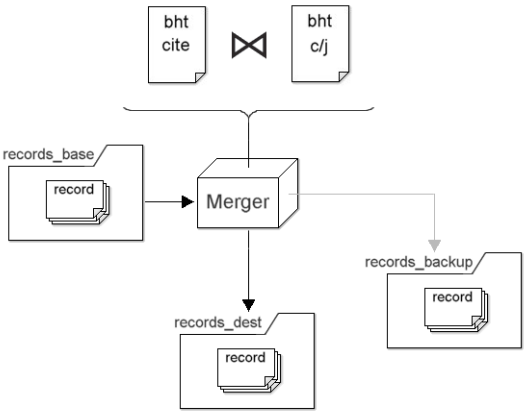


Abb. 6.4: Szenario F-2': Fusion einer $\mathrm{BHT}_{\mathrm{cite}}$-Datei mit einer $\mathrm{BHT}_{\mathrm{c/j}}$-Datei

records entspricht hierbei exakt der in Szenario F-2 beschriebenen Vorgehensweise, denn auch dort forderten wir die Möglichkeit, Sicherheitskopien der Original-Records anzufertigen.

Verändert hat sich allerdings die Eingabe. Diese besteht nun lediglich aus einer $\mathrm{BHT}_{\mathrm{cite}}$-Datei, welche von der Software geparst werden muss. Wie in Kapitel 2.2.2 beschrieben, enthält diese `<cite>`-Tags innerhalb ungeordneter Listen:

```
<ul>
<li><cite key="conf/aose/WeissFNR05" style=ee>
<li><cite key="conf/aose/CheongW05" style=ee>
<li><cite key="conf/aose/CervenkaTC05" style=ee>
</ul>
```

Diese `<cite>`-Tags enthalten die Schlüssel der zugehörigen *records*, und aus diesen lässt sich bekanntermaßen sofort der Dateiname und -pfad ermitteln. Die Software muss also lediglich die entsprechenden Dateien einlesen und anschließend wie in Szenario F-2 verfahren. Abgesehen von der modifizierten Eingabe sind die Szenarien F-2 und F-2' also identisch, vor allem in Bezug auf die Ausgabe der *records*.

In der praktischen Umsetzung der Merger-Software wurden nur die Szenarien F-1 und F-2' implementiert. Ein direktes Einlesen einzelner *records* aus einem Verzeichnis, wie es Szenario F-2 fordert, ist nicht möglich, da dies aus o.g. Gründen nicht zu empfehlen ist.

6.1.4 Szenario F-2'$_{\text{LNCS}}$: Austausch alter URLs gegen DOIs in den LNCS

Prinzipiell decken die beiden definierten Szenarien sämtliche Anforderungen an die Software ab. Es ist jedoch sinnvoll, einen Spezialfall des modifizierten zweiten Szenarios zu betrachten. Viele ältere Bände der LNCS-Reihe des Springer-Verlags (vgl. Kapitel 3.2.10) enthalten, wie in Kapitel 2.3.4 im Kontext der DOIs beschrieben, URLs, die teilweise nicht mehr gültig sind. Mittlerweile wurden jedoch bei Springerlink oftmals DOIs nachgetragen, welche in DBLP noch fehlen. Nach Szenario F-2' ist es uns nun jedoch ohne Weiteres möglich, diese zu ergänzen: Zunächst müssen die entsprechenden Daten eines Volumes mittels des Wrappers extrahiert und in einer BHT$_{\text{c}}$-Datei abgespeichert werden, anschließend wird die BHT$_{\text{cite}}$-Datei des entsprechenden Volumes mit dieser fusioniert. Da es sich hier jedoch um mehrere tausend Bände handelt, bietet es sich an, diesen Vorgang weiter zu automatisieren.

Beschreibung Durch Angabe einer oder mehrerer Bandnummern der LNCS-Reihe soll eine automatische Bearbeitung dieser Daten erfolgen. Hierzu soll die Software eigenständig die BHT$_{\text{cite}}$-Datei des Bandes ausfindig machen und diese mit einer BHT$_{\text{c}}$-Datei gemäß Szenario F-2' fusionieren, welche unter Verwendung der Wrapper-Software ebenfalls eigenständig erzeugt wurde.

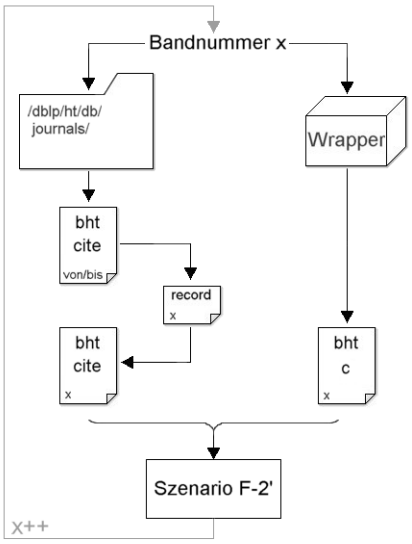


Abb. 6.5: Szenario F-2'$_{\text{LNCS}}$: Austausch alter URLs gegen DOIs in den LNCS

Erläuterung Wie man Abbildung 6.5 entnehmen kann, ist dieses Szenario lediglich eine Erweiterung des Szenarios F-2'. Die Software erhält zunächst einen Wert x, der die Bandnummer der LNCS-Serie angibt. Dieser Wert wird nun in zwei Richtungen propagiert. Auf der rechten Seite läuft hierbei ein reiner Extraktionsvorgang gemäß Szenario E-1 (Kapitel 4.1.1) ab, bei welchem eine BHT$_{\text{c}}$-Datei mit den Daten des Bandes x entsteht. Der auf der linken Seite skizzierte

Ablauf ist dagegen etwas spezifischer und erfolgt in zwei Schritten. Entsprechend dem Aufbau der HTML-Übersichtsseite der LNCS[1] muss hier zunächst eine BHT_{cite}-Datei ausgelesen werden, welche Informationen zum gewünschten Band x enthält. All jene Dateien befinden sich im Verzeichnis `/dblp/ht/db/journals/`[2] und tragen Namen der Form `lncs1400-1499.bht`, d.h. jede Datei beinhaltet Verweise zu jeweils 100 Bänden. Aus jener Datei, in welcher die Informationen zu Band x stehen, muss nun ein entsprechendes `<cite>`-Tag ausgelesen werden, welches den Schlüssel des *records* enthält, in welchem die Bandinformationen abgespeichert sind. Dieses lässt sich durch die voranstehende Bandnummer leicht identifizieren:

```
<dt>2201</dt><dd><cite key="conf/huc/2001" style="series"></dd>
<dt>2202</dt><dd><cite key="conf/ictcs/2001" style="series"></dd>
<dt>2203</dt><dd><cite key="conf/esaw/2001" style="series"></dd>
```

Innerhalb des *records* des Bandes x findet sich wiederum ein `<url>`-Attribut, aus welchem der Name der gesuchten BHT_{cite}-Datei des Bandes x ausgelesen werden kann. Damit besitzt die Software alle nötigen Voraussetzungen zur Fusion gemäß Szenario F-2': eine BHT_{cite}- und eine BHT_c-Datei. Ist der Fusionsvorgang abgeschlossen, so wird ggf. die Bandnummer x inkrementiert und mit dem nächsten Band fortgefahren.

Wie bereits erwähnt, ergibt die Implementierung dieses äußerst spezifischen Szenarios wegen der großen Menge an zu bearbeitenden Bänden einen Sinn. Die beiliegende Software enthält hierzu ein eigenes Kommando (`fixLNCS`, siehe Anhang A.3.2).

6.2 Fusion zweier bibliographischer Objekte

Nachdem wir unsere Anforderungen und Ziele definiert haben, wissen wir, dass die Fusion der bibliographischen Daten entsprechend den im vergangenen Abschnitt definierten Praxisszenarien stets auf die Fusion zweier Record-Listen unseres internen Datenmodells zurückgeführt werden kann. Wir führen an dieser Stelle also die erste Phase der Datenintegration, das in Kapitel 5.3.1 beschriebene 'Schema Mapping', durch, indem wir die Daten beider Quellen in das gleiche Format transformieren. Da Record-Listen (L^R) aus Records (R) bestehen, und diese sich wiederum nur durch deren Attribute (*title*, *authors* etc.) definieren, muss eine Fusion der Listen auf eine Fusion von Records, und diese wiederum auf die Fusion der Attribute zurück geführt werden.

Um im Folgenden stets exakt über das betrachtete Datenmodell sprechen zu können, sind einige Definitionen notwendig, mit deren Hilfe es uns möglich sein wird, die untersuchten Sachverhalte eindeutig zu beschreiben. Auch wenn die Definitionen recht mathematisch anmuten, so werden

[1] `http://dblp.uni-trier.de/db/journals/lncs.html`,
siehe auch die rechte Seite von Abbildung 3.12 auf Seite 65

[2] Die Wahl des Verzeichnisses `journals` ist historisch bedingt und sollte hierbei nicht weiter verwirren.

wir keinerlei Beweise o.ä. durchführen, sondern sie lediglich zur Formulierung untersuchter Tatbestände nutzen.

Betrachten wir also die bibliographischen Objekte des in Kapitel 2.4 definierten Datenmodells gemäß der dort vorgestellten Terminologie. Ein solches dort beschriebenes Objekt kürzen wir im allgemeinen Fall mit einem kleinen o ab. Sprechen wir über ein spezielles Objekt, so benennen wir es mit dessen Typ und evtl. einem entsprechenden Index, also z.B. R für ein Record, ${L^R}_1$ für eine Liste von Records der ersten Quelle oder $title_2$ für den Titel eines Records der zweiten Quelle. Allgemein bezeichnen wir Attribute mit einem kleinen a.

Den Typ eines Objektes werden wir mit $o.type$ ansprechen, dessen Wert mit $o.value$. Falls also beispielsweise der Titel eines Artikels "Informatik heute" lautet, so können wir $a.type = title$ und $a.value =$ "Informatik heute" schreiben, oder das Attribut a gleich als $title$ bezeichnen mit $title.value =$ "Informatik heute". Da bei dieser Schreibweise der Typ bereits im Namen auftaucht, schreiben wir bei Attributen hin und wieder auch einfach $title =$ "Informatik heute" statt $title.value =$ "Informatik heute".

Bei komplexen Objekten besteht der Wert aus einer Liste einzelner Elemente, welche wir als mathematisches Tupel auffassen können, also z.B. ${L^R}_1.value = (R_{11}, R_{12}, \cdots, R_{1n})$, wobei der erste Index die Quelle (1 oder 2) benennt und der zweite entsprechende der Anzahl der Objekte inkrementiert wird.

Es ist auch möglich, dass Attribute nicht gesetzt sind. Entsprechend der Definition innerhalb unseres Datenmodells werden wir dann auch davon sprechen, dass solche Attribute den Wert *null* haben und "$a.value = null$" schreiben.

Der Vorgang, bei welchem zwei bibliographische Objekte gleichen Typs zu einem neuen Objekt des gleichen Typs zusammengefasst werden, kann auf unterschiedliche Art und Weise geschehen. Prinzipiell ist hier eine Unterscheidung zwischen trivialer und nicht-trivialer Fusion möglich.

6.2.1 Triviale Fusion

Eine – aus Sicht der Softwareentwicklung – *triviale Fusion* liegt dann vor, wenn der Benutzer vorgibt, welcher Wert zu wählen ist. Wir nennen dies trivial, da die Programmierung dieser Selektion entsprechend trivial ist: Die Software übernimmt lediglich den vorbestimmten Wert ins Ergebnis. Dieser Fall der Konfliktlösung kommt der in Abschnitt 5.3.3 beschriebenen Strategie PASS IT ON recht nahe, jedoch entscheidet der Benutzer hier nicht erst bei Auftreten eines Konfliktes, sondern bereits im Vorhinein – bei Programmaufruf – und legt seine Entscheidung global für einen Typ von Objekten fest.

Beispiel 6.1
Betrachten wir die beiden *title*-Attribute mit

$$title_1.value = \text{"Grundlagen der Informationsextraktion"} \quad \text{und}$$
$$title_2.value = \text{"Grundlagen der Informations-Extraktion"}.$$

Hat der Benutzer im Vorhinein festgelegt, dass bei der Fusion der bibliographischen Daten stets der erste Datensatz gewählt werden soll, so werden diese Namen zum Ergebnis $title_{result}$ = "Grundlagen der Informationsextraktion" fusioniert. Entsprechend lautet das Ergebnis im Falle, dass der Benutzer stets den zweiten Datensatz wünscht, $title_{result}.value$ = "Grundlagen der Informations-Extraktion".

Die triviale Fusion stellt also stets eine simple *Selektion* dar. Mittels der in Abschnitt 6.2.3 definierten Fusions-Modi wird es dem Benutzer der Merge-Software möglich sein, eben jene Auswahl zu treffen.

In manchen Fällen ist es sinnvoll, manuell entscheiden zu können, welcher Datenquelle das Ergebnis entnommen werden soll. Stellen wir uns beispielsweise den folgenden, in der Praxis bereits aufgetretenen, Fall vor:

Beispiel 6.2
In IEEE Xplore (vgl. Kapitel 3.2.7) wurde eine neue Konferenz eingetragen, allerdings wurde hierbei ein Fehler begangen: Die angegebenen DOIs sind falsch und führen nicht zum gewünschten Artikel. Dieser Fehler wird von Seiten der IEEE nach einigen Tagen bemerkt und korrigiert, jedoch wurden die fehlerhaften Datensätze zwischenzeitlich in DBLP eingetragen. Zur automatischen Korrektur dieser Fehler kann die Merge-Software verwendet werden: Zunächst werden die Daten mittels des Wrappers erneut in eine BHT_c-Datei eingelesen, danach werden die bestehenden *records* mit jener Datei fusioniert. Wir erhalten daher Wertepaare der Form

$$ee_1.value = \text{"}\texttt{http://dx.doi.org/[falscherDOI]}\text{"} \quad \text{und}$$
$$ee_2.value = \text{"}\texttt{http://dx.doi.org/[korrekterDOI]}\text{"}.$$

Der Software wird es nicht – oder zumindest nicht ohne erheblichen Aufwand – möglich sein, diese beiden Werte automatisch zu fusionieren, da sie anhand der Syntax nicht feststellen kann, welcher DOI korrekt ist. Da wir jedoch wissen, dass die sekundäre Quelle die korrekten DOIs enthält, wählen wir hier die triviale Fusion und erhalten das gewünschte Ergebnis.

Die triviale Fusion birgt jedoch zwei entscheidende Nachteile:

1. Sie muss pro Typ stets global festgelegt werden.

2. Sie stellt stets eine Selektion dar.

Der erste Punkt besagt, dass wir also beispielsweise entscheiden können, dass das *ee*-Attribut immer der zweiten, der Titel aber immer der ersten Quelle entnommen werden soll. Wir können jedoch nicht angeben, dass der Titel des fünften Records dann doch der sekundären Quelle entstammen soll. Eine derartige Differenzierung ist nicht möglich.

Viel wichtiger ist jedoch der zweite Punkt, wie das folgende Beispiel zeigt:

Beispiel 6.3
Betrachten wir die beiden Namensattribute mit

$$aname_1.value = \text{"H.-P. Wurst"} \quad \text{und}$$
$$aname_2.value = \text{"Hans Wurst"}.$$

Wie wir erkennen können, beinhalten beide Attribute Informationen, die im jeweils anderen Attribut nicht enthalten sind. Eine Selektion mittels der trivialen Fusion wäre also in beiden Fällen – je nachdem, ob wir uns für die erste oder zweite Quelle entscheiden – stets unzureichend, da das von uns intuitiv geforderte Ergebnis ($aname_{result}.value$ = "Hans-P. Wurst") auf diese Weise nicht erreicht werden kann. Dies ist nur mittels der nicht-trivialen Fusion möglich, welche wir im folgenden Abschnitt eingehend untersuchen werden.

Es sei aber noch angemerkt, dass eine triviale Fusion nur bei Attributen, nicht aber bei Records sinnvoll ist: Sollen stets nur die Records der primären oder sekundären Quelle ins Ergebnis eingehen, so benötigt man keine Fusion, sondern kann gleich die unveränderte Quelle als Ergebnis ausgeben.

6.2.2 Nicht-triviale Fusion

Im Gegensatz zum vorherigen Abschnitt wollen wir eine Fusion als *nicht-trivial* bezeichnen, wenn die Software selbst mittels geeigneter Algorithmen ein Fusionsergebnis produzieren, d.h. die auftretenden Konflikte selbständig vermeiden oder auflösen kann – oder es zumindest versuchen soll. Ist im Folgenden lediglich von einer "Fusion" die Rede, so ist damit stets eine nicht-triviale Fusion gemeint. Wir werden zunächst eine Notation definieren, mittels derer wir anschließend Fusionsregeln für sämtliche betrachtete bibliographische Objekte angeben können.

Der 'merge'-Operator Können zwei bibliographische Objekte o_1 und o_2 des gleichen Typs sinnvoll zu einem Objekt o_{result} des gleichen Typs zusammengefasst werden, so dass der Wert von o_{result} den gesamten Informationsgehalt von o_1 und o_2 beinhaltet, dann bezeichnen wir diesen Vorgang als *(nicht-triviale) Fusion* und schreiben dafür kurz:

$$o_1 \bowtie o_2 \longrightarrow o_{result}.$$

Diese recht intuitive Notation sollte in der Art "o_1 fusioniert mit o_2 ergibt o_{result}" oder "o_1 und o_2 werden zu o_{result} fusioniert" gelesen werden. Eine äußerst praktische und noch kürzere Lesart stellt "o_1 merge o_2 ergibt o_{result}" dar.[3]

[3]Die zur Arbeit gehörige Software, welche die hier vorgestellten Konzepte umsetzt, wird durchgängig als 'Merger' bezeichnet. Im vorliegenden Text wurde jedoch aus ästhetischen Gründen auf eine unschöne Eindeutschung dieses Begriffs verzichtet und tunlichst vermieden, von der *Mergebarkeit* zweier Objekte zu schreiben. Die Bezeichnung des Operators als 'merge'-Operator allerdings klingt recht passabel und ist zudem äußerst praktisch.

Als Operator wurde bewusst das aus der relationalen Algebra der Datenbanktheorie bekannte *JOIN*-Symbol gewählt, das dort ebenfalls eine Art der Fusion – die Fusion zweier Datentabellen – beschreibt. Daher ist die Bedeutung jenes Operators intuitiv verständlich; In den schematischen Darstellungen der Szenarien zu Anfang dieses Kapitels haben wir ihn bereits im Kontext zweier zu fusionierender Datenquellen verwendet. Es besteht allerdings ein Unterschied zum JOIN-Operator, der zwar Zeilen und Spalten von Tabellen verändert, niemals jedoch die Inhalte einzelner Zellen. Bei der Fusion (beispielsweise zweier Autorennamen) ist es jedoch unser Ziel, mittels des 'merge'-Operators aus zwei Werten einen dritten zu generieren, der zuvor nicht im Datenbestand enthalten war.

Es ist zu beachten, dass obige Schreibweise impliziert, dass die beiden Objekte auch tatsächlich fusioniert werden können und das Ergebnis ebenfalls ein Objekt des gleichen Typs darstellt. Der binäre Operator '$\bowtie$' ist damit stets im mathematischen Sinne abgeschlossen.

Es ist jedoch auch möglich, dass Objekte nicht fusioniert werden können. Hierfür werden wir

$$o_1 \not\bowtie o_2$$

schreiben und dies als "o_1 und o_2 können nicht fusioniert werden" ("o_1 not mergeable o_2") interpretieren.

Diese Definition ist theoretisch zwar recht nützlich, mit Blick auf die Implementierung jedoch etwas umständlich. Wir möchten aus Gründen der Effizienz nicht erst testen, ob zwei Objekte fusioniert werden können und im positiven Fall anschließend das Ergebnis berechnen. Statt dessen werden wir sofort versuchen, ein entsprechendes Ergebnis zu erzeugen und im Fall, dass an einer Stelle des entsprechenden Algorithmus ein Konflikt auftritt, der nicht automatisch zu lösen ist, ein fest definiertes Ergebnis < FAILED > zurück liefern. Daher definieren wir:

$$o_1 \bowtie o_2 \longrightarrow < \text{FAILED} > \quad :\Leftrightarrow \quad o_1 \not\bowtie o_2$$

Wie bereits zu Beginn dieses Kapitels erläutert, können Objekte unterschiedlicher Typen niemals in sinnvoller Weise zusammengefasst werden. Da uns jedoch nur sinnvolle Ergebnisse interessieren und wir nicht beabsichtigen, Autorennamen mit Zeitschriftennummern zu verschmelzen, ergibt sich sofort Folgerung (6.1):

$$o_1.type \neq o_2.type \quad \Rightarrow \quad o_1 \not\bowtie o_2. \tag{6.1}$$

Selektion und Kombination Gehen wir davon aus, dass zwei Objekte fusioniert werden können, so kann das Ergebnis sich entweder aus einer *Selektion* oder einer *Kombination* der beiden Objekte ergeben:

Bei einer *Selektion* entspricht das Ergebnis-Objekt einem der beiden Quellen-Objekte, d.h. es wird lediglich nach unterschiedlichen Kriterien eine Auswahl getroffen. Es gilt also stets

$$o_1 \bowtie o_2 \longrightarrow o_1 \quad \text{oder} \quad o_1 \bowtie o_2 \longrightarrow o_2.$$

Eine *Kombination* dagegen zeichnet sich dadurch aus, dass als Ergebnis ein neues Objekt geschaffen wird, welches in dieser Form zuvor nicht existierte.

$$\begin{aligned} o_1 \bowtie o_2 \longrightarrow\ & o_{result} \\ & \text{mit} \quad o_{result}.value \neq o_1.value \\ & \text{und} \quad o_{result}.value \neq o_2.value \end{aligned}$$

Betrachten wir beispielsweise noch einmal obiges Beispiel 6.3: Bei den beiden Autorennamen mit den Werten $aname_1.value$ ="H.-P. Wurst" und $aname_2.value$ ="Hans Wurst" haben wir intuitiv festgestellt, dass die Selektion eines der beiden Werte kein optimales Ergebnis liefern würde. Beide Attributswerte enthalten Informationen, die im jeweils anderen Wert nicht vorhanden sind. Unser Ziel ist daher eine Kombination der Werte, im vorliegenden Fall zu einem neuen Attribut mit $aname_{result}.value$ ="Hans-P. Wurst".

Es leuchtet jedoch auch ein, dass in einigen Fällen eine Selektion die einzig sinnvolle Art der Fusion zweier Objekte darstellt. Betrachten wir beispielsweise zwei unterschiedliche *ee*-Attribute mit den Werten

$ee_1.value$ = "`http://www.abc.com/?id=12345`" und
$ee_2.value$ = "`http://www.xyz.net/?id=67890`.

Ein sinnvolles Fusionsergebnis ist offensichtlich nur dann zu erzielen, wenn einer der beiden Werte ausgewählt wird; eine Kombination zu einem neuen Wert, wie etwa

$ee_{result}.value$ = "`http://www.abc.net/?id=1234567890`",

ist völlig unsinnig.

Theoretisch können wir die Selektion als einen Spezialfall der Kombination ansehen: Eine Kombination völlig zu Gunsten einer der beiden Quellen. Im Gegensatz zur trivialen Fusion, bei der ebenfalls eine Selektion erfolgt, wird hier das Ergebnis jedoch nicht durch eine Benutzervorgabe bestimmt, sondern von der Software mittels verschiedener Algorithmen *berechnet*. Um eine Kombination zu erhalten, muss entsprechend der in Kapitel 5.3.3 vorgestellten Strategien zur Konfliktbehandlung stets eine vermittelnde ('mediating') Strategie gewählt werden.

6.2.3 Fusions-Modi

In den vorherigen Abschnitten wurde mehrfach erwähnt, dass ein Benutzer die Möglichkeit erhalten solle, die Fusion der bibliographischen Daten über die Wahl der Quellen hinaus zu beeinflussen. In manchen Fällen möchte der Benutzer, dass nur ganz bestimmte Attribute fusioniert werden (beispielsweise beim Austausch alter URLs gegen DOIs bei den Bänden der LNCS-Serie, siehe Kapitel 6.1.4), oder er möchte einen Teil der Attribute gerade *nicht* aus der primären, sondern stets aus der sekundären Quelle entnehmen, also eine triviale Selektion erzwingen. Daher definieren wir verschiedene *Fusions-Modi*, über welche der Benutzer die volle Kontrolle darüber erhält, welche Daten auf welche Art und Weise fusioniert werden sollen.

Die *Fusions-Modi* entscheiden darüber, wie das Ergebnis o_{result} einer Fusion zweier bibliographischer Objekte o_1 und o_2 aussehen soll:

- `mode = 1` $\Rightarrow$ $o_{result} = o_1$
- `mode = 2` $\Rightarrow$ $o_{result} = o_2$
- `mode = merge` $\Rightarrow$ $o_{result} = o_1 \bowtie a_2$

Die Modi `1` und `2` bilden also die triviale Fusion (vgl. Abschnitt 6.2.1) ab. Die Software wird in diesen Fällen stets einfach den entsprechenden Wert aus der primären bzw. sekundären Quelle entnehmen und dem Ergebnis hinzufügen. Der Modus `merge` dagegen entspricht der in Abschnitt 6.2.2 definierten nicht-trivialen Fusion.

Unabhängig von der Wahl eines der drei o.g. Fusions-Modi wird die Software dennoch das nicht-triviale Fusionsergebnis $o_1 \bowtie o_2$ berechnen und eine Meldung ausgeben, falls jenes vom mittels des Fusions-Modus gewählten Ergebnis abweicht. Diese 'Verbesserungsvorschläge' werden im Protokoll der Software, dem so genannten 'Mergelog', angezeigt und können bei einer manuellen Nachbearbeitung durchgeführt werden.

Es ist jedoch auch möglich, dass bestimmte Attribute bei der Fusion völlig ignoriert werden sollen. Aus diesem Grund definieren wir einen vierten Modus:

- `mode = ignore` $\Rightarrow$ $o_{result} = o_1$

Wie man sieht, sind die Ergebnisse der Modi `1` und `ignore` völlig identisch; es wird stets eine triviale Fusion zu Gunsten der primären Quelle durchgeführt. Im zweiten Fall wird jedoch keinerlei Berechnung bzgl. des entsprechenden Attributes durchgeführt und somit natürlich auch kein Verbesserungsvorschlag geliefert.

Kapitel 7

Fusion komplexer Objekte

In diesem und dem folgenden Kapitel wollen wir uns nun eingehend mit der Fusion der bibliographischen Objekte unseres internen Datenmodells beschäftigen, wobei wir uns zunächst den komplexen und im folgenden Kapitel dann den simplen Objekten widmen werden. Als *komplexe Objekte* bezeichnen wir alle Tupel, welche aus mehr als einem gleichartigen Objekt bestehen:

- Listen von Records: $L^R = (R_1, R_2, \cdots, R_n)$
- Listen von Autorennamen: $authors = (aname_1, aname_2, \cdots, aname_n)$
- Listen von Namensteilen: $aname = (n_1, n_2, \cdots, n_p)$

Letztgenannte Listen von Namensteilen (die einen Autorennamen ergeben) nehmen bei der Fusion eine Sonderstellung zwischen den simplen und komplexen Objekten ein, da die Namensteile nicht unabhängig voneinander betrachtet werden können. Wir werden die Fusion zweier Autorennamen daher gesondert und ausführlich in Kapitel 8.3 behandeln. Die im Folgenden vorgestellten Algorithmen zur Fusion komplexer Objekte finden daher nur in Bezug auf Record-Listen (R^L) und Listen von Autorennamen ($authors$) Anwendung.

Die zweite Phase des Datenintegrationsprozesses nach Bleiholder und Naumann ('Duplicate detection, siehe Kapitel 5.3.2) ist lediglich für komplexe Objekte von Bedeutung. Haben wir zwei Listen gleichartiger Objekte vorliegen, so müssen zunächst entsprechende Objekte gefunden werden, die den gleichen realen Sachverhalt darstellen und somit in sinnvoller Weise fusioniert werden können. Der allgemeine Fall der 'Duplicate detection' weicht hier dem speziellen Fall der *Partnersuche*, welche wir in diesem Kapitel definieren werden.

Als *Partner* wollen wir zwei Objekte gleichen Typs bezeichnen, die den gleichen realen Sachverhalt ausdrücken (also z.B. zwei Autorennamen, die die gleiche reale Person bezeichnen), wobei der erste Partner der ersten Quelle, der zweite Partner der zweiten Quelle entstammen muss. Ein solches Tupel aus zwei Partnern nennen wir *Paar*. Entsprechend bezeichnen wir diejenigen Objekte, zu denen kein Partner gefunden werden kann, als *Singles*.

Da ein Paar stets nur aus zwei Objekten besteht, kann es passieren, dass Singles auftreten, die dennoch entsprechend einer allgemeinen Duplikatserkennung mit weiteren Daten zusammen gefasst worden wären. Betrachten wir beispielsweise den folgenden Fall:

Beispiel 7.1
Die beiden folgenden Listen von Autorennamen sollen fusioniert werden:

$authors_1$ = ("H. Wurst","P. Wurst"),
$authors_2$ = ("Hans-Peter Wurst").

Bei einer allgemeinen Duplikatserkennung würden alle drei Namen als ähnlich eingestuft und anschließend zu einem einzigen Datensatz fusioniert. Bei der in diesem Kapitel beschriebenen Partnersuche dagegen würde das Ergebnis anders aussehen; hier würden das Paar ("H. Wurst","Hans-Peter Wurst"), sowie der Single ("P. Wurst") entstehen.

Wie in Kapitel 5.3.2 erläutert, benötigen wir hier zunächst geeignete Ähnlichkeitsfunktionen, welche wir in Abschnitt 7.1 für die jeweiligen Objekte definieren werden. Anschließend diskutieren wir in Abschnitt 7.2 geeignete Algorithmen zur Partnersuche, während wir in Abschnitt 7.3 erläutern, wie mit evtl. entstandenen Singles zu verfahren ist – da es sich bei jenen entweder um wirkliche zu eliminierende Dubletten (Abschnitt 7.4), oder aber zusätzliche Informationen handeln kann.

7.1 Ähnlichkeits-Algorithmen

Wir können i.A. nicht davon ausgehen, dass die Objekte beider zu fusionierender Listen in gleicher Reihenfolge auftreten. Daher sind Heuristiken nötig, welche in geeigneter Weise die Ähnlichkeit zweier Objekte berechnen, um somit den jeweils 'wahrscheinlichsten Partner' ermitteln zu können. Diese Ähnlichkeit soll der Wahrscheinlichkeit, dass es sich bei den beiden Objekten um Partner handelt, entsprechen. Auf diese Weise können wir stets Werte zwischen 0 und 1 betrachten, wobei eine Ähnlichkeit von 1 besagt, dass es sich *mit absoluter Sicherheit* um Partner handelt, während eine Ähnlichkeit von 0 entsprechend ausdrückt, dass es sich mit Sicherheit *nicht* um Partner handelt. Hierzu definieren wir eine Ähnlichkeitsfunktion *sim* (von engl. "similarity") mit

$$sim(o_1, o_2) \in [0; 1],$$

welche für je zwei Objekte die beschriebene Ähnlichkeit liefert. Wie dieser jeweilige Wert berechnet wird, hängt natürlich vom Typ der jeweiligen Objekte ab.

7.1.1 Ähnlichkeit zweier Records

Die Ähnlichkeitsberechnung zweier Records ist recht komplex, da verschiedene Attribute miteinander verglichen werden müssen. Da die Records eine Abstraktion realer Artikel einer Publi-

kation sind, soll die `sim()`-Funktion also einen Wert liefern, der angibt, wie ähnlich sich zwei Artikel sind. Hierzu werden verschiedene Attribute beider Records R_1 und R_2 miteinander verglichen. Sind sich diese ähnlich (oder gar gleich), so werden entsprechende Schlüsse bgzl. der Ähnlichkeit der Records gezogen. Es werden jedoch *keine* negativen Schlüsse gezogen, falls sich Werte nicht ähnlich sind, denn es ist ja gerade unser Ziel, neue Informationen zu erhalten, was stets voraussetzt, dass sich die Records in eben jenen Attributen unterscheiden. Gleiches gilt demnach natürlich auch für nicht gesetzte Attribute: Ist ein Attribut in einem der Records nicht gesetzt, so kann dieses nicht dazu verwendet werden, eine Aussage über eine Ähnlichkeit der Records abzugeben. Dies beinhaltet den Fall, dass das Attribut in *beiden* Records fehlt, denn auch dann wäre eine Annahme einer Ähnlichkeit unsinnig.

Wir werden also im Folgenden stets voraussetzen, dass beide Attribute einen Wert ungleich '*null*' enthalten. Zunächst definieren wir eine äußerst simple Regel zur Bestimmung der Ähnlichkeit, welche uns im Erfolgsfall eine schnelle Entscheidung garantiert:

$$ee_1.value = ee_2.value \quad \Rightarrow \quad R_1 = R_2$$

Sind also die Werte der beiden *ee*-Attribute der Records identisch, so handelt es sich *mit Sicherheit* um ein und denselben realen Artikel. In der Software können wir sofort folgern, dass '`sim(R1, R2) = 1.00`' gilt und brauchen keine weiteren Untersuchungen durchzuführen, denn unabhängig davon, ob es sich bei den Werten um DOIs oder URLs handelt, verweisen sie auf dieselbe digitale Ressource.

Hier zeigt sich auch, warum wir keine negativen Folgerungen ziehen dürfen. Es kann vorkommen, dass ein *ee*-Attribut den Wert eines DOIs beinhaltet, das andere einen URL. Unser Ziel ist ja gerade die Ersetzung jener Werte, also würde ein Schluss auf Ungleichheit keinen Sinn ergeben. Handelt es sich bei beiden Werten um DOIs, sind diese aber verschieden, so könnte angenommen werden, dass die Artikel in jedem Fall verschieden sind. Doch auch hier zeigt uns die Praxis, dass hin und wieder falsche DOIs vergeben wurden, welche gegen korrekte Werte ersetzt werden sollen – was bei Ausschluss einer Ähnlichkeit anhand unterschiedlicher Werte nicht mehr möglich wäre.

Kann also über die *ee*-Attribute keine Aussage bzgl. der Ähnlichkeit getroffen werden, so muss versucht werden, eine solche über Betrachtung anderer Werte zu erlangen. Dies geschieht innerhalb der entwickelten Software anhand der Werte der Attribute *title*, *authors*, *pages* und *issue*. Bei den beiden Erstgenannten kommen jeweils modifizierte Varianten des klassischen Levenshtein-Algorithmus' ([Lev66]) zum Einsatz, und bei den Seitenangaben wird untersucht, ob sich die Informationen zumindest nicht ausschließen.[1] Die zusätzliche Untersuchung des *issue*-Attributes hat den Sinn, gleich lautende Artikel in unterschiedlichen Heften aufzuspüren. Nehmen wir beispielsweise an, ein jeder Band eines Journals würde über einen Artikel "Editorial" verfügen, der stets vom gleichen Autoren geschrieben würde und stets auf der ersten Seite erschiene. Um diese verschiedenen Artikel unterscheiden zu können, ist die Betrachtung des *issue*-Wertes notwendig. Eine Betrachtung des *volume*-Wertes dagegen ist völlig nutzlos, da wir, bedingt durch die Aufspaltung mehrerer Volumes in mehrere BHT_j-Dateien, an dieser Stelle ohnehin stets nur die Daten eines einzigen Volumes betrachten.

[1] Die Angaben "4" und "4-9" schließen sich nicht aus und könnten auf zusätzliche Informationen hindeuten, während sich die Angaben "4-9" und "5-12" klar ausschließen.

Insgesamt wird eine Ähnlichkeit der Titel sehr hoch gewichtet, während die Ähnlichkeit anderer o.g. Attribute nur bei Gleichheit mehrerer Titel eine Tendenz geben soll. Eine Gleichheit (d.h. ein Ergebnis von `1.00` wird auf diese Weise jedoch niemals erreicht, denn eine 100%ige Sicherheit, dass es sich um exakt den gleichen Artikel handelt, kann nur durch die o.g. Identität der *ee*-Werte erreicht werden. In der Praxis werden mittels dieser Methode in den meisten Fällen Partner gefunden. Selbstverständlich sind an dieser Stelle jedoch weitere Verbesserungen an den Parameterwerten möglich.

7.1.2 Ähnlichkeit zweier Autorennamen

Bei den Autorennamen ist die Berechnung der Ähnlichkeit um ein Vielfaches einfacher als bei den Records, da wir hier lediglich die Ähnlichkeit zweier Strings vergleichen müssen. Sind die Strings identisch, so liefern wir sofort ein Ergebnis von 1 zurück. Ansonsten werden eine Reihe von Heuristiken angewandt, von denen am Ende der maximale Wert als Ähnlichkeit zurück geliefert wird. Ein einfacher Stringmatching-Algorithmus reicht hier bei Weitem nicht aus, da es ja gerade unser Ziel ist, unvollständige Namen der einen Quelle durch vollständige Informationen der anderen Quelle zu ergänzen und wir somit annehmen müssen, dass Teile eines Namens beispielsweise nur in Form von Initialen vorkommen.

Die exakten Berechnungen der Ähnlichkeitswerte sind dem Quelltext der Software direkt zu entnehmen. An dieser Stelle sollen jedoch einige Beispiele verdeutlichen, welche Fälle allesamt Berücksichtigung fanden.

```
 [1]  sim("Hans Wurst","Hans Wurst")       = 1.00
 [2]  sim("Hans Wurst","Häns Würst")       = 0.80
 [3]  sim("Hans Wurst","Hanswurst")        = 0.90
 [4]  sim("Hans Wurst","Wans Hurst")       = 0.80
 [5]  sim("Hans Wurst","Horst Wurst")      = 0.73
 [6]  sim("Hans Wurst","Hans Wurstbrot")   = 0.71
 [7]  sim("Hans Wurst","H. Wurst")         = 0.97
 [8]  sim("Hans Wurst","P. Wurst")         = 0.60
 [9]  sim("Hans Wurst","H.-P. Wurst")      = 0.64
[10]  sim("Hans Wurst","H.-Peter Wurst")   = 0.64
[11]  sim("Hans Wurst","Hans Honig")       = 0.50
[12]  sim("Hans Wurst","Hildegard Hirse")  = 0.33
[13]  sim("Hans Wurst","Alfred Abendbrot") = 0.19
[14]  sim("Hans Wurst","Otto Ohne")        = 0.10
```

Man sieht, dass der Fall `[7]`, bei welchem ein Partner über einen vollständigen Vornamen verfügt, der andere über ein Initial, welches zu diesem Vornamen ergänzt werden könnte, besonders begünstigt wird. Auch bei völlig unterschiedlichen Namen (in den Fällen `[11]` bis `[14]`) wird eine gewisse Ähnlichkeit angenommen, was auf die entsprechenden Stringmatching-Verfahren zurück zu führen ist. Sogar zu dem Namen `Otto Ohne`, der keinerlei gleiche Buchstaben aufweist, wird wegen der gleichen Anzahl an Namensteilen eine – wenn auch geringe

– Ähnlichkeit erreicht. Dies widerspricht natürlich dem Prinzip der Effektivität (vgl. Kapitel 5.3.2), nach welchem keine falschen Datensätze im Ergebnis auftreten dürfen. Wir werden solche Datensätze daher bei der Partnersuche in Abschnitt 7.2 durch die Wahl eines geschickten Schwellenwertes, wie er auch im Integrationsmodell von Bleiholder und Naumann Anwendung findet, eliminieren.

Die anderen Fälle, die eine gewisse Ähnlichkeit der Namen implizieren sollen, sind: Ähnlichkeit bis auf Sonderzeichen `[2]`, Verbindungen von Namensteilen `[3]`, Vertauschungen `[4]` oder Ergänzungen `[6]`. Bei `[5]` handelt es sich dagegen um offensichtlich unterschiedliche Namen, die jedoch eine recht hohe Ähnlichkeit aufweisen. Hier wird deutlich, warum wir nicht einfach den 'erstbesten' Namen wählen dürfen, zu dem eine recht hohe Ähnlichkeit besteht, sondern innerhalb einer Autorenliste stets nach dem Namen mit der höchsten Ähnlichkeit suchen müssen. In den Fällen `[8]` bis `[10]` besteht jeweils eine Ähnlichkeit zu einem der Namensteile, in `[9]` und `[10]` entspricht zudem das Initial dem ersten Buchstaben des Vornamens `Hans`, was eine höhere Gewichtung verursacht als in Fall `[8]`. Wir erkennen im Vergleich zwischen `[7]` und `[9]` aber auch, dass die Existenz weiterer Namensteile das Ergebnis sofort erheblich abschwächt, aber dennoch nicht unbedingt ausschließt. Hierbei ist es unerheblich, ob der zusätzliche Namensteil lediglich ein Initial (wie in `[9]`) oder ein vollständiger Vorname (wie in `[10]`) ist; beide Fälle werden gleich gewertet.

7.2 Partnersuche

Nun, da wir entsprechende Ähnlichkeitsfunktionen zur Verfügung haben, können wir mit der eigentlichen Suche nach geeigneten Fusionspartnern beginnen. Um diese zu finden, ist es in einem naiven Ansatz notwendig, jedes Objekt der einen Liste mit jedem der anderen zu vergleichen.

Bemerkung
Es reicht nicht aus, zu jedem Objekt der ersten Liste einfach dasjenige Objekt der zweiten Liste zu suchen, welches diesem am ähnlichsten ist. Betrachten wir zur Veranschaulichung das folgende Beispiel:

Beispiel 7.2
Nehmen wir die beiden folgenden Liste an:

$authors_1$ = ("Hans Wurst", "Klaus Wurst"),
$authors_2$ = ("K. Wurst").

Wie man unschwer erkennen kann, sind "Klaus Wurst" und "K. Wurst" Partner, während sich "Hans Wurst" und "K. Wurst" jedoch auch recht ähnlich sind. Würden wir nun auf triviale Weise die erste Liste durchlaufen und jedem Eintrag denjenigen zum Partner geben, der ihm am meisten ähnelt, so würde die Partnerbeziehung ("Hans Wurst", "K. Wurst") hergestellt. Diese

würden den entsprechenden Listen entnommen, um keine doppelten Zuordnungen zuzulassen, und somit wäre “Klaus Wurst” ein Single. Das Ergebnis wäre offensichtlich falsch.

7.2.1 Naiver Algorithmus zur Partnersuche

Zunächst werden wir betrachten, wie die Partnersuche prinzipiell ablaufen sollte. Der hieraus resultierende naive Algorithmus weist zwar noch einige erhebliche Schwächen auf, doch können wir diese anschließend in Abschnitt 7.2.2 ausmerzen oder zumindest verbessern.

Grundsätzlich müssen wir zunächst die Ähnlichkeit im Sinne einer ‘Jeder mit Jedem’-Beziehung berechnen. Daraus ergibt sich für zwei Listen $L_1 = (o_{11}, o_{12}, ..., o_{1n})$ und $L_2 = (o_{21}, o_{22}, ..., o_{2m})$ mit $n, m \in \mathbb{N}$ die $n \times m$-Matrix

$$SIM_{L_1,L_2} = \begin{pmatrix} sim(o_{11}, o_{21}) & sim(o_{11}, o_{22}) & \cdots & sim(o_{11}, o_{2m}) \\ sim(o_{12}, o_{21}) & sim(o_{12}, o_{22}) & \cdots & sim(o_{12}, o_{2m}) \\ \vdots & \vdots & \ddots & \vdots \\ sim(o_{1n}, o_{21}) & sim(o_{1n}, o_{22}) & \cdots & sim(o_{1n}, o_{2m}) \end{pmatrix} \in [0;1]^{n,m}. \quad (7.1)$$

Ist diese Matrix berechnet, so können wir mit folgendem simplen Algorithmus die entsprechenden Partner identifizieren:

[1] Berechne für alle $i = 1, ..., n$ und alle $j = 1, ..., m$ den Wert $sim(o_{1i}, o_{2j})$.

[2] Falls SIM_{L_1,L_2} leer ist, so breche ab.

[3] Bestimme $max(SIM_{L_1,L_2})$, den maximalen Wert in der Matrix.

[4] Bestimme die kleinsten Indizes i und j, für die $sim(o_{1i}, o_{2j}) = max(SIM_{L_1,L_2})$ gilt.

[5] Speichere das Tupel (o_{1i}, o_{2j}) als Partner.

[6] Streiche die i-te Zeile und die j-te Spalte aus SIM_{L_1,L_2}.

[7] Wiederhole ab 2.

Dieser simple Algorithmus schafft es tatsächlich, sämtliche Partner korrekt zu finden. Nachdem in [1] die Ähnlichkeitsmatrix berechnet wurde, von welcher wir annehmen wollen, dass sie zu Anfang noch nicht leer ist ([2]), werden nun im [4]. Schritt zwei Objekte identifiziert, deren Ähnlichkeit in der derzeitigen Matrix maximal ist ([3]). Haben mehrere Objekte die gleiche Ähnlichkeit, so werden jene mit kleinen Indizes bevorzugt, was der stärkeren Gewichtung der primären Quelle Rechnung trägt, da jenen Daten, die in dieser Quelle weiter vorne auftauchen, auch zuerst ein Partner zugeordnet wird. Danach werden beide Objekte als Partner vermerkt ([5]) und der Matrix entnommen, indem die entsprechende Zeile und Spalte gestrichen werden

([6]). Sie gehen somit in keine weitere Berechnung mehr ein. Entfernen wir die entsprechenden Objekte auch aus ihren ursprünglichen Listen, so sind nach Beendigung des Algorithmus entweder beide Listen leer (genau dann, wenn die Matrix quadratisch ist, also $n = m$ gilt), oder eine der Listen enthält noch weitere Objekte, die dann als Singles behandelt werden (siehe Abschnitt 7.3).

Machen wir uns die Arbeitsweise dieses Algorithmus an einem Beispiel klar:

Beispiel 7.3
Nehmen wir die folgenden beiden Autorenlisten an:

$authors_1$ = ("Alfons Abendrot", "Birgit Blumenkohl", "Gerhard Gibtsnicht") und
$authors_2$ = ("Birgit Blumenkohl", "Alfons Abendbrot", "Wilhem Werwohl").

Zur Berechnung der SIM-Matrix müssen nun in Schritt [1] neun Berechnungen durchgeführt werden, die je nach Definition der $sim()$-Funktion wie folgt aussehen könnten.[2]

```
(1)   sim("Alfons Abendrot", "Birgit Blumenkohl")    = 0,18
(2)   sim("Alfons Abendrot", "Alfons Abendbrot")     = 0,94
(3)   sim("Alfons Abendrot", "Wilhelm Werwohl")      = 0,13
(4)   sim("Birgit Blumenkohl", "Birgit Blumenkohl")  = 1,00
(5)   sim("Birgit Blumenkohl", "Alfons Abendbrot")   = 0,12
(6)   sim("Birgit Blumenkohl", "Wilhelm Werwohl")    = 0,29
(7)   sim("Gerhard Gibtsnicht", "Birgit Blumenkohl") = 0,22
(8)   sim("Gerhard Gibtsnicht", "Alfons Abendbrot")  = 0,17
(9)   sim("Gerhard Gibtsnicht", "Wilhelm Werwohl")   = 0,17
```

Die anfängliche Ähnlichkeitsmatrix würde dann wie folgt aussehen:

$$SIM_{authors_1,authors_2} = \begin{pmatrix} 0{,}18 & 0{,}94 & 0{,}13 \\ 1{,}00 & 0{,}12 & 0{,}29 \\ 0{,}22 & 0{,}17 & 0{,}17 \end{pmatrix}$$

In der ersten Iteration berechnet der Algorithmus nun $max(SIM_{authors_1,authors_2}) = 1{,}00$ und identifiziert $aname_{12}$ und $aname_{21}$ als Partner. Dies ist auch zu begrüßen, da beide Strings identisch sind. Nach dem Streichen der zweiten Zeile und der ersten Spalte bleibt die 2×2 -Matrix

$$SIM_{authors_1,authors_2} = \begin{pmatrix} 0{,}94 & 0{,}13 \\ 0{,}17 & 0{,}17 \end{pmatrix}$$

zurück. Das neue Maximum beträgt 0,94 und identifiziert $aname_{11}$ und $aname_{22}$ als Partner. Diese unterscheiden sich lediglich durch einen Buchstaben ("Abendrot", "Abend**b**rot"), wodurch auch dies gerechtfertigt erscheint. Übrig bleibt die 1×1 -Matrix

$$SIM_{authors_1,authors_2} = \begin{pmatrix} 0{,}17 \end{pmatrix}.$$

[2]Die angegebenen Werte entsprechen den auf zwei Nachkommastellen gerundeten Ergebnissen der derzeit implementierten `sim()`-Methode.

Hier sieht man nun das bereits in Abschnitt 7.1.1 angesprochene Problem bzgl. der Effektivität: Der Algorithmus würde die Namen "Gerhard Gibtsnicht" und "Wilhelm Werwohl" als Partner ausweisen, obwohl ihre Ähnlichkeit (0,17) recht gering ist und wir annehmen müssen, dass es sich um unterschiedliche Personen handelt.

Aber auch von Seiten der Effizienz ist dieser Algorithmus äußerst ungünstig: Er hat stets quadratische Laufzeit und quadratischen Platzbedarf.

7.2.2 Verbesserter Algorithmus zur Partnersuche

Um obige Missstände auszugleichen, muss der Algorithmus ein wenig verändert werden. Das genannte Effektivitätsproblem, dass Objekte zusammengefasst werden, die offensichtlich keine Partner sind, lässt sich leicht durch Einführung eines geeigneten Schwellenwertes σ beheben. Unser Algorithmus verändert sich dadurch nur geringfügig:

[1] Berechne für alle $i = 1, ..., n$ und alle $j = 1, ..., m$ den Wert $sim(o_{1i}, o_{2j})$.

[2] Falls SIM_{L_1,L_2} leer ist, so breche ab.

[3] Bestimme $max(SIM_{L_1,L_2})$, den maximalen Wert in der Matrix.

[4] Falls $(max(SIM_{L_1,L_2}) < \sigma)$, so breche ab.

[5] Bestimme die kleinsten Indizes i und j, für die $sim(o_{1i}, o_{2j}) = max(SIM_{L_1,L_2})$ gilt.

[6] Speichere das Tupel (o_{1i}, o_{2j}) als Partner.

[7] Streiche die i-te Zeile und die j-te Spalte aus SIM_{L_1,L_2}.

[8] Wiederhole ab 2.

Wie man sieht ist lediglich eine simple Abfrage, ob der maximale Ähnlichkeitswert unter dem definierten Schwellenwert σ liegt, hinzu gekommen ist (Schritt [4]). Ist dies der Fall, so bricht der Algorithmus ab; beide Listen können demnach am Ende noch Singles enthalten. Wie mit diesen zu verfahren ist, wird in Abschnitt 7.3 diskutiert.

Wenden wir uns jedoch nun dem im vorherigen Abschnitt genannten Problem der quadratischen Laufzeit zu. Diese Laufzeit kommt durch den [1]. Schritt zustande, in welchem jedes Objekt der ersten mit jedem Objekt der zweiten Liste verglichen werden muss. Gerade wenn wir aber zwei auf gleiche Weise sortierte Listen fusionieren möchten, stellt dies einen erheblichen Mehraufwand dar, da wir ansonsten die Objekte einfach der Reihe nach zusammenfassen könnten. Zwar sind wir nicht primär auf Performance bedacht und eine solche Laufzeit ist bei den Autorennamen auch weniger problematisch, da die Anzahl der Namen i.d.R. recht gering

und die Ähnlichkeit zweier Namen leicht berechenbar ist. Doch da der Algorithmus auch die erheblich komplexeren Recordlisten bearbeiten soll, von denen jedes Record für einen Artikel steht und zur Bestimmung der Ähnlichkeit teilweise erhebliche Berechnungen innerhalb der einzelnen Attribute durchgeführt werden müssen und jede Liste aus mehreren hundert Records bestehen kann (z.B. bei langen Konferenzen oder großen Zeitschriftenbänden), ist hier eine Verbesserung unabdingbar.

Beispiel 7.4 Stellen wir uns dazu zwei Listen vor, die fusioniert werden sollen. Die Records beider Listen verfügen sowohl über die korrekten Angaben der Seitenzahlen, nach welchen sie auch sortiert sind, als auch über jeweils identische DOIs, anhand deren man bei Gleichheit sofort eine Ähnlichkeit von 1,00 für beide Records bestimmen kann. Eine nach dem naiven Algorithmus erstellte Matrix sähe nun wie folgt aus:

$$SIM_{L_1,L_2} = \begin{pmatrix} 1{,}00 & s_{12} & \cdots & s_{1n} \\ s_{21} & 1{,}00 & \cdots & s_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ s_{n1} & s_{n2} & \cdots & 1{,}00 \end{pmatrix}, \text{ mit } 0 \leq s_{ij} < 1 \text{ für alle } 1 \leq i \leq n, 1 \leq j \leq n.$$

Während die Berechnung der Ähnlichkeit von 1,00 anhand der DOIs recht schnell geschieht, müssen alle übrigen Werte s_{ij}, die eine kleinere Ähnlichkeit ergeben, da unter der Annahme, dass es sich innerhalb jeder Liste stets um unterschiedliche Artikel handelt, keine zwei DOIs gleich sein können, evtl. durch komplizierte Berechnungen ermittelt werden – und dies völlig ohne Nutzen. Wir werden den Algorithmus also dahingehend abändern, dass bei Erhalt eines Ähnlichkeitswertes von 1,00 (was ja bedeutet, dass beide Objekte mit einer Wahrscheinlichkeit von 1 Partner sind) keine weiteren Berechnungen mit den entsprechenden Objekten mehr durchgeführt werden. Wir verfahren an dieser Stelle also bereits bei der Berechnung der Matrix entsprechend und streichen sofort die i-te Spalte und j-te Zeile, wenn $sim(o_{1i}, o_{2j}) = 1{,}00$ gilt.

In Beispiel 7.4 würde also gar keine Matrix aufgebaut, da wir im [1]. Schritt des Algorithmus zunächst $sim(o_{11}, o_{21})$ berechnen würden. Dieser Wert ist 1,00, und so würden wir die erste Zeile und erste Spalte streichen und das Tupel (o_{1i}, o_{2j}) als Partner ausweisen. Der nächste Vergleich würde demnach – wegen der Streichung der ersten Werte – mit o_{12} und o_{22} fortfahren, deren Ähnlichkeit ebenfalls 1,00 beträgt. Nach dem letzten Vergleich ($sim(o_{1n}, o_{2n})$) wären alle Partner bestimmt, die Matrix SIM_{L_1,L_2} wäre leer und der obige Algorithmus würde sofort in Zeile [2] abbrechen und hätte sämtliche Zuordnungen in linearer Laufzeit korrekt getroffen.

Im Normalfall wird zwar keine lineare Laufzeit erreicht, doch lässt sich mittels dieser Verbesserung in vielen Fällen eine Reduzierung der Zeitkomplexität erreichen. In Beispiel 7.3 des vorangehenden Abschnittes würde die Zuordnung der identischen Namen “Birgit Blumenkohl” bereits im 1. Schritt erfolgen – der Vergleich mit der Nummer `(4)` – und somit die weiteren Berechnungen `(5)`, `(6)` und `(7)` ersparen, was immerhin einer Reduzierung auf $\frac{2}{3}$ der ursprünglichen Berechnungen bedeutet. Eine quadratische Laufzeit wäre damit nicht mehr wie zuvor in jedem, sondern lediglich noch im schlimmsten Fall zu erwarten.

Abschließend soll noch das Problem des stets benötigten quadratischen Platzbedarfs angesprochen werden. Dieser ist implementierungsabhängig, und es ist möglich, hier ebenfalls erheblich performanter vorzugehen. In einer naiven Implementierung würde obige Matrix in Form eines zweidimensionalen Arrays implementiert werden, welches in jedem Feld den entsprechenden Ähnlichkeitswert aufnimmt. In obigem Beispiel 7.2 sah diese wie folgt aussehen:

$$SIM_{authors_1,authors_2} = \begin{pmatrix} 0{,}18 & 0{,}94 & 0{,}13 \\ 1{,}00 & 0{,}12 & 0{,}29 \\ 0{,}22 & 0{,}17 & 0{,}17 \end{pmatrix}$$

Bei Wahl eines geeigneten Schwellenwertes σ sehen wir jedoch, dass wir viele der Werte dieser Matrix niemals benötigen werden. Setzen wir beispielsweise einen Wert von $\sigma = 0{,}60$ voraus, so lässt sich das Array dergestalt reduzieren, dass wir alle Werte, die kleiner als σ sind, sofort streichen können:

$$\begin{bmatrix} -- & 0{,}94 & -- \\ 1{,}00 & -- & -- \\ -- & -- & -- \end{bmatrix}$$

Ungeachtet der Tatsache, dass nach obiger Reduzierung der Zeitkomplexität die zweite Zeile und die erste Spalte ohnehin gestrichen würden, sehen wir, dass es gar nicht notwendig ist, ein großes Array aufzubauen, da der Großteil der Werte völlig uninteressant ist. Gerade bei großen Recordlisten (bei 100 Artikeln würde das entsprechende Array bereits 10.000 Felder beinhalten, deren Werte teilweise überhaupt nicht berechnet würden) wird sich diese Reduzierung des Platzbedarfs äußerst positiv auswirken: Bei geschickter Wahl von σ ist es offensichtlich äußerst unwahrscheinlich, dass sich, gerade in langen Listen, alle bzw. viele Objekte so sehr gleichen, dass deren Werte allesamt in die Matrix aufgenommen werden müssen.

Bei der Implementierung in Java wurden zur Speicherung der Ähnlichkeitswerte HashMaps verwendet, die eine entsprechende Zuordnung zu Zeile und Spalte zulassen. Details der Implementierung sind den ausführlichen Kommentaren im Quellcode zu entnehmen.

Abschließend lässt sich feststellen, dass der hier vorgestellte verbesserte Algorithmus

- in der Lage ist, stets alle Partner korrekt zu identifizieren, sofern die $sim()$-Funktion korrekte Ergebnisse liefert. Singles werden korrekt ignoriert.
- im best-case lineare, im worst-case jedoch quadratische Laufzeit besitzt.
- in der Regel mit wenigen Matrixfeldern auskommt und somit auf keinen Fall quadratischen Platzbedarf hat.

7.3 Handhabung der Singles

Als Ergebnis des in Abschnitt 7.2 vorgestellten Algorithmus' erhalten wir eine Ergebnisliste, in welcher alle identifizierten Partnertupel enthalten sind, welche anschließend fusioniert werden

sollen. Alle Objekte, die nicht als Teil eines Paares identifiziert werden konnten, bleiben als Singles zurück.

Die einfachste Möglichkeit, jene Singles zu behandeln, ist es, diese dem Ergebnis unbearbeitet hinzuzufügen. Betrachten wir hierzu noch einmal das im vergangenen Abschnitt beschriebene Beispiel 7.3:

$authors_1$ = (“Alfons Abendrot”, “Birgit Blumenkohl”, “Gerhard Gibtsnicht”),
$authors_2$ = (“Birgit Blumenkohl”, “Alfons Abendbrot”, “Wilhelm Werwohl”).

Der Partnersuche-Algorithmus liefert ein Ergebnis der Form

$partners$ = ((“Alfons Abendrot”, “Alfons Abendbrot”), (“Birgit Blumenkohl”, “Birgit Blumenkohl”)).

Zusätzlich erhalten wir zwei Listen mit Singles, die wie folgt aussehen:

$singles_1$ = (“Gerhard Gibtsnicht”),
$singles_2$ = (“Wilhelm Werwohl”).

Werden nun die Partner fusioniert und die Singles der Liste hinzugefügt, so ergibt sich – unter der Annahme, dass [“Alfons Abendrot” $\bowtie$ “Alfons Abendbrot” $\longrightarrow$ “Alfons Abendrot”] gelte, die Ergebnisliste

$authors_{result}$ = (“Alfons Abendrot”, “Birgit Blumenkohl”, “Gerhard Gibtsnicht”,
“Wilhelm Werwohl”).

Wir haben also durch den Fusionsvorgang zusätzliche Informationen in Form weiterer, voneinander verschiedener (Co-)Autoren erhalten. Dennoch besteht der dringende Verdacht, dass in diesem konstruierten Beispiel, gerade weil je ein Name nur in einer Quelle auftrat, eine manuelle Überprüfung von Nöten ist. Die Software wird einen solchen Fall anzeigen und – je nach Konfiguration – die Singles aus einer oder beiden Listen übernehmen.

7.4 Eliminierung von Dubletten

Doch ganz so einfach wie es scheint, ist die Handhabung der Singles nicht immer. Betrachten wir hierzu ein ‘reales’ Beispiel:

Beispiel 7.5
Abbildung 7.1 zeigt einen Artikel der Konferenz “Chinacom 2007” innerhalb der EUDL.[3] Wie

[3] `http://eudl.eu/?eudlQuery=CHINACOM%202007&type=Papers&page=21`, Datum des Screenshots: 28.06.2009

man unschwer erkennen kann, werden dort zwei der drei Autorennamen doppelt angegeben, einmal in korrekter Reihenfolge von Vor- und Nachname, einmal in umgekehrter. Die korrekten Namen, die mittels der Namenssuche von DBLP leicht aufgefunden werden können, lauten: "Dirk Hetzer", "Ilka Miloucheva" und "Karl Jonas". Diese entsprechen im genannten Beispiel dem ersten, vierten und fünften Namen. Beim zweiten und dritten Namen handelt es sich um *Dubletten*.[4]

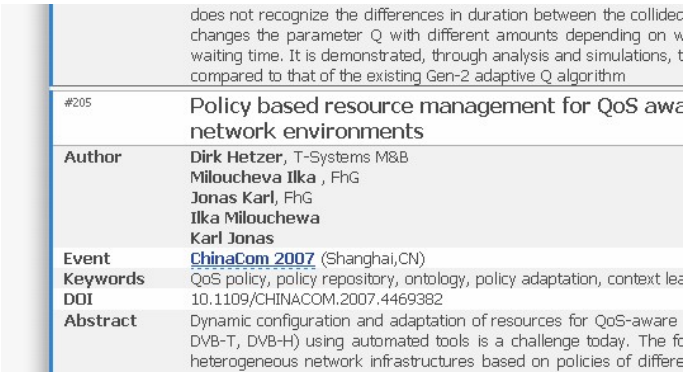


Abb. 7.1: Beispiel fehlerhafter Namen in der EUDL.
Quelle: `http://eudl.eu/?eudlQuery=CHINACOM%202007&type=8&page=21`

Stellen wir uns nun vor, wir möchten die Namen zweier Quellen fusionieren, von welchen die eine die Namen in korrekter Reihenfolge und Anzahl, allerdings mit lediglich abgekürzten Vornamen besitzt, die andere jedoch die im Bild gezeigten Dubletten enthält. Unsere Listen lauten demnach

$authors_1$ = ("D. Hetzer", "I. Miloucheva", "K. Jonas") und
$authors_2$ = ("Dirk Hetzer", "Miloucheva Ilka", "Jonas Karl", "Ilka Milouchewa", "Karl Jonas").

Wir sehen an diesem Beispiel nicht nur, dass die $sim()$-Funktion auch und besonders die Reihenfolge der Namen in ihr Ergebnis einfließen lassen sollte, sondern auch, welches Problem nun auftaucht. Unsere Ähnlichkeitsmatrix könnte daher – in obiger Schreibweise als Array, in welchem nur relevante Werte $> \sigma = 0{,}60$ auftauchen, wie folgt aussehen:

$$\begin{bmatrix} 0{,}97 & -- & -- & -- & -- \\ -- & 0{,}88 & -- & 0{,}75 & -- \\ -- & -- & 0{,}88 & -- & 0{,}90 \end{bmatrix}$$

Nach Beendigung des Partnersuche-Algorithmus' und anschließender Fusionierung der Namen erhalten wir ein vorläufiges Ergebnis von

[4] Die Begriffe *Duplikat* und *Dublette* werden hier bewusst verwandt, um unterschiedliche Sachverhalte darzustellen. Als 'Dublette' werden nur jene Singles bezeichnet, die eliminiert werden sollen, da sie ein reales Objekt benennen, für welches wir bereits ein Paar gefunden haben. Als 'Duplikat' im Sinne des Integrationmodells werden dagegen alle Daten bezeichnet, die den gleichen realen Sachverhalt darstellen; in unserem Falle also sowohl die Paare als auch die Dubletten.

$result$ = (“Dirk Hetzer”, “Ilka Miloucheva”, “Karl Jonas”),

während die Single-Listen wie folgt aussehen:

$singles_1 = null$,
$singles_2$ = (“Jonas Karl”, “Ilka Milouchewa”).

Wir dürfen nun also die übrigen Singles nicht einfach dem Ergebnis hinzufügen, sondern müssen einen weiteren Algorithmus anwenden, welcher jeden Single ($single_j$) mit allen Ergebnisobjekten – d.h. fusionierten Partnern – (o_i) vergleicht. Gilt hier $sim(o_i, single_j) \geq \sigma$ für ein beliebiges i, so wird $single_j$ verworfen, da es sich wahrscheinlich um ein Duplikat handelt.

Dieser Schritt ist im Integrationsmodell nach Bleiholder und Naumann nicht notwendig, da dort davon ausgegangen wird, dass in Phase 2 *alle* Duplikate – und nicht wie in unserem Fall nur jeweils zwei – gefunden und anschließend fusioniert wurden; der Fall tritt bei uns also nur deshalb auf, weil wir stets Paare von Duplikaten fordern und keinen dritten Wert ins Fusionsergebnis einfließen lassen. Dieses Vorgehen ist auch notwendig, wie Beispiel 7.6 zeigt:

Beispiel 7.6 Betrachten wir den Artikel “Design and Implementation of a Distributed System Level Evaluation Platform for Mobile WiMAX” oder Konferenz CHINACOM 2007, der sowohl in der EUDL als auch bei IEEE Xplore zu finden ist.[5] Nehmen wir Xplore als primäre, die EUDL als sekundäre Quelle, so ergeben sich die beiden folgenden Autorenlisten

$authors_1$ = (“Jian Zhang”, “Yu Zhao”, “Jian Li”, “Jian Zhang”, “Xiaodong Zhang”),
$authors_2$ = (“Jian Zhang”, “Jian Li”, “Yu Zhao”, “Jian Zhang”, “Yu Zhao”, “Jian Li”, “Xiaodong Zhang”).

Wir erkennen zum einen das typische Problem der EUDL, dass gleiche Autorennamen mehrfach auftreten. So sind die Namen “Yu Zhao” und “Jian Li” in $authors_2$ zweimal, in $authors_1$ jedoch nur einmal zu finden und müssen somit als Dubletten eliminiert werden. Der Name “Jian Zhang” dagegen tritt in beiden Quellen zweimal auf, jeweils an erster und vierter Position. Es ist daher zu vermuten, dass es sich hierbei um zwei *verschiedene* Personen handelt, die den gleichen Namen tragen. Würden alle Duplikate eines Namens ins Ergebnis einfließen, so würde das zweite Vorkommen dieses Namens irrtümlicherweise gelöscht werden, doch da beide Namen doppelt auftreten, werden sie von unserem Algorithmus auch zweimal als unterschiedliche Partner identifiziert. Wir erhalten somit das gewünschte Ergebnis

$authors_{result}$ = (“Jian Zhang”, “Yu Zhao”, “Jian Li”, “Jian Zhang”, “Xiaodong Zhang”),

[5] EUDL: `http://eudl.eu/?eudlQuery=CHINACOM%202007&type=8&page=5`,
Xplore: `http://ieeexplore.ieee.org/search/wrapper.jsp?arnumber=4469319`;
In der Autorenliste von Xplore liegen die Vornamen zweier Autoren – vermutlich bedingt durch OCR-Fehler – in inkorrekter Schreibweise vor (Jaii statt Jian, Xiaoclong statt Xiaodong). Diese Tatsache wollen wir im Beispiel vernachlässigen und die Namen daher korrekt schreiben; wir werden genau dieses Problem bei der Fusion einzelner Namen wieder aufgreifen.

welches dem ursprünglichen $authors_1$ entspricht.

Abschließend noch ein paar Worte zur Reihenfolge: Da die primäre Quelle eine höhere Priorität besitzt, soll die Reihenfolge der Objekte sich an der Reihenfolge, in welchen sie in der ersten Liste auftreten, orientieren. Dies mag bei Recordlisten unerheblich sein, da diese ohnehin anschließend erneut nach Seitenzahlen sortiert werden, doch bei Autorennamen spielt die Reihenfolge in jedem Fall eine Rolle. Wir werden uns daher innerhalb der Software zu jedem Partner und jedem Single der ersten Liste die Position merken und die Fusionsergebnisse bzw. die Singles, bei welchen es sich nicht um Dubletten handelt, im Ergebnis an eben jene Position setzen. Möglicherweise entstehen auf diese Weise Lücken durch entfernte Dubletten, so dass am Ende eine erneute Nummerierung nötig werden kann. Singles aus der zweiten Quelle werden dagegen stets ans Ende der Ergebnisliste angehangen, ganz gleich, welche Position sie innerhalb ihrer Liste hatten.

Kapitel 8

Fusion simpler Objekte

Der im vorherigen Kapitel vorgestellte Algorithmus zur Partnersuche transformiert zwei komplexe Objekte – die wir bekanntermaßen auch als zwei Tupel gleichartiger Objekte auffassen – in ein einzelnes Tupel geordneter Paare. So werden beispielsweise zwei Autorenlisten $authors_1$ und $authors_2$ mit

$authors_1.value = (aname_{11}, aname_{12}, \cdots, aname_{1n})$ und
$authors_2.value = (aname_{21}, aname_{22}, \cdots, aname_{2m})$

in ein einziges Tupel $authors_{result}$ mit

$$authors_{result}.value = ((aname_{1i_1}, aname_{2j_1}), (aname_{1i_2}, aname_{2j_2}), \cdots, (aname_{1i_p}, aname_{2j_p}))$$

verwandelt, wobei die Indizes i_1 bis i_p natürliche Zahlen $\leq n$, j_1 bis j_p natürliche Zahlen $\leq m$ andeuten sollen. Zudem entstehen zwei Tupel $singles_1$ und $singles_2$, deren Elemente jedoch keiner weiteren Fusion bedürfen, sondern je nachdem, ob es sich um Dubletten handelt, verworfen oder zum Ergebnis hinzugefügt werden. Die Paare einer solchen Ergebnisliste müssen nun also in geeigneter Weise fusioniert werden, ebenso wie sämtliche simplen Objekte, bei denen eine Identifizierung von Partnern ohnehin nicht nötig ist.

Dieses Kapitel beschäftigt sich nun mit eben jener Fusion simpler Objekte und entspricht der 3. Phase des Integrationsmodells nach Bleiholder und Naumann, der 'Datenfusion' im engeren Sinne. Zu den simplen Objekten zählen nach der Definition in Kapitel 2.4 sämtliche Attribute (mit Ausnahme von *authors*), aber auch die Records. Wir werden uns daher zunächst in Abschnitt 8.1 mit der Fusion einzelner Attribute beschäftigen, bevor wir uns in Abschnitt 8.2 der Fusion ganzer Records widmen werden.

Eine Sonderstellung diesbezüglich nehmen die Autorennamen (*aname*) ein. Bei diesen handelt es sich um komplexe Objekte, welche sich aus einzelnen Namensteilen $(n_1, n_2, \cdots, n_p)$ zusammensetzen. Wir werden die Fusion zweier Autorennamen dennoch in diesem Kapitel behandeln (Abschnitt 8.3), da wir niemals zwei einzelnen Namensteile (bei welchen es sich um simple Objekte handelt) fusionieren werden, sondern Namen stets als Ganzes betrachten müssen. Die im vorherigen Kapitel vorgestellten Algorithmen können hier nur bedingt angewandt werden.

8.1 Fusion simpler Attribute

Bei allen Attributen unseres Datenmodells, mit Ausnahme des *authors*-Attributes, handelt es sich um simple Objekte, d.h. solche Objekte, die nur aus einem einzigen Wert bestehen. In diesem Abschnitt werden wir uns mit deren Fusion und dem jeweils zu errechnenden Ergebnis beschäftigen.[1] Dabei werden wir auch die Semantik des jeweiligen Datensatzes betrachten müssen, denn wie bereits erwähnt macht es offensichtlich einen großen Unterschied, ob wir zwei Personennamen, zwei Seitenangaben oder zwei URLs miteinander fusionieren.

8.1.1 Allgemeine Beobachtungen

Zunächst wollen wir einige Aussagen formulieren, die für sämtliche simple Attribute von Gültigkeit sind.

Nicht fusionierbare Attribute Da wir nach Gleichung (6.1) auf Seite 112 wissen, dass stets nur solche Objekte fusioniert werden können, die den gleichen Typ haben, wollen wir dies für die Attribute im Folgenden stets implizit voraussetzen. Wir werden jedoch sehen, dass es auch bei gleichem Typ Fälle geben wird, in welchen eine Entscheidung, wie das Ergebnis aussehen soll, nicht durch einen Algorithmus getroffen werden kann. Auch dann werden wir von Daten sprechen, die nicht fusioniert werden können.

Leere Attribute Da das Ziel der Fusion stets die Gewinnung zusätzlicher Informationen ist, definieren wir folgende Spezialfälle, die für alle Typen von Attributen gelten sollen:

$$\begin{aligned} a_1 \bowtie a_2 &\longrightarrow a_1, && \text{falls} \quad a_1.value \neq null \quad \text{und} \quad a_2.value = null \\ a_1 \bowtie a_2 &\longrightarrow a_2, && \text{falls} \quad a_1.value = null \quad \text{und} \quad a_2.value \neq null \\ a_1 \bowtie a_2 &\longrightarrow null, && \text{falls} \quad a_1.value = a_2.value = null \end{aligned} \tag{8.1}$$

Attribute, die einen Wert besitzen, werden also stets leeren Attributen vorgezogen. Dies erscheint auch als völlig sinnvoll, da ein leeres Attribut in jedem Fall weniger Informationsgehalt trägt als eines, das einen Wert besitzt. Wir verfahren also stets nach der in Kapitel 5.3.3 vorgestellten Strategie Take the information, um Konflikte der Kategorie 'Unsicherheit' zu bearbeiten.

Attribute mit identischen Werten Es gilt:

$$a_1.value = a_2.value \quad \Rightarrow \quad a_1 \bowtie a_2 \longrightarrow a_1. \tag{8.2}$$

[1] Ist also innerhalb dieses Kapitels von Attributen die Rede, so wollen wir stets von simplen Attributen ausgehen; das *authors*-Attribut wird gemäß den im vorherigen Kapitel vorgestellten Algorithmen in Paare einzelner Autorennamen zerlegt, deren Fusion wir uns in Abschnitt 8.3 annehmen werden.

Da die Attribute identisch sind – $a_1.type = a_2.type$ gilt nach Voraussetzung ja ohnehin – spielt es keine Rolle, ob wir a_1 oder a_2 als Ergebnis definieren. In diesem Fall tritt kein Konflikt auf, und es ist demnach auch nicht notwendig, einen solchen aufzulösen.

Haben beide Attribute einen Wert, der jedoch nicht bei beiden gleich ist, so hängt die Frage, ob diese fusioniert werden können, vom jeweiligen Typ der Attribute ab. Wir werden uns nun also eingehend mit den unterschiedlichen Typen auseinander setzen und jeweils davon ausgehen, dass keines der Attribute den Wert *null* hat und ihre Werte auch nicht identisch sind.

8.1.2 Fusion zweier Titel

Die Fusion zweier Titel stellt eine simple Selektion dar. Es gilt stets eine der drei folgenden Aussagen:

$$title_1 \bowtie title_2 \longrightarrow title_1 \tag{8.3a}$$
$$title_1 \bowtie title_2 \longrightarrow title_2 \tag{8.3b}$$
$$title_1 \not\bowtie title_2 \tag{8.3c}$$

Ein Titel ist prinzipiell nichts anderes als eine beliebige Kombination von Zeichen. Da wir nicht in der Lage sind, die Semantik einer solchen Zeichenketten in einer formalen Definition zu erfassen, kann auch nicht mittels eines Algorithmus entschieden werden, welcher von zwei gegebenen, unterschiedlichen Titeln der bessere ist. Wir sollten daher für Titel-Attribute stets die Fusions-Modi `1` oder `ignore` wählen. Der Titel dient vielmehr zur Identifikation von Partnern (siehe Kapitel 7.2) und wird in allen derzeit benötigten Praxisszenarien (siehe Kapitel 6.1) unverändert belassen.

Dennoch können wir ein paar Regeln angeben, mit deren Hilfe wir in der Mehrzahl der Fälle den ‘besseren’ Titel, d.h. jenen, der mehr Informationsgehalt zu besitzen scheint, herausfinden können. Diesen Titel werden wir dann als Ergebnis der Fusion zurück liefern und bei der Umsetzung in die Software eine Meldung ausgeben, falls dieser sich vom gewählten Titel unterscheidet.

Zur Auswahl des wahrscheinlich informationsreicheren Titels formulieren wir die folgenden Regeln:

1. “Je länger, desto besser”: Haben die Titel unterschiedliche Länge, so vermuten wir, dass der längere von beiden mehr Informationen enthält und ziehen diesen vor.

2. “Klein ist besser als groß”: Wir wir bereits wissen, geben einige Verlage die Titel ihrer Artikel komplett in GROSSBUCHSTABEN an. Der Wrapper versucht zwar, diesen Missstand zu beheben, doch gelingt es ihm nicht immer, die korrekte Schreibweise der Worte zu ermitteln (vgl. Kapitel 4.3.1, “*normalisiere spezielle Daten*”). Andere Titel wiederum sind derart geschrieben, dass jedes Wort mit einem Großbuchstaben beginnt, was

ebenfalls nicht der korrekten englischen (deutschen, französischen,...) Schreibweise entspricht. Wir wollen daher Strings vorziehen, die eine höhere Anzahl an Kleinbuchstaben aufweisen.

3. "Höhere Zeichencodes bieten mehr Information": Enthält einer der zu vergleichenden Titel Zeichen, die nicht im ASCII- oder gar Latin-1-Zeichensatz enthalten sind, so scheint dieser speziellere Informationen (z.B. Buchstaben mit besonderen Akzenten, griechische Buchstaben, mathematische Sonderzeichen etc.) zu enthalten und wird daher vorgezogen.

Kritik Wie bereits erwähnt sind die oben definierten Regeln äußerst *instabil*, d.h. es wird in der Praxis immer wieder Fälle geben, in denen diese versagen. Nehmen wir beispielsweise an, unsere Titel unterschieden sich lediglich in den Worten "algorithm" und "algortithm", d.h. im zweiten Titel habe sich ganz offensichtlich ein Tippfehler in Form eines überzähligen Buchstabens eingeschlichen. Dennoch würde dieser falsche Titel nach obiger Regel 1 als der bessere angesehen. Die Selektion eines Titels dient daher nur dem Ziel, einen Vorschlag zu unterbreiten – der im Falle des o.g. Beispiels bei einer manuellen Überarbeitung sofort abgelehnt würde.

8.1.3 Fusion zweier Seitenangaben

Jedes *pages*-Attribut enthält stets einen Wert in einem der folgenden Formate:

(1) 0-

(2) x

(3) x-

(4) x-y

Hierbei sind x und y Platzhalter für natürliche Zahlen > 0 oder römische Zahlen.[2] Die Verwendung der Zahlensymbole muss jedoch innerhalb eines Attributes einheitlich sein, zudem muss stets die Bedingung $x < y$ gelten. Erlaubt sind demnach beispielsweise die Werte "1-10", "20", "50-", "ii-v" oder "0-", verboten sind jedoch Angaben wie "5-5", "10-9", "x-ix", "2-vi" oder "0-5".

Da der Nullwert '0-' dem Fehlen des Attributes entspricht, wird im Fall, dass eines der Seitenattribute den Wert "0-" hat, auch analog zur am Anfang dieses Kapitels vorgestellten Bearbeitung eines *null* -Wertes verfahren (Gleichung (8.1)). Deshalb gehen wir davon aus, dass

[2]Römische Zahlen treten oftmals in Einleitungen oder Anhängen auf. Wurden die entsprechenden Artikel mit einem Titel und einer Autorenangabe versehen, so möchten wir sie ebenfalls in DBLP erfassen. Daher sollen alle Teile der Software mit römischen Zahlzeichen umgehen und diese korrekt berechnen können.

die beiden zu fusionierenden Attribute $pages_1$ und $pages_2$ in einem der Formate (2)-(4) vorliegen, jedoch voneinander verschieden sind – denn ansonsten würde ohnehin Gleichung (8.2) gelten. Mit x_1 und y_1 bzw. x_2 und y_2 wollen wir analog zur obigen Darstellung jeweils die erste und (soweit vorhanden) zweite Zahl des Wertes benennen, mit $pages_i.format$ bezeichnen wir das Format der Seitenangaben gemäß obiger Auflistung in Gestalt einer natürlichen Zahl – wenn also beispielsweise $pages_1$ den Wert "10-18" hat, so gilt: $pages_1.format = 4$, $x_1 = 10$ und $y_1 = 18$.

Dann können wir folgende Aussagen bzgl. der Fusion von $pages_1$ und $pages_2$ treffen:

$$x_1 \neq x_2 \quad \Rightarrow \quad pages_1 \not\bowtie pages_2 \tag{8.4a}$$

$$x_1 = x_2 \quad \text{und} \quad pages_1.format = pages_2.format \quad \Rightarrow \quad pages_1 \not\bowtie pages_2 \tag{8.4b}$$

$$x_1 = x_2 \quad \text{und} \quad pages_1.format \neq pages_2.format$$
$$\Rightarrow \quad pages_1 \bowtie pages_2 \longrightarrow \begin{cases} pages_1, & \text{falls} \quad pages_1.format > pages_2.format \\ pages_2, & \text{sonst} \end{cases} \tag{8.4c}$$

Etwas weniger formal ausgedrückt bedeutet dies:

- Stimmen die Angaben der Seite, auf welcher ein Artikel *beginnt*, nicht miteinander überein, so können die Seitenangaben nicht fusioniert werden (8.4a).

- Stimmen die x-Werte jedoch überein und sind beide Angaben von gleichem Format, so wissen wir implizit, dass beide Seitenangaben im Format '*x-y*' vorliegen müssen (da sie ansonsten ja identisch wären, was wir bereits ausgeschlossen haben) und sich daher die Angaben der Seiten, auf welchen der Artikel *endet*, voneinander unterscheiden. Auch hier sind wir nicht in der Lage festzustellen, welche Angabe korrekt ist, so dass die Attribute nicht fusioniert werden können (8.4b).

- In jedem anderen Fall enthält eines der Seitenattribute zusätzliche Informationen, die jedoch nicht widersprüchlich zu denen des anderen Attributes sind – beispielsweise $pages_1.value$ ="100" und $pages_2.value$ ="100-120". Wir wählen dann dasjenige mit der größeren Formatnummer, da die Formate bewusst in der Reihenfolge aufsteigenden Informationsgehalts angeordnet wurden (8.4c). In unserem Beispiel gilt: $pages_1.format = 2$ und $pages_2.format = 4$, woraus $pages_1 \bowtie pages_2 \longrightarrow pages_2$ resultiert, was offensichtlich einen Gewinn an Informationen darstellt – selbstverständlich immer nur unter der Voraussetzung, dass beide Werte korrekt sind.

8.1.4 Fusion zweier EE-Attribute

Die Verbesserung der Angaben eines Links zu einer 'Electronic Edition' des betreffenden Artikels ist eine der Hauptaufgaben der Merge-Software. Wie in Szenario F-2'$_{\text{LNCS}}$ in Kapitel 6.1.4 beschrieben, enthalten tausende von Bänden der LNCS-Serie lediglich normale URL-Angaben, obwohl bei Springerlink, der DL des Verlags, mittlerweile zu den meisten Artikeln

DOIs verfügbar sind. Wir möchten die Software nutzen, um eben jene URLs gegen die permanenten DOIs auszutauschen.

Die Fusion zweier *ee*-Attribute ist daher recht einfach. Wir definieren uns hierzu lediglich eine einfache bool'sche Funktion:

$$doi(ee) = \begin{cases} true, & \text{falls } ee.value \text{ ein DOI ist} \\ false, & \text{sonst} \end{cases}$$

In der folgenden Definition werden wir das Symbol $\wedge$ gemäß seiner aus der bool'schen Logik bekannten Bedeutung eines 'and'-Operators nutzen. Die Negation werden wir durch Voranstellen des Operators '$\neg$' ausdrücken; so liefert also $\neg doi(ee)$ den Wert *true*, falls es sich bei *ee.value* gerade *nicht* um einen DOI handelt.

Damit sind wir in der Lage, die Fusion zweier *ee*-Attribute zu definieren:

$$doi(ee_1) = doi(ee_2) \quad \Rightarrow \quad ee_1 \not\bowtie ee_2 \tag{8.5a}$$

$$doi(ee_1) \wedge \neg doi(ee_2) \quad \Rightarrow \quad ee_1 \bowtie ee_2 \longrightarrow ee_1 \tag{8.5b}$$

$$\neg doi(ee_1) \wedge doi(ee_2) \quad \Rightarrow \quad ee_1 \bowtie ee_2 \longrightarrow ee_2 \tag{8.5c}$$

In (8.5a) liefert die bool'sche Funktion $doi()$, angewandt auf jedes der beiden Attribute, den gleichen Wert. Dies bedeutet, dass beide Attributwerte entweder DOIs oder normale URLs sind. In beiden Fällen kann nicht entschieden werden, welcher Wert der bessere ist.[3]

Die Aussagen (8.5b) und (8.5c) dagegen sind symmetrisch: Ist der Wert des einen Attributes ein DOI, der des anderen ein einfacher URL, so ziehen wir den DOI in jedem Falle vor und definieren diesen als Ergebnis unserer Fusion.

8.1.5 Fusion zweier Zwischenüberschriften

Ein Ziel der Fusion ist es, Zwischenüberschriften zu ergänzen. Hier tritt jedoch das gleiche Problem wie bei den Titeln auf, dass die Software nicht in der Lage sein wird zu entscheiden, welcher von zwei Strings *besser* ist als der andere. Da, wie in Kapitel 2.3.6 beschrieben, jedoch bereits das Vorhandensein von Zwischenüberschriften eine Steigerung der Datenqualität ausmacht, möchten wir uns damit zufrieden geben, wenn eine der Quellen über eine solche verfügt. Daher werden wir die *section*-, *subsection* und *subsubsection*-Attribute, sowie das *header*-Attribut im Falle von Konferenzen, nur dann fusionieren, wenn in nur genau einer der Quellen ein Wert vorhanden ist. Ansonsten definieren wir, dass die Objekte nicht fusioniert werden können.

[3] Prinzipiell müsste sogar davon ausgegangen werden, dass zwei Artikel mit verschiedenen DOIs auch in jedem Fall verschieden sind. Diese Schlussfolgerung werden wir jedoch außer Acht lassen, da es vorkommen könnte, dass vormals falsch eingetragene DOIs korrigiert werden sollen und es sich tatsächlich doch um gleiche Artikel handelt.

Seien a_1 und a_2 Attribute des gleichen Typs mit

$$a_1.type \in \{header, section, subsection, subsubsection\}.$$

Dann gilt:

$$a_1.value \neq null \quad \text{und} \quad a_2.value \neq null \quad \Rightarrow \quad a_1 \not\approx a_2 \tag{8.6}$$

Falls einer der beiden Werte ungleich *null* ist, so gilt ohnehin Gleichung (8.1). Ist eine Fusion möglich, so liegt also stets eine Selektion in Form einer Ergänzung eines leeren Wertes vor.

8.1.6 Fusion der übrigen Attribute

Für alle übrigen, bisher nicht betrachteten simplen Attribute existiert keine besondere Fusionsvorschrift, d.h. wir benötigen keine Überprüfung dieser Attribute auf Unterschiede. Dies liegt darin begründet, dass wir stets davon ausgehen, bibliographische Daten der gleichen Konferenz bzw. des gleichen Journalvolumes/-issues miteinander zu fusionieren. Es ist daher nicht notwendig, eine Fusion der Attribute *volume*, *number*, *year* und *month* zu definieren, da wir diese als gleich voraussetzen. Bei der *articleNr* handelt es sich um ein nur in Ausnahmefällen (z.B. bei Daten des BMC, vgl. Kapitel 3.2.3) verwendetes Attribut, dessen gesonderte Betrachtung nicht lohnenswert ist. Das *key*-Attribut dagegen werden wir nur in der primären Quelle zulassen, und zwar genau dann, wenn wir gemäß Szenario F-2 (bzw. F-2' / F-2'$_{\text{LNCS}}$) bestehende DBLP-Records verbessern möchten.

8.2 Fusion zweier Records

Im vorangegangenen Abschnitt haben wir ausführlich studiert, wie die Fusion aller für unsere Zwecke relevanter Attribute abläuft, welches Ergebnis wir erwarten und wann zwei Attribute nicht fusioniert werden können. Dieses Wissen nutzen wir nun aus, um zwei Records, welche durch eben jene Attribute definiert sind, miteinander zu fusionieren.

Da sich Records durch deren Attribute kennzeichnen, liegt es nahe, die Fusion zweier Records durch die Fusion all ihrer Attribute zu definieren, und zwei Records genau dann $R_1 \not\approx R_2$ zu definieren, wenn mindestens eines ihrer Attribute nicht fusioniert werden kann. Diese Definition hätte allerdings erhebliche Nachteile in der Praxis, da wir wissen, dass bei einigen Attributen (beispielsweise den Titeln) bereits ein kleiner Unterschied ausreichen kann, um eine Fusion unmöglich zu machen.

In der Praxis erscheint uns ein Ergebnis der Form $R_1 \not\approx R_2$ jedoch als völlig unzureichend. Wir möchten nicht, dass die Software in 9 von 10 Fällen mit der Meldung "`not mergeable`" abbricht, nur weil zwei Attribute der Records nicht fusioniert werden können. Dies würde keine Erleichterung der Arbeit bedeuten, sondern diese sogar noch erschweren, da jeder derartige Fall manuell überprüft werden müsste. Daher werden wir fordern, dass die Software im Falle, dass

zwei Attribute nicht fusioniert werden können, eine eigenständige Auswahl trifft und somit die Fusion zweier Records in jedem Fall ermöglicht – der theoretische durchaus denkbare Fall '$R_1 \not\approx R_2$' wird in der Praxis niemals eintreffen.

Um dies zu erreichen, formulieren wir folgende allgemeingültige Regel:

- Tritt bei der Fusion zweier Records R_1 und R_2 der Fall ein, dass für zwei Attribute a_1 und a_2 (mit a_1 aus R_1, a_2 aus R_2) des gleichen Typs die Aussage '$a_1 \not\approx a_2$' gilt, so gelte stets $a_{result} = a_1$.

An dieser Stelle tritt also zum bisher ersten Mal eine Form der *Asymmetrie* bei der Fusion auf: Das erste Record wird bei der Fusion im Zweifelsfall stets bevorzugt behandelt. Dies entspricht der in Kapitel 5.3.3 vorgestellten Konfliktlösungsstrategie TRUST YOUR FRIENDS: Die Daten der primären Quelle werden stets als 'vertrauenswürdiger' erachtet als jene der sekundären.

Diese Form der Asymmetrie ist auch in jedem Falle erstrebenswert, denn auf diese Weise ist es möglich, einem bereits bestehenden Datensatz gezielt mit Hilfe einer sekundären Quelle zu verbessern, die in einigen Attributen eventuell Vorzüge birgt (beispielsweise DOIs enthält, wohingegen die primäre Quelle lediglich URLs besitzt), in anderen Attributen jedoch Unterschiede aufweisen kann (beispielsweise Tippfehler in einigen Titelangaben), welche ohne diese asymmetrische Behandlung eine Fusion der Titel oder gar der gesamten Records verhindern würde.

8.3 Fusion zweier Autorennamen

Wie bereits zu Beginn dieses Kapitels erläutert, nimmt die Fusion zweier Autorennamen eine Sonderstellung ein. Prinzipiell handelt es sich bei Autorennamen (*aname*) um komplexe Objekte, die aus einzelnen Namensteilen $(n_1, n_2, \cdots, n_p)$ bestehen, welche wiederum zu den simplen Objekten gezählt werden. Es ist jedoch nicht möglich, die Autorennamen nach dem in Kapitel 7 vorgestellten Partnersuche-Algorithmus zu fusionieren, wie wir in Abschnitt 8.3.1 sehen werden.

Daher werden wir Autorennamen auf andere Weise fusionieren als die übrigen komplexen Objekte. Zunächst werden wir die gesamten Namen als einheitliche Strings (und somit simple Objekte) auffassen und diese, falls nötig, einer Vorverarbeitung unterziehen (Abschnitt 8.3.2). Anschließend werden wir versuchen, die einzelnen Namensteile zu identifizieren, da diese je nach Typ von unterschiedlicher Bedeutung sein werden (Abschnitt 8.3.3). Abschließend wird der Algorithmus, welcher zur Fusion zweier derartig vorbereiteter Autorennamen verwendet wird, skizziert (Abschnitt 8.3.4). Auf eine exakte Beschreibung des äußerst umfangreichen Fusionsprozesses wird jedoch verzichtet. Diese ist dem ausführlich kommentierten Quelltext der beiliegenden CD-ROM zu entnehmen. In Abschnitt 8.3.5 wird schließlich die konkrete Fusion zweier einzelner Namensteile diskutiert.

8.3.1 Probleme bei der Fusion zweier Autorennamen

Wie bereits erwähnt, ist es nicht ohne Weiteres möglich, den in Kapitel 7.2 vorgestellten Partnersuche-Algorithmus auf Autorennamen, die entsprechend unseres Datenmodells Listen von Namensteilen darstellen, anzuwenden. Dies liegt darin begründet, dass einzelne Namensteile nicht völlig separat betrachtet werden können, sondern immer eine Einheit bilden. Ein einzelnes Record oder ein einzelner Autorname benennt stets ein reales Objekt (einen Artikel oder eine Person), ein einzelner Namensteil jedoch nicht. Eine Aufzählung einzelner Artikel oder Personen kann in nahezu beliebiger Reihenfolge geschehen und dennoch einen Sinn ergeben; so können Personen beispielsweise in auf- oder absteigender Reihenfolge ihres Alters oder ihrer Größe, in alphabetischer Sortierung ihrer Vor- oder Nachnamen oder – wie im Falle unserer Daten – nach ihrer Priorität bzgl. des von ihnen verfassten Artikels genannt werden. Keine dieser Aufzählungen ist jedoch prinzipiell falsch, denn sowohl "Horst, Günther, Eduard", "Günther, Eduard, Horst", "Eduard, Günther, Horst" und sämtliche weiteren Permutationen jener Namen sind theoretisch denkbar.

Bei Namensteilen verhält es sich dagegen anders, hier spielt die innere Logik des Aufbaus eines Namens eine Rolle, und eine Vertauschung von Namensteilen kann in vielen Fällen zu unsinnigen Ergebnissen führen; während "Hans Peter von der Wurst" einen durchaus denkbaren – wenn auch nicht wünschenswerten – Personennamen darstellt, so ergeben die meisten Permutationen, wie beispielsweise "der Wurst Peter Hans von", keinen Sinn.

Beim Partnersuche-Algorithmus in Kapitel 7.2 werden Singles der zweiten Liste, die keine Dubletten sind, einfach ans Ende des Ergebnisses angefügt. Ein solches Vorgehen ist im Falle der Namensteile aus oben genanntem Grund nicht möglich, da hier falsche Ergebnisse der Form

"Hans Wurst" $\bowtie$ "Hans P. Wurst" $\longrightarrow$ "Hans Wurst P."

auftreten würden. Darüber hinaus existieren eine Reihe weiterer Probleme, von welchen die beiden gravierendsten nachfolgend beschrieben werden:

Bindestriche
Einzelne Namensteile können mittels eines Bindestriches miteinander verbinden werden. Bei einer Fusion nach dem Partnersuche-Algorithmus könnte der Bindestrich als eigener Namensteil berücksichtigt werden, doch träte er nur in der sekundären Quelle auf, so ergäben sich auch hier inkorrekte Ergebnisse der Art

"Hans Peter Wurst" $\bowtie$ "Hans-Peter Wurst" $\longrightarrow$ "Hans Peter Wurst-"

Würde der Bindestrich als Teil einer der angrenzenden Namen angesehen, so ergäben sich ebenfalls Probleme bzgl. der Reihenfolge (welche sich ja nach der Reihenfolge der primären Quelle richtet), wie beispielsweise

"P. H. Wurst" $\bowtie$ "Hans-Peter Wurst" $\longrightarrow$ "Peter Hans-Wurst"

bzw.

“P. H. Wurst” ⋈ “Hans-Peter Wurst” ⟶ “-Peter Hans Wurst”.

Würden die mittels Bindestrich verbundenen Namen als ein einziger Namensteil aufgefasst, so bestünde das nachfolgend beschriebene Problem der Verbindung von Namensteilen.

Verbindungen von Namensteilen
Ein oder mehrere Namensteile können zu einem einzigen verbunden werden, was bei Verwendung des Partnersuche-Algorithmus' zu Verdopplungen führen würde, beispielsweise bei

“Hans Peter Wurst” ⋈ “Hanspeter Wurst” ⟶ “Hanspeter Peter Wurst”

bzw.

“Hanspeter Wurst” ⋈ “Hans Peter Wurst” ⟶ “Hanspeter Wurst Peter”.

8.3.2 Vorverarbeitung zweier Autorennamen

Um obiges Problem verbundener Namensteile zu lösen, werden zunächst beide zu fusionierenden Namen als Einheiten betrachtet (d.h. als simple Objekte) und auf gemeinsame Teilstrings hin untersucht. Dabei werden mehrere auseinander geschriebene Namensteile stets einer Verbindung selbiger vorgezogen, d.h. wir wählen “Hans Peter” statt “Hanspeter”. Hierzu wird derjenige Name, der eine solche Verbindung enthält, entsprechend dem Namen der anderen Quelle modifiziert. Selbiges gilt für Namen, die in einer Quelle direkt, in der anderen mittels eines Bindestrichs verbunden sind (z.B. “Hanspeter” und “Hans-Peter”). Auch hier wählen wir die Version, die den Bindestrich enthält, da diese eine zusätzliche Information in sich birgt.

Weiterhin wird an dieser Stelle, mit Blick auf die nachfolgende Identifikation einzelner Namensteile, versucht, gemeinsame Nachnamen in beiden Quellen zu finden. Da die Namen laut Konvention in DBLP – und somit auch in allen bei der Fusion betrachteten Datenformaten – in der Form ‘Vorname(n) Nachname’ vorliegen (sollten), werden wir letzteren Namensteil stets als Nachnamen (SURNAME) markieren (vgl. Abschnitt 8.3.3). Probleme sind nur dann zu erwarten, wenn es sich bei den beiden zu fusionierenden Autorennamen um Permutationen handelt, wie diese gerade bei Namen des asiatischen Sprachraumes verstärkt auftreten (z.B. “Yabo Dong” und “Dong Yabo”). Hier wird im Vorhinein anhand geeigneter Entscheidungsstrategien versucht, den Nachnamen einer Person zu identifizieren und die Namensteile dabei in die richtige Reihenfolge zu bringen.

Gerade bei asiatischen Namen ist eine Identifikation des Nachnamens oftmals möglich, da sich laut Wikipedia beispielsweise “die meisten Chinesen nur etwa 20 sehr häufig vorkommende Namen [teilen]”[4]. Hier kann mittels einer simplen Datei, welche eine Reihe geläufiger chinesi-

[4] `http://de.wikipedia.org/wiki/Familienname`, Abschnitt “Asien”

scher Nachnamen enthält, in vielen Fällen entschieden werden, bei welchem Namensteil es sich um den Nachnamen handelt. In obigem Beispiel würde die Entscheidung auf die Schreibweise "Yabo Dong" fallen, da es sich bei "Dong", nicht aber bei "Yabo", um einen entsprechend häufig auftretenden Nachnamen handelt.

In anderen Fällen hilft ein DBLP-Lookup, d.h. eine Anfrage an die Namenssuche von DBLP.[5] Hier werden bei Anfrage eines der Namen in einer Ergebnisliste alle Treffer aufgelistet, auch solche, die auf Permutationen beruhen. In dieser Liste muss nun lediglich nach beiden Schreibweisen gesucht werden; tritt eine von diesen erheblich öfter auf, so kann sie als korrekt angesehen werden.

8.3.3 Identifikation einzelner Namensteile

Eine Identifikation der Namensteile ist in DBLP normalerweise nicht üblich, da diese zahlreiche Probleme mit sich bringt (vgl. [Ley09]). Dennoch ist es aus den in Abschnitt 8.3.1 genannten Gründen äußerst hilfreich, Namensteile zu identifizieren, um ihnen somit unterschiedliche Bedeutung bzgl. ihrer Position im Fusionsergebnis zuzuweisen.

Wir definieren daher die folgenden Typen von Namensteilen:

- PRENAME (P, Vorname)
- INITIAL (I, Initial)
- SURNAME (S, Nachname)
- TITLE (T, Titel)
- DASH (D, Bindestrich)
- OTHERS (O, sonstige)

Anschließend wird jedem Namensteil von *aname* ein entsprechender Typ zugeordnet, wobei simple Regeln zum Einsatz kommen:

- Ein INITIAL besteht aus einem einzelnen Buchstaben, dem ein Punkt folgt.
- TITLE und OTHERS werden über fest definierte Werte identifiziert. Ein TITLE ist beispielsweise "Dr." oder "Prof."[6], während mit OTHERS Zusätze wie "Junior" bzw. "Jr." oder römische Zahlen wie "III" bezeichnet werden.

[5] `http://dblp.uni-trier.de`

[6] In wissenschaftlichen Arbeiten ist die Angabe eines akademischen Titels nicht üblich, weshalb wir weder im Datenbestand von DBLP noch auf den Servern der in Kapitel 3 vorgestellten Extraktionsquellen derartige Titel finden werden. Da wir die hier vorgestellten Fusionsalgorithmen jedoch auch

- Ein DASH entspricht stets einem einfachen Minuszeichen ("-"), da die Daten in normierter Form vorliegen und das Minuszeichen das einzige ASCII-Zeichen ist, welches als Bindestrich verwendet werden kann.

- Der letzte Namensteil wird i.d.R. als SURNAME definiert. Einzige Ausnahme bilden Namensteile des Typs OTHERS, die einem Nachnamen nachgestellt sein können. Wurde der letzte Namensteil entsprechend als Typ OTHERS identifiziert, so erhält der vorletzte den Typ SURNAME.

- Jeder andere Namensteil, der nach obigen Regeln nicht zugeordnet werden konnte, gilt als PRENAME.

Auf diese Weise können die Namensteile in vielen Fällen in korrekter Weise zugeordnet werden, wie die folgenden Beispiele einiger, mit Annotation der jeweiligen Typen (in Kurzschreibweise) versehener, Namen zeigen.

Hans/P Peter/P Wurst/S
Hans/P -/D Peter/P Wurst/S
Hans/P P./I Wurst/S III/O
Dr./T H./I -/D P./I Wurst/S Jr./O

In anderen Fällen schlägt die korrekte Zuordnung jedoch fehl, beispielsweise bei

Hans/P Roth/P -/D Wurst/S
Hans/P von/P der/P Wurst/S
Hanswurst/S

Hier werden in den ersten beiden Fällen Teile der Nachnamen ("Roth-Wurst" und "von der Wurst") fälschlicherweise als Vornamen identifiziert, während der Name im letzten Fall nur aus einem einzigen Namensteil besteht, welcher somit im wörtlichen Sinne weder ein *Vor*- noch ein *Nach*name ist.

Dies braucht uns allerdings nicht zu grämen, da dies keinerlei negative Auswirkungen auf den Ablauf der Fusion, welcher im folgenden Abschnitt skizziert ist, hat. Die Identifikation der Namensteile dient im Wesentlichen dem Zweck der korrekten Ersetzung von Initialen durch Vornamen, sowie der Identifikation des letzten Namensteils (des Typs PRENAME), um eventuell gefundene zusätzliche Vornamen an der richtigen Stelle einzufügen.

bei der Fusion mit unstrukturierten Quellen (vgl. Kapitel 9 und 10) nutzen möchten, in welchen durchaus derartige Angaben auftreten können, ergibt eine Definition jenes Types einen Sinn.

8.3.4 Ablauf der Fusion zweier Autorennamen

Der komplette Algorithmus, mit welchem zwei Autorennamen fusioniert werden, ist äußerst kompliziert und beachtet eine Reihe von Spezialfällen, die während der Testphase aufgetreten sind. An dieser Stelle soll daher lediglich eine grobe Skizze jenes Algorithmus' anhand konkreter Beispiele dargestellt werden.

Zunächst werden beide Namen ($aname_1$ und $aname_2$) einer Vorverarbeitung entsprechend der in Abschnitt 8.3.2 erklärten Vorgehensweisen unterzogen, bei welcher Permutationen beseitigt und verbundene Namensteile getrennt werden. Anschließend werden den Namensteile beider Namen die jeweiligen Typen nach obigen Regeln zugeordnet.

Beispiel 8.1
Seien beispielsweise $aname_1$ = "Hans Wurst" und $aname_2$ = "H.-P. Wurst". Dann können wir uns beide Namen als Listen vorstellen, wie sie in Abbildung 8.1 zu sehen sind. Weiterhin stellen wir uns zwei Zeiger (pointer p_1 und p_2) vor, welche zu Beginn des Algorithmus jeweils auf den ersten Namensteil zeigen. Die Ergebnisliste ($aname_{result}$)ist zu Anfang leer.

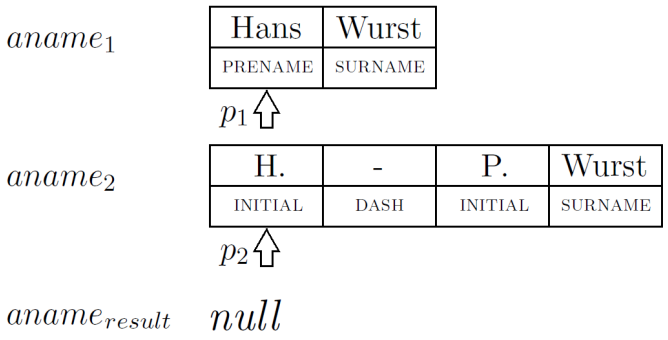


Abb. 8.1: Beispiel der Fusion zweier Autorennamen I

Nun werden jeweils beide durch die Zeiger markierten Felder betrachtet, damit mittels geeigneter Regeln entschieden werden kann, welcher Wert dem Ergebnis hinzugefügt werden soll. Im Beispiel zeigt p_1 auf einen Vornamen, während p_2 auf ein Initial weist. In diesem Fall gilt die Regel: 'Beginnt der Vorname mit dem Buchstaben des Initials, so füge ihn dem Ergebnis hinzu und inkrementiere beide Zeiger'. Den hieraus resultierenden Zustand unseres Beispiels zeigt Abbildung 8.2.

p_1 zeigt nun auf einen Nachnamen, p_2 auf einen Bindestrich. Die entsprechende Regel lautet hier: 'Weist ein Zeiger auf einen Nachnamen, der andere auf einen Namensteil eines anderen Typs, so übernimm jenen anderen Namensteil und inkrementiere dessen Zeiger so lange, bis auch er auf einen Nachnamen verweist.' An dieser Stelle zeigt sich der Nutzen der Namenstypen. Diese stellen hier sicher, dass die Namensteile an korrekter Position eingefügt werden. Abbildung 8.3 zeigt die aktuellen Werte des Beispiels.

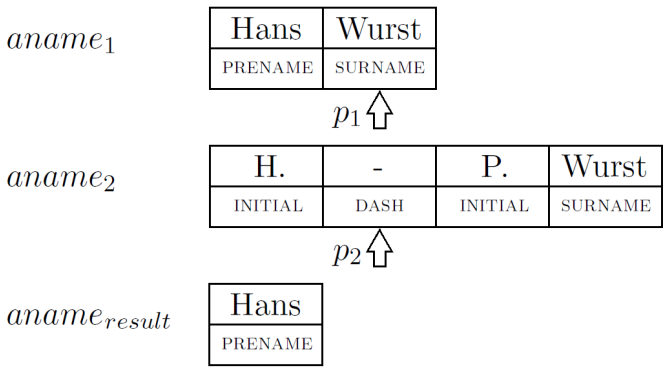


Abb. 8.2: Beispiel der Fusion zweier Autorennamen II

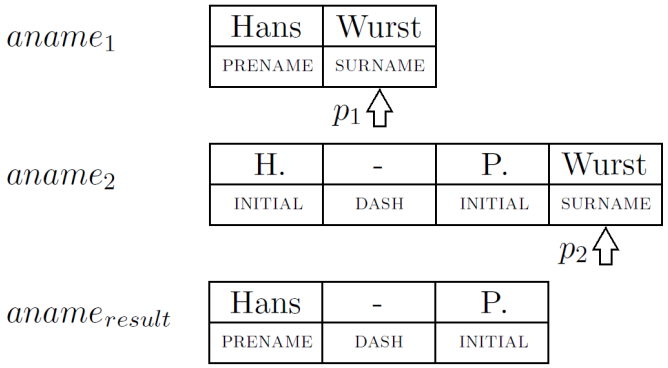


Abb. 8.3: Beispiel der Fusion zweier Autorennamen III

Nun weisen beide Zeiger auf Namensteile des Typs SURNAME, was eine entsprechende Fusion beider Teile bedingt. In unserem Fall sind beide Werte identisch; es tritt kein Konflikt auf. Die zur Anwendung kommende Regel lautet hier: 'Sind Typ und Wert identisch, so übernimm den Wert ins Ergebnis und inkrementiere beide Zeiger.' Hieraus ergibt sich die in Abbildung 8.4 dargestellte Situation.

Abschließend greift nun die Regel: 'Weisen beide Zeiger auf *null*, so breche ab und liefere a_{result} als Ergebnis. Die Fusion war erfolgreich.' Ein Wert von *null* bedeutet in diesem Fall, dass die Zeiger auf eine Position weisen, die über die Länge der Liste hinausgeht. Wie wir sehen, enthält $aname_{result}$ das erwartete Ergebnis.

Während obiges Beispiel den Eindruck erwecken könnte, die Fusion zweier Autorennamen würde stets derart unproblematisch ablaufen, sollen die folgenden Beispiele die Komplexität jener Problemstellung verdeutlichen. Wir werden dabei nur die 'kritischen Stellen' untersuchen, d.h. jene Stationen, an welchen der Algorithmus problematische Entscheidungen fällen muss. Auch hier soll auf eine komplette Beschreibung der Lösung – die der Dokumentation des Quelltextes zu entnehmen ist – verzichtet werden; es gilt lediglich, die Probleme aufzuzeigen.

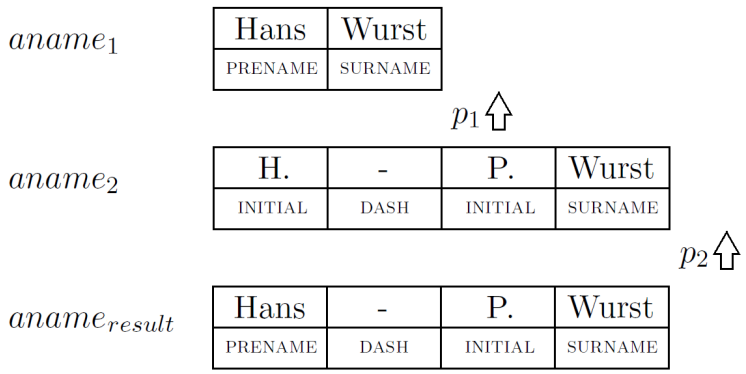


Abb. 8.4: Beispiel der Fusion zweier Autorennamen IV

Die Beispiele sind in einer vereinfachten Form dargestellt, entsprechen aber prinzipiell dem oben vorgestellten Modell; eine Box soll hier stets die Position des jeweiligen Zeigers verdeutlichen.

Beispiel 8.2
Betrachten wir zunächst den folgenden Fall:

$aname_1$	Günter/P	Sülzbach/S
$aname_2$	Gunther/P	Sultzbach/S
$aname_{result}$	*null*	

Hier unterscheiden sich sowohl die Vornamen als auch im Nachnamen, was uns vor das Problem der Fusion zweier Namensteile (siehe Abschnitt 8.3.5) stellt. Können beide Namensteile fusioniert werden, so wird das Ergebnis jener Fusion in $aname_{result}$ übernommen, und anschließend werden beide Zeiger inkrementiert.

Beispiel 8.3
Ein anderes Problem liegt hier vor:

$aname_1$	Erich/P	Horst/P	Günther/P	Sülzbach/S
$aname_2$	Horst/P	Günther/P	Sülzbach/S	
$aname_{result}$	*null*			

Hier sind die Namensteile weder identisch noch ähnlich (vgl. Abschnitt 8.3.5) und können somit nicht fusioniert werden ($n_{11} \not\approx n_{21}$). Es stellt sich die Frage, welcher Namensteil nun ins Ergebnis übernommen werden soll. Hierzu ist eine Betrachtung der übrigen Namensteile notwendig: Wir müssen prüfen, ob n_{11} irgendwo in $aname_2$ bzw. n_{21} in $aname_1$ auftritt. Im Beispiel ist ersteres der Fall, letzteres nicht. Daher lautet hier die Regel: 'Übernimm den Namensteil, welcher *nicht* im anderen Namen auftritt, ins Ergebnis und inkrementiere dessen Zeiger.' Wir erhalten also:

$aname_1$	Erich/P	Horst/P	Günther/P	Sülzbach/S
$aname_2$	Horst/P	Günther/P	Sülzbach/S	
$aname_{result}$	Erich/P			

Man erkennt, dass der Rest der Fusion keine Probleme mehr bereiten wird und wir das zufrieden stellende Ergebnis "Erich Horst Günther Sülzbach" erhalten werden. Es ist zu beachten, dass obige Regel symmetrisch ist, d.h. sie auch bei Vertauschung der Quellen zum Tragen käme.

Beispiel 8.4
Anders verhält es sich in folgendem Beispiel:

$aname_1$	Anton/P	Bertram/P	Wunderlich/S
$aname_2$	Bertram/P	Anton/P	Wunderlich/S
$aname_{result}$	*null*		

Hier ergibt eine Untersuchung der Vorkommen beider Namensteile im jeweils anderen Namen ein positives Ergebnis. Es muss also entschieden werden, welcher dieser Namensteile ins Ergebnis einfließt, und welcher verworfen wird. Da wir auch im Falle der Namensteile die primäre Quelle bevorzugen möchten, ergibt sich hier erneut eine Asymmetrie bei der Fusion der Namen. Die Regel, welche Anwendung findet, lautet: 'Übernimm den Namensteil der primären Quelle ins Ergebnis und inkrementiere deren Zeiger.'

Ist dies geschehen, so weisen beide Zeiger auf identische Namensteile, was keine Probleme verursacht. Danach jedoch ergibt sich die folgende Situation:

$aname_1$	Anton/P	Bertram/P	Wunderlich/S
$aname_2$	Bertram/P	Anton/P	Wunderlich/S
$aname_{result}$	Anton/P	Bertram/P	

Hier zeigt sich, dass die in Beispiel 8.1 völlig logisch erscheinende Regel 'Weist ein Zeiger auf einen Nachnamen, der andere auf einen Namensteil eines anderen Typs, so übernimm jenen anderen Namensteil[...]' hier zu einem falschen Ergebnis führen würde: "Anton Bertram Anton Wunderlich". Wir müssen also in jenem Fall außerdem sicher stellen, dass der einzeln auftretende Namensteil nicht bereits im Ergebnis vorhanden ist. Da das Ergebnis bereits den Namen "Anton" enthält, würde dieser also *nicht* erneut hinzugefügt. Gleiches gilt im Übrigen nicht nur für gleiche, sondern auch für ähnliche Namensteile (z.B. bei "Anton Bertram Wunderlich" und "Bertram Antonio Wunderlich") oder die Ersetzung von Initialen (z.B. bei "A. B. Wunderlich" und "Bertram Anton Wunderlich").

Die Zahl weiterer Probleme und Sonderfälle ist groß, und mit obigen Beispielen sollte wie bereits erwähnt lediglich ein Eindruck der Komplexität dieser Problematik geboten werden.

Die wichtigste Regel jener Fusion lautet jedoch: ‘Befindet sich der Algorithmus in einem Zustand, in welchem keine Lösung gefunden werden kann, so breche ab und gib < FAILED > als Ergebnis aus.’ Nach unserer Definition in Kapitel 6.2.2 entspricht dies genau der Aussage “$aname_1 \not\approx aname_2$”. Im Falle, dass zwei Attribute nicht fusioniert werden können, wird jedoch, wie in Abschnitt 8.2 bei der Fusion zweier Records definiert, nach dem Grundsatz einer asymmetrischen Fusion zu Gunsten der primären Quelle stets der Wert des ersten Attributs als Ergebnis übernommen. Die Software wird demnach eine Meldung (“Fusion failed!”) ausgeben, am ursprünglichen Wert jedoch keine Änderungen vornehmen; der entsprechende Fall bedarf einer manuellen Kontrolle.

8.3.5 Fusion zweier Namensteile

Neben der korrekten Zuordnung der Namensteile stellt auch deren konkrete Fusion ($n_{1i} \bowtie n_{2j}$), wie sie in Beispiel 8.2 des vorherigen Abschnitts angesprochen wurde, ein Problem dar. Hier muss in geeigneter Weise entschieden werden, welcher von zwei Namensteilen der ‘bessere’ ist.

Hierzu muss die Software zunächst herausfinden, ob die ungleichen Namensteile sich fusionieren lassen. Sind ihre Werte identisch, so gilt nach Gleichung 8.2 auf Seite 130 $n_{result} = n_{1i}$. Handelt es sich um ein Initial und einen Vornamen, so können diese genau dann fusioniert werden, wenn der Vorname mit dem Initial beginnt; das Ergebnis der Fusion ist der Vorname, da dieser einen höheren Informationsgehalt besitzt. Entspricht der erste Buchstabe des Vornamens nicht dem Initial, so können die beiden Namensteile nicht fusioniert werden.

Liegen dagegen zwei Namensteile vor, deren Werte nicht identisch sind und von welchen keiner ein Initial ist, so muss überprüft werden, ob diese sich zumindest *ähnlich* sind. Da es sich um einfache Strings handelt, kann dies mit Hilfe eines klassischen Stringmatching-Algorithmus’ geschehen. Wir werden hierzu den Levenshtein-Algorithmus ([Lev66]) verwenden, müssen jedoch unbedingt darauf achten, auch die Länge der Namensteile mit einfließen zu lassen: Die offensichtlich äußerst ähnlichen Namensteile “Joseph” und “Josef” haben eine Levenshtein-Distanz von 2 – ebenso wie die völlig verschiedenen, im asiatischen Sprachraum oftmals auftauchenden Namensteile “He” und “Ma”. Sind sich die Namensteile – wie etwa in obigem Beispiel 8.2 – ähnlich, so muss entschieden werden, welcher Teil ins Ergebnis einfließt, d.h. es muss $n_{11} \bowtie n_{21}$ berechnet werden. Auch hier ist also die Wahl einer geeigneten Strategie zur Konfliktlösung, wie sie in Kapitel 5.3.2 vorgestellt wurden, von Nöten. Sind sie sich nicht ähnlich, wie beispielsweise “Erich” und “Horst” in Beispiel 8.3, so wird das Ergebnis < FAILED > zurück geliefert. Der Algorithmus zur Fusion zweier Autorennamen muss dann entscheiden, wie in einem solchen Fall weiter verfahren werden soll.

Hier können ähnliche Überlegungen wie bei den Titeln angestellt werden (vgl. Kapitel 8.1.2), d.h. solche Namensteile bevorzugt werden, die länger sind oder mehr Sonderzeichen, d.h. Zeichen mit höheren Zeichencodes, enthalten. Auch eine buchstabenweise Kombination beider Namensteile ist denkbar– wie beispielsweise bei der Fusion von n_{1i} = “Fréderic” und n_{2j} =

"Fredéric" –, sofern sich diese tatsächlich nur durch eben solche Sonderzeichen unterscheiden.[7] Um dies zu kontrollieren, können beide Namensteile in reine ASCII-Zeichen konvertiert werden, was in obigem Fall beide Strings auf "Frederic" abbilden würde. Ein zeichenweiser Vergleich, bei welchem jeweils das Zeichen mit höherem Code gewählt wird, ergäbe somit das Ergebnis n_{result} = "Frédéric", welches in der Tat den gemeinsamen Informationsgehalt beider Quellen in sich vereint.

Auch ein DBLP-Lookup, d.h. eine Anfrage an die Namenssuche von DBLP, wie sie in Abschnitt 8.3.2 erläutert wurde, kann in Bezug auf einzelne Namensteile hilfreich sein. In Beispiel 7.6 des vorherigen Kapitels (Seite 127) wurde erwähnt, dass zwei der dort betrachteten Namen bei IEEE Xplore offensichtliche OCR-Fehler aufweisen: Bei "Jiaii Zhang" wurde ein "n" fälschlicherweise als "ii" eingelesen, während bei "Xiaoclong" ein "d" als "cl" interpretiert wurde. Bei der Entscheidung, welcher der Namensteile gewählt werden soll, hilft ein DBLP-Lookup weiter: Sowohl für "Jiaii" als auch für "Xiaoclong" werden hier keine Treffer erzielt, die Namensteile "Jian" und "Xiaodong" sind jedoch bekannt und treten in zahlreichen Autorennamen auf.

Auch an dieser Stelle wird es nicht möglich sein, jeden in der Praxis auftretenden Fall vorherzusehen. Ist es der Software nicht möglich, zwei ähnliche Namensteile zu fusionieren, so wird auch hier das Ergebnis < FAILED > geliefert.

[7] Diese Vorgehensweise entspricht einer Anwendung der Strategie MEET IN THE MIDDLE, wie sie in Kapitel 5.3.3 erläutert wurde: Wir erstellen einen neuen Wert, der beiden Ausgangswerten möglichst ähnlich ist und versuchen zudem, die jeweils informationsreicheren Teilwerte einfließen zu lassen.

Kapitel 9

Praxisstudie: Konferenzprogramme in HTML-Format

Bisher haben wir uns intensiv mit der Fusion zweier strukturierter Quellen beschäftigt. Wir bezeichneten diese Art der Fusion als *asymmetrisch*, da wir die Quellen in unterschiedlicher Weise gewichteten: Die Informationen der primäre Quelle wurden im Falle eines unlösbaren Konflikts stets denen der sekundären Quelle vorgezogen.

Nun wollen wir jedoch die Asymmetrie weiter erhöhen, indem wir als sekundäre Quelle eine *unstrukturierte* Quelle zulassen. Dabei kann es sich wie in Kapitel 1.4 erläutert, um unterschiedlichste Ausprägungen der Unstrukturiertheit handeln, von verschiedenen Arten von HTML-Dokumenten bis hin zu freiem Text der natürlichen Sprache. Aus diesen Quellen müssen die jeweiligen Informationen zunächst extrahiert werden, um anschließend die Fusion mit der primären Quelle durchführen zu können. Hier schließt sich also der Kreis: Wir werden nun sowohl eine regelbasierte Extraktion als auch eine asymmetrische Fusion bibliographischer Daten vornehmen.

9.1 Fusion mit einem Konferenzprogramm in HTML-Format

In Kapitel 6 haben wir gesehen, dass es uns oftmals möglich ist, bestehende bibliographische Daten einer primären Quelle durch Fusion mit einer zweiten Datenquelle zu ergänzen oder gar zu korrigieren. In vielen Fällen existieren jedoch derartige Daten nicht in strukturierter Form, sondern liegen beispielsweise in HTML-, PDF- oder DOC-Dateien vor.

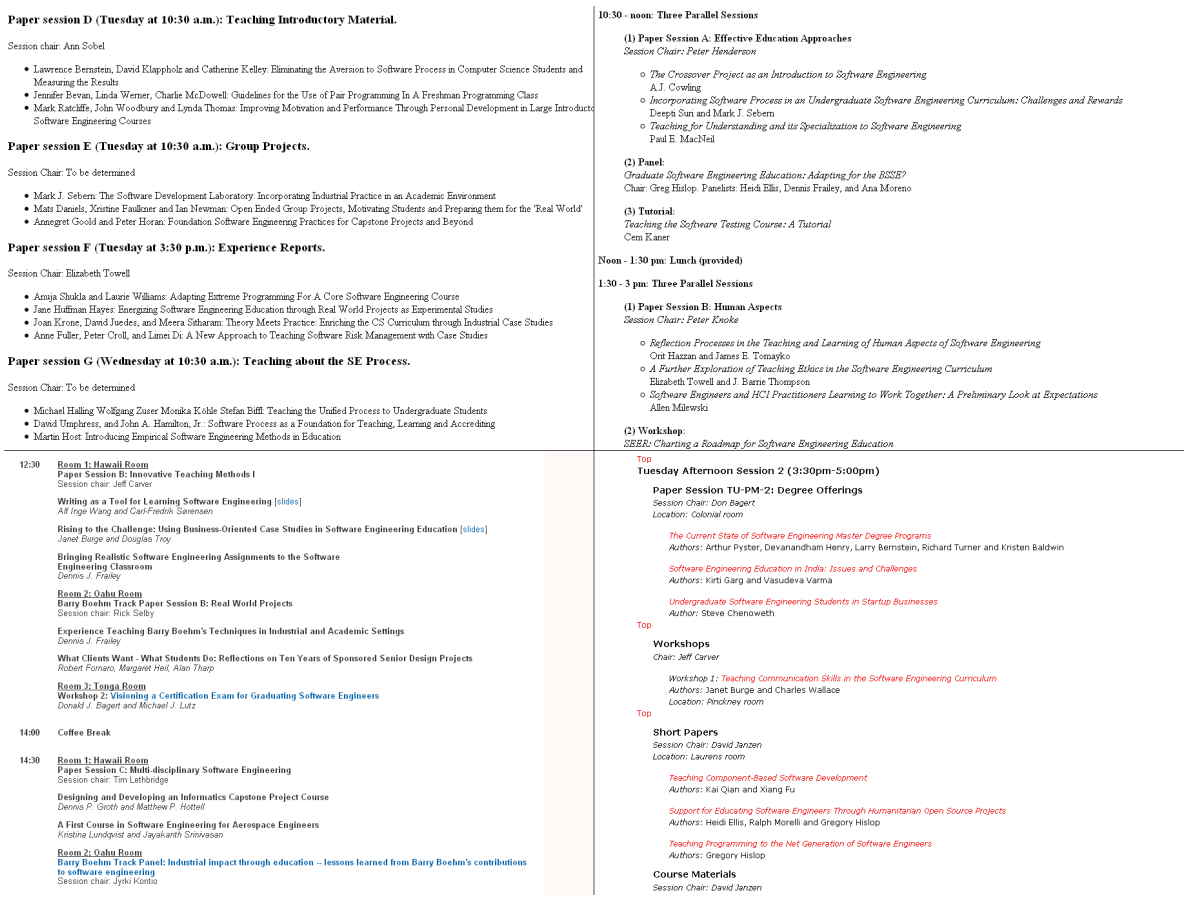
Abbildung 9.1 zeigt bibliographische Daten der "Fourth Annual IEEE International Conference on Pervasive Computing and Communications" (PerCom 2006), links bei IEEE Xplore und rechts innerhalb des Konferenzprogramms der offiziellen Konferenz-Website. Man erkennt, dass beide Seiten die gleichen Artikel beinhalten (durch graue Linien markiert), das Konferenzprogramm jedoch zusätzliche Informationen bietet: Zum einen sind die Autorennamen bei Xplore

Abb. 9.1: Konferenzdaten bei IEEE Xplore und in einem Konferenzprogramm: Das Konferenzprogramm (rechts) liefert vollständige Vornamen sowie Zwischenüberschriften. *Quellen:* `http://ieeexplore.ieee.org/xpl/tocresult.jsp?isnumber=33716&isYear=2006`, `http://cnd.iit.cnr.it/percom2006/program.html`

lediglich als Initialen vorhanden, während sie im Konferenzprogramm ausgeschrieben sind, zum anderen beinhaltet das Programm Zwischenüberschriften, die bei Xplore völlig fehlen.

Es ist also möglich, Informationen aus dem Konferenzprogramm zu extrahieren und diese mit den (entweder neu extrahierten oder bereits in DBLP enthaltenen) Daten von Xplore zu fusionieren. Diese Arbeit wird bisher oftmals manuell durchgeführt, was äußerst mühselig ist. Ziel wird es daher sein, diese Informationen mit Hilfe von Software automatisch zu extrahieren und anschließend mit einer $\text{BHT}_{\text{c/j}}$-Datei (bei Neueintrag) bzw. einer BHT_{cite}-Datei (im Falle bestehender *records*) gemäß den Szenarien F-1 und F-2' in Kapitel 6.1 zu 'mergen'.

Leider können wir bei der Extraktion jener Daten nicht ebenso wie in Kapitel 4 vorgehen. Konferenzen werden i.d.R. nicht immer am gleichen Ort veranstaltet, und so wechselt der Veranstalter einer Konferenz ebenso wie der Designer der Webseiten. Betrachtet man mehrere Konferenzsites ein und derselben Konferenz aus unterschiedlichen Jahren, so stellt man fest, dass sich diese oftmals grundlegend in Aufbau und Präsentation der Daten unterscheiden. Abbildung 9.2 zeigt Ausschnitte aus vier solchen Konferenz-Websites der "IEEE Conference on Software Engineering Education and Training" (CSEE&T) aus den Jahren 2002, 2004, 2006 und 2008. Wie unschwer zu erkennen, unterscheiden sich die Darstellungen bereits optisch deutlich voneinander. Betrachtet man den entsprechenden HTML-Quellcode, so werden die Unterschiede noch offensichtlicher.



Paper session D (Tuesday at 10:30 a.m.): Teaching Introductory Material.

Session chair: Ann Sobel

- Lawrence Bernstein, David Klappholz and Catherine Kelley: Eliminating the Aversion to Software Process in Computer Science Students and Measuring the Results
- Jennifer Bevan, Linda Werner, Charlie McDowell: Guidelines for the Use of Pair Programming In A Freshman Programming Class
- Mark Ratcliffe, John Woodbury and Lynda Thomas: Improving Motivation and Performance Through Personal Development in Large Introductc Software Engineering Courses

Paper session E (Tuesday at 10:30 a.m.): Group Projects.

Session Chair: To be determined

- Mark J. Sebern: The Software Development Laboratory: Incorporating Industrial Practice in an Academic Environment
- Mats Daniels, Xristine Faulkner and Ian Newman: Open Ended Group Projects, Motivating Students and Preparing them for the 'Real World'
- Annegret Goold and Peter Horan: Foundation Software Engineering Practices for Capstone Projects and Beyond

Paper session F (Tuesday at 3:30 p.m.): Experience Reports.

Session Chair: Elizabeth Towell

- Anuja Shukla and Laurie Williams: Adapting Extreme Programming For A Core Software Engineering Course
- Jane Huffman Hayes: Energizing Software Engineering Education through Real World Projects as Experimental Studies
- Joan Krone, David Juedes, and Meera Sitharam: Theory Meets Practice: Enriching the CS Curriculum through Industrial Case Studies
- Anne Fuller, Peter Croll, and Limei Dr: A New Approach to Teaching Software Risk Management with Case Studies

Paper session G (Wednesday at 10:30 a.m.): Teaching about the SE Process.

Session Chair: To be determined

- Michael Halling Wolfgang Zuser Monika Köhle Stefan Biffl: Teaching the Unified Process to Undergraduate Students
- David Umphress, and John A. Hamilton, Jr.: Software Process as a Foundation for Teaching, Learning and Accrediting
- Martin Host: Introducing Empirical Software Engineering Methods in Education

10:30 - noon: Three Parallel Sessions

(1) Paper Session A: Effective Education Approaches
Session Chair: Peter Henderson

- *The Crossover Project as an Introduction to Software Engineering*
  A.J. Cowling
- *Incorporating Software Process in an Undergraduate Software Engineering Curriculum: Challenges and Rewards*
  Deepti Suri and Mark J. Sebern
- *Teaching for Understanding and its Specialization to Software Engineering*
  Paul E. MacNeil

(2) Panel:
Graduate Software Engineering Education: Adapting for the BSSE?
Chair: Greg Hislop. Panelists: Heidi Ellis, Dennis Frailey, and Ana Moreno

(3) Tutorial:
Teaching the Software Testing Course: A Tutorial
Cem Kaner

Noon - 1:30 pm: Lunch (provided)

1:30 - 3 pm: Three Parallel Sessions

(1) Paper Session B: Human Aspects
Session Chair: Peter Knoke

- *Reflection Processes in the Teaching and Learning of Human Aspects of Software Engineering*
  Orit Hazzan and James E. Tomayko
- *A Further Exploration of Teaching Ethics in the Software Engineering Curriculum*
  Elizabeth Towell and J. Barrie Thompson
- *Software Engineers and HCI Practitioners Learning to Work Together: A Preliminary Look at Expectations*
  Allen Milewski

(2) Workshop:
SEER: Charting a Roadmap for Software Engineering Education

12:30 Room 1: Hawaii Room
Paper Session B: Innovative Teaching Methods I
Session chair: Jeff Carver

Writing as a Tool for Learning Software Engineering [slides]
Alf Inge Wang and Carl-Fredrik Sørensen

Rising to the Challenge: Using Business-Oriented Case Studies in Software Engineering Education [slides]
Janet Burge and Douglas Troy

Bringing Realistic Software Engineering Assignments to the Software Engineering Classroom
Dennis J. Frailey

Room 2: Oahu Room
Barry Boehm Track Paper Session B: Real World Projects
Session chair: Rick Selby

Experience Teaching Barry Boehm's Techniques in Industrial and Academic Settings
Dennis J. Frailey

What Clients Want - What Students Do: Reflections on Ten Years of Sponsored Senior Design Projects
Robert Fornaro, Margaret Heil, Alan Tharp

Room 3: Tonga Room
Workshop 2: Visioning a Certification Exam for Graduating Software Engineers
Donald J. Bagert and Michael J. Lutz

14:00 **Coffee Break**

14:30 Room 1: Hawaii Room
Paper Session C: Multi-disciplinary Software Engineering
Session chair: Tim Lethbridge

Designing and Developing an Informatics Capstone Project Course
Dennis P. Groth and Matthew P. Hottell

A First Course in Software Engineering for Aerospace Engineers
Kristina Lundqvist and Jayakanth Srinivasan

Room 2: Oahu Room
Barry Boehm Track Panel: Industrial impact through education – lessons learned from Barry Boehm's contributions to software engineering
Session chair: Jyrki Kontio

Top
Tuesday Afternoon Session 2 (3:30pm-5:00pm)

Paper Session TU-PM-2: Degree Offerings
Session Chair: Don Bagert
Location: Colonial room

The Current State of Software Engineering Master Degree Programs
Authors: Arthur Pyster, Devanandham Henry, Larry Bernstein, Richard Turner and Kristen Baldwin

Software Engineering Education in India: Issues and Challenges
Authors: Kirti Garg and Vasudeva Varma

Undergraduate Software Engineering Students in Startup Businesses
Author: Steve Chenoweth

Top
Workshops
Chair: Jeff Carver

Workshop 1: Teaching Communication Skills in the Software Engineering Curriculum
Authors: Janet Burge and Charles Wallace
Location: Pinckney room

Top
Short Papers
Session Chair: David Janzen
Location: Laurens room

Teaching Component-Based Software Development
Authors: Kai Qian and Xiang Fu

Support for Educating Software Engineers Through Humanitarian Open Source Projects
Authors: Heidi Ellis, Ralph Morelli and Gregory Hislop

Teaching Programming to the Net Generation of Software Engineers
Authors: Gregory Hislop

Course Materials
Session Chair: David Janzen

Abb. 9.2: Ausschnitte aus Konferenzprogrammen der CSEE&T: 2002 (links oben), 2004 (rechts oben), 2006 (links unten) und 2008 (rechts unten)
Quellen: `http://www.site.uottawa.ca/cseet2002/program.html#panel1`, `http://www.cs.virginia.edu/~horton/cseet04//program-sched.html`, `http://db-itm.shidler.hawaii.edu/cseet2006/program.php`, `http://www.csc2.ncsu.edu/conferences/cseet/schedule.php`

Um die Daten einer solchen Website zu extrahieren, sind daher andere Strategien notwendig. Lernende Wrapper (vgl. Kapitel 1.6.1) könnten hier Erfolg versprechen. Doch wir wollen versuchen, einfachere Wege zu gehen, um die entsprechenden Daten zu extrahieren. Wir besitzen ja bereits Informationen über die Daten, welche auf der Konferenzseite zu finden sind: Die Titel der Artikel sowie zumindest Teile der Autorennamen. Mit Hilfe dieser Informationen könnte es uns gelingen, die entsprechenden Daten von den Konferenzseiten zu extrahieren. Zur Identifikation der Titel, Autorennamen und Zwischenüberschriften innerhalb des Konferenzprogramms könnten wir das reine Wissen um Teile der entsprechenden Strings ausnutzen (z.B. nach Vornamen eines Autors suchen, von welchem uns Nachname und Initiale der Vornamen bekannt sind), oder auch Informationen bzgl. des diese Daten umgebenden HTML-Codes einfließen lassen. Wir könnten die DOM-Struktur der HTML-Seiten ausnutzen und versuchen, die Hierarchieebene ausfindig zu machen, auf welcher die entsprechenden Daten zu finden sind. Zwischenüberschriften, welche sich zwischen den Artikeln befinden, könnten identifiziert und eingefügt werden.

9.2 Praxisstudie: Konferenzprogramme

Um einen Eindruck zu erhalten, welche Probleme grundsätzlich zu erwarten sind – und um gleich eine relevante Menge an Testdaten zur Verfügung zu haben – wurde zunächst eine Studie durchgeführt, in welcher 100 ausgewählte Programm-Websites von Konferenzen des IEEE untersucht wurden. Mit Hilfe der dort gewonnenen Erkenntnisse werden wir anschließend versuchen, möglichst Erfolg versprechende Strategien zur Extraktion der Daten zu entwickeln.

9.2.1 Beschreibung

Zur Durchführung dieser Studie wurden 100 Testdatensätze manuell erfasst. Jeder Testdatensatz besteht neben einer fortlaufenden Nummer im Wesentlichen aus zwei URLs, von welchen der erste auf eine Konferenz bei IEEE Xplore verweist, der zweite auf die Programm-Website der gleichen Konferenz. Wird in diesem oder dem folgenden Kapitel ein bestimmter Testdatensatz angesprochen, so wird dessen Nummer in eckigen Klammern angegeben (z.B. [81]).

Zur Gewinnung der Daten wurden zunächst entsprechende Konferenzseiten mittels der Anfrage "ieee conference" an Google[1] ermittelt. Auf einer solchen Konferenz-Website wurde nun nach einem ausführlichen Programm (oftmals als 'Technical Program' bezeichnet) gesucht. Oft liegen entsprechende Programme als PDF-Dokumente vor, seltener auch in DOC-Dateien. Solchen Konferenzen wurden nicht in den Testdatensatz aufgenommen, sondern ausschließlich solche, deren Programm in HTML verfügbar ist. Anschließend wurde innerhalb von Xplore gesucht, ob die gleiche Konferenz auch dort gefunden werden konnte – was besonders bei erst kürzlich veranstalteten Konferenzen nicht der Fall war. Konnte die Konferenz in Xplore gefunden werden, so wurden beide URLs den Testdaten hinzugefügt. Zusätzlich wurden auch der Name (bzw. das Akronym) sowie die Hauptseite der Konferenz dort eingetragen. Beide Daten haben jedoch für diese Studie keine direkte Bedeutung und dienen nur dazu, die einzelnen Schritte der Suche manuell nachvollziehen zu können.

Die Konferenzen wurden prinzipiell in der Reihenfolge untersucht, wie sie bei Google in der Ergebnisliste erschienen. Beinhaltete eine Konferenz-Site Links zu vergangenen oder zukünftigen Veranstaltungen, so wurde diesen gefolgt und die entsprechenden Seiten, sofern sie obige Bedingungen erfüllten, ebenfalls hinzugefügt. Wie bereits erwähnt, spielt es in den meisten Fällen keine Rolle, wenn mehrere Jahrgänge derselben Konferenz betrachtet werden, da deren Aufbau sich i.d.R. nicht gleicht. Eine Verfälschung des Ergebnisses ist daher durch diese Vorgehensweise nicht zu befürchten.

[1] Genauer gesagt an die deutsche Version unter `http://www.google.de`

9.2.2 Beobachtungen der Studie

Dieser Abschnitt liefert eine Zusammenfassung der Beobachtungen, die bei der Studie obiger Konferenzseiten gemacht wurden, sowie entsprechende hieraus resultierende Überlegungen bzgl. der Extraktionssoftware.

Allgemeine Betrachtung Insgesamt wurden auf die im vorherigen Abschnitt beschriebene Weise Testdaten von 53 unterschiedlichen Konferenzen gesammelt, d.h. jede Konferenz ist durchschnittlich mit zwei Jahrgängen enthalten. Tatsächlich sind einige Konferenzen jedoch erheblich öfter vertreten (z.B. *Cluster* mit 8 Jahrgängen), während insgesamt 29 Konferenzen nur mit einem einzigen Jahrgang vertreten sind. Wie zuvor erklärt, spielt es für unsere Studie jedoch keine Rolle, ob die Daten verschiedenen Konferenzen entstammen.

Interessanter ist eine Betrachtung der Jahrgänge. Knapp die Hälfte der untersuchten Seiten entstammt den Jahren 2007 bis 2009; je weiter der Jahrgang zurückliegt, desto weniger Websites konnten gefunden werden. Die älteste betrachtete Seite stammt aus dem Jahre 1998. Abbildung 9.3 zeigt die genaue Verteilung der Jahrgänge. Die Auswahl der Testdaten deckt also zum einen

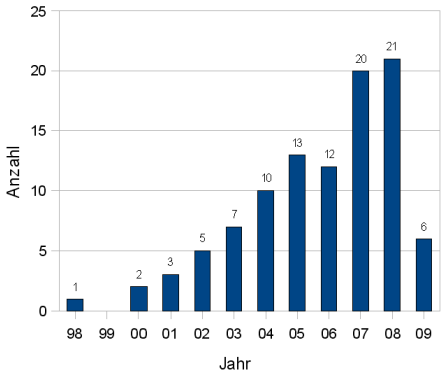


Abb. 9.3: Verteilung der Jahrgänge der untersuchten Konferenzseiten
Quelle: Eigene Erstellung

eine ausreichend große Zeitspanne ab, berücksichtigt hierbei aber hauptsächlich neue Websites, was uns demnach eine entsprechende Aktualität der Ergebnisse verspricht.

Die meisten Konferenzprogramme sind auf einer einzigen Webseite untergebracht. In nur 6 % der untersuchten Fälle waren die bibliographischen Daten auf mehrere Seite verteilt vorzufinden. In einem dieser Fälle ([55]) sind die Daten auf über 200 Seiten verteilt; hier wurde jeder Konferenz-Session eine einzelne HTML-Seite gewidmet. In den anderen Fällen wurde meist je Konferenztag eine einzelne Seite gewählt, wodurch die Daten auf maximal drei Seiten verteilt vorliegen.

45 % der Seiten geben vor, entsprechend der 'HTML 4.01 Transitional'-DTD erstellt worden zu sein, 26 % behaupten selbiges bzgl. 'xhtml 1.0 Transitional'. Eine genaue Übersicht hierüber bietet die linke Seite von Abbildung 9.4.

Auch hier wurde eine Validierung unter `http://validator.w3.org` durchgeführt, und wider Erwarten stellte sich eine der Seiten ([68]) tatsächlich als *valide* heraus. Die übrigen Seiten produzierten zwischen fünf und mehreren hundert Fehlern.

Einige der Seiten lieferten in einem alternativen Metadaten-Attribut (`<meta>`) Informationen über deren 'Generator', d.h. die Software, mit welcher sie erstellt wurden. Wie das rechte Diagramm in Abbildung 9.4 zeigt, konnte hier in knapp der Hälfte der Fälle ein Eintrag gefunden werden. Mehr als ein Drittel aller Seiten wurden unter Zuhilfenahme von Microsoft Frontpage, Word oder Excel entworfen.

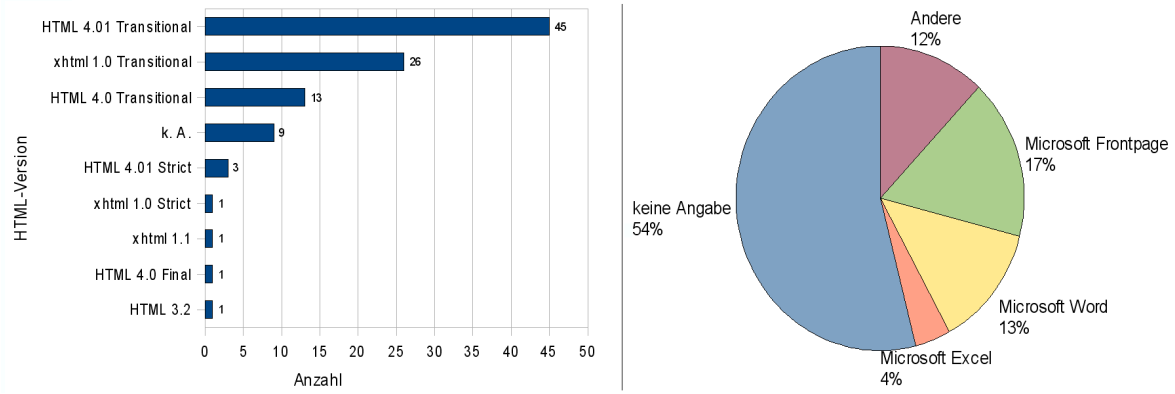


Abb. 9.4: HTML-Versionen und zur Erstellung benutzte Software
Quelle: Eigene Erstellung

Gerade jene Seiten weisen oftmals einen äußerst komplizierten Quelltext auf. Zum einen werden Formatierungen des Textes zumeist nicht in Form von CSS-Klassen, sondern direkt im Code mittels entsprechender `style`-Attribute vorgenommen. Dies macht den Text auch und vor allem für das menschliche Auge unleserlich. Hinzu kommen zahlreiche, jeweils nur ein einziges Wort umschließende Tags der Form '`<span class=SpellE>...</span>`', mit deren Hilfe Informationen zur Sprache in den Text eingeflochten sind. Oft treten auch die völlig nutzlosen (und invaliden) Tags '`<o:p></o:p>`' direkt hintereinander auf.

Informationsgehalt Die Fusion mit einem HTML-Konferenzprogramm soll zweierlei Nutzen bringen: Zum einen sollen Autorennamen, die nur abgekürzt vorliegen, ergänzt werden, und zum anderen sollen Zwischenüberschriften gefunden werden, um die vorhandenen Artikel besser untergliedern zu können. Voraussetzung hierfür ist natürlich, dass die entsprechenden Informationen auf den Konferenzseiten auch existieren.

Insgesamt wiesen 92% der untersuchten Seiten vollständige Autorennamen auf, 93% enthielten zusätzliche Informationen bzgl. der Konferenz-Sessions, welche als Zwischenüberschriften dienen können. Dies bestätigt den Eindruck, dass die Entwicklung einer Software, die diese Daten ausliest und mit den Daten einer BHT-Datei fusioniert, äußerst sinnvoll und arbeitserleichternd ist; bisher muss eine Ergänzung um die Daten einer solchen Konferenz-Homepage stets manuell erfolgen.

Technische Eigenschaften der Daten Neben den inhaltlichen Informationen interessieren natürlich vor allem technische Details. Diese sind der Hauptgrund zur Durchführung jener Studie, da wir erfahren möchten, wie die gesuchten Daten vorliegen, um daraus eine sinnvolle Strategie zur Extraktion eben jener Daten zu ersinnen.

Zunächst wurde die Reihenfolge der Artikel untersucht. In insgesamt 58% der Fälle stimmten die Reihenfolge bei Xplore und auf der Konferenzseite überein, in den restlichen Fällen waren die Daten entweder blockweise in der korrekten Reihenfolge (d.h. innerhalb einer Session) oder vollständig unsortiert. Wir werden bei der Extraktion also nicht davon ausgehen können, dass die Daten in gleicher Reihenfolge vorliegen und werden eine Strategie wählen, bei welcher die Reihenfolge keine Rolle spielt.

Die meisten Webseiten verfügen über für unsere Zwecke irrelevante Codeblöcke. Zumeist handelt es sich dabei um den HTML-Header, den Seitenheader (der z.B. ein Banner der Konferenz enthält) oder um ein oder mehrere Benutzermenüs, die die Navigation über die Seiten innerhalb des Browsers ermöglichen, seltener um Fußzeilen. Oftmals sind auch zusätzliche Tabellen auf der Seite enthalten, die eine zeitliche und/oder räumliche Übersicht der einzelnen Konferenzsessions bieten. Insgesamt enthielten 78% der untersuchten Seiten derartige irrelevante Blöcke vor dem eigentlichen Datenbereich, 30% der Seiten enthielten entsprechende Blöcke hinter den für uns interessanten Informationen. Es würde sich daher anbieten, einen HLRT-Wrapper (vgl. Kapitel 1.6.2) zu generieren, der zunächst jene irrelevanten Blöcke eliminiert und nur auf den wirklich nützlichen Daten operiert.

Da wir die einzelnen Artikel mit Hilfe der Titel identifizieren möchten, ist es wichtig zu untersuchen, in welcher Reihenfolge Titel und Autorennamen in den Konferenzprogrammen auftreten. Hier konnten in 89% der Fälle zunächst die Titel und anschließend die Autorennamen gefunden werden, bei 9% der Seiten verhielt es sich umgekehrt. Zwei Seiten (Testfälle [58] und [100]) enthielten verschiedene Blöcke, von denen in einigen die erstgenannte, in anderen die zweite Reihenfolge auftrat.

In 93% der Fälle befand sich mindestens ein HTML-Tag zwischen Titel und Autorennamen. In den restlichen Fällen fand die Trennung lediglich mittels Satzzeichen (Komma, Doppelpunkt), einer öffnende Klammer oder des Schlüsselwortes ‘by’ statt. Diese Fälle sollten zwar ebenfalls Beachtung finden, doch kann davon ausgegangen werden, dass Titel und Autoren i.d.R. durch Betrachtung der HTML-Elemente identifiziert und auseinander gehalten werden können. Bei 81% der Seiten waren die Strings, welche zwischen Titel und Autorennamen stehen, statisch, d.h. bei sämtlichen Artikeln gleich. Bei den übrigen 19% unterschieden sie sich, meist wegen zusätzlicher, variabler Informationen in jenem Zwischenbereich.

Bei einem Viertel der Seiten – vorzugsweise bei jenen, die mittels Software von Microsoft generiert waren – wurden innerhalb des Bereiches der Autorennamen HTML-Tags gefunden, bei etwas mehr als 10% der Seiten traf dies auch auf die Titel zu. Hauptsächlich handelt es sich hier um die o.g. Tags, die Informationen zur Sprache beinhalten. Jene müssen bei (bzw. vor) einer Extraktion in jedem Fall eliminiert werden.

Die Autorennamen sind in den meisten Fällen in der Reihenfolge ‘Vorname(n) Nachname’ angegeben (99%), in nur einem einzigen Fall ([85]) wurden sie in umgekehrter Reihenfolge ‘Nachname Vorname(n)’ gefunden. Einzelne Namen werden mittels unterschiedlicher Satzzeichen voneinander abgegrenzt, häufig werden hinter einem Namen zudem weitere Informationen (z.B. Stadt/Land, Unternehmen, Forschungsgruppe etc.) – meist in runden Klammern – angegeben, welche bei der Extraktion heraus gefiltert werden müssen.

Zwischenüberschriften sind meist zwischen speziellen HTML-Tags wie ausgezeichneten Überschriften-Tags (`<h2>`, `<h3>`,...) oder Tags zur textuellen Hervorhebung (`<u>`, `<b>`,...) untergebracht. Manchmal stehen sie jedoch auch in einfachen `<span>`-Tags, die über `style`- oder `class`-Attribute verfügen. Eine Identifizierung der Zwischenüberschriften wird in jedem Fall die größere Herausforderung an die Software darstellen. Hilfreich könnte hier die Tatsache sein, dass oftmals feste Schlüsselwörter gefunden werden können, die eine Zwischenüberschrift einleiten. Das Diagramm in Abbildung 9.5 zeigt das Vorkommen entsprechender Schlüsselwörter innerhalb der Testdaten. Bei einem Vorkommen von 53% könnten wir in Erwägung ziehen,

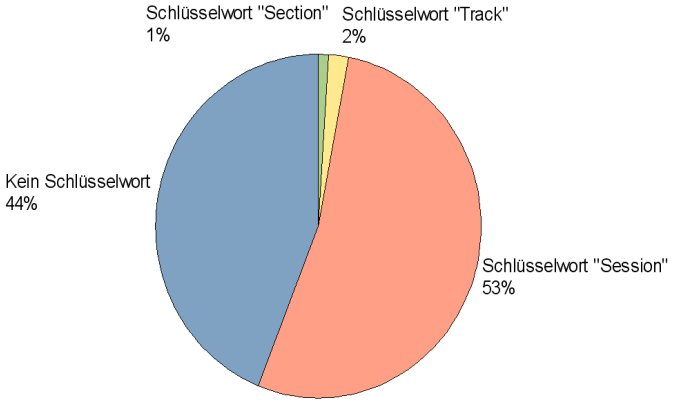


Abb. 9.5: Vorkommen fester Schlüsselwörter in Zwischenüberschriften
Quelle: Eigene Erstellung

Zwischenüberschriften anhand des Schlüsselwortes “Session” (oftmals auch “Paper Session”) zu identifizieren.

9.2.3 Besonderheiten einiger Konferenzseiten

Insgesamt haben wir im vorherigen Abschnitt eine Reihe mehr oder weniger eindeutiger Ergebnisse erhalten, die uns bei der Implementierung der Extraktionssoftware hilfreich sein werden. Ziel ist es, eine möglichst große Zahl der Seiten mit einheitlichen Mitteln bearbeiten zu können. Einige ganz spezielle Seiten stachen jedoch aus der Masse heraus. Bei diesen sind mit den allgemein sinnvollen Lösungen keine zufriedenstellenden Ergebnisse zu erzielen.

In drei Fällen wird eine Extraktion der Daten mittels Identifizierung der Titel- und Autorenpositionen nicht möglich sein, da hier die Tabellen strikt zeilenweise aufgebaut sind und ein

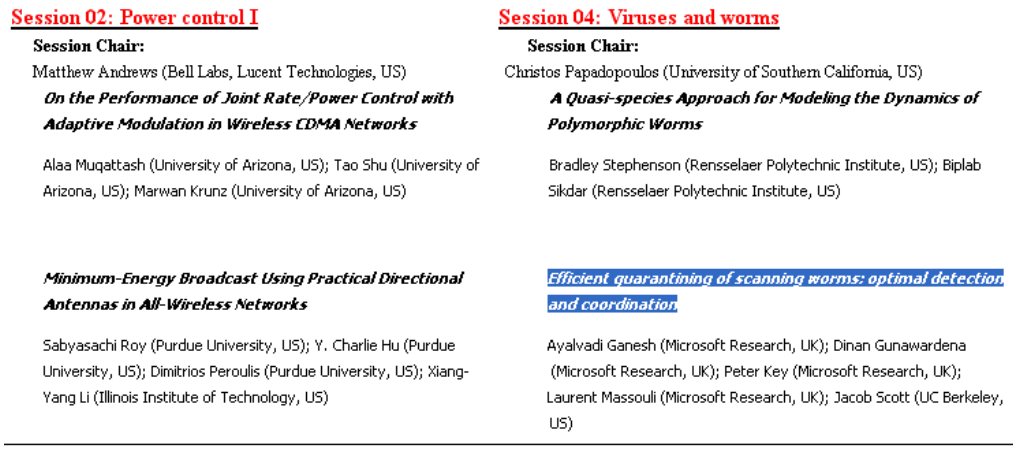
Titel, der sich über mehr als eine Zeile erstreckt, daher ‘zerstückelt’ wird. Abbildung 9.6 zeigt einen beispielhaften Ausschnitt der Website der INFOCOM 2006 ([23]). Im oberen Bereich erkennt man klar zusammenhängende Textblöcke (wie beispielsweise der markierte Titel), die im darunter abgebildeten Quellcode an entfernten Stellen auftauchen. Jener Quellcode ist zudem äußerst schwer leserlich; er wurde mittels ‘Microsoft Word 10’ generiert.

Eine weitere Besonderheit bringt [17] (ICIP 2008)[2] mit sich: Hier können einzelne Artikel mittels einer Suchmaske spezifiziert gesucht werden. Sendet man das Suchformular ohne Eingaben ab, so erscheint eine Sicherheitsabfrage, ob man tatsächlich die gesamten 808 Datensätze sehen möchte. Nach einer Bestätigung baut sich die Seite (recht langsam) auf und enthält somit alle gewünschten Informationen. Der Klick auf den Absenden-Knopf kann jedoch von unserer Software nicht automatisiert getätigt werden. Das Problem lässt sich jedoch dadurch lösen, dass zur Spezifizierung der sekundären Quelle neben der Angabe eines URLs auch die direkte Auswahl einer lokal gespeicherten HTML-Seite ermöglicht wird. Auf diese Weise kann man das Programm manuell im Browser erstellen lassen und anschließend abspeichern.

Das bereits oben in Abschnitt 9.2.2 beschriebene Problem, dass manche Konferenzsites die Programme nach Tagen sortiert anbieten, lässt sich natürlich beheben, indem die primäre Quelle nacheinander mit jeder dieser Seiten fusioniert wird. Im Falle der Einteilung nach Sessions müssten jedoch über 200 Seiten manuell eingegeben werden. Dies kommt der Arbeit, die Daten manuell zu erfassen, recht nahe, weshalb in diesem speziellen Fall ([55])[3] keine zufriedenstellenden Ergebnisse von der Software zu erwarten sind.

Ausgehend von den hier gemachten Beobachtungen werden wir im folgenden Kapitel Strategien entwickeln, mit welchen eine Extraktion der gewünschten Informationen möglich erscheint, und diese anschließend auf die 100 Testdatensätze anwenden (siehe Kapitel 10.1.5). Die Tabellen, welche zur Durchführung und Auswertung dieser Studie erstellt wurden, befinden sich in Anhang E.

[2] `http://www.icip08.org/Papers/AbstractSearch.asp?show=search`
[3] `http://www.icassp2008.com/RegularProgram.asp`



Session 02: Power control I

Session Chair:
Matthew Andrews (Bell Labs, Lucent Technologies, US)

On the Performance of Joint Rate/Power Control with Adaptive Modulation in Wireless CDMA Networks

Alaa Muqattash (University of Arizona, US); Tao Shu (University of Arizona, US); Marwan Krunz (University of Arizona, US)

Minimum-Energy Broadcast Using Practical Directional Antennas in All-Wireless Networks

Sabyasachi Roy (Purdue University, US); Y. Charlie Hu (Purdue University, US); Dimitrios Peroulis (Purdue University, US); Xiang-Yang Li (Illinois Institute of Technology, US)

Session 04: Viruses and worms

Session Chair:
Christos Papadopoulos (University of Southern California, US)

A Quasi-species Approach for Modeling the Dynamics of Polymorphic Worms

Bradley Stephenson (Rensselaer Polytechnic Institute, US); Biplab Sikdar (Rensselaer Polytechnic Institute, US)

Efficient quarantining of scanning worms: optimal detection and coordination

Ayalvadi Ganesh (Microsoft Research, UK); Dinan Gunawardena (Microsoft Research, UK); Peter Key (Microsoft Research, UK); Laurent Massouli (Microsoft Research, UK); Jacob Scott (UC Berkeley, US)

```
<p class=MsoNormal style='margin-top:34.2pt;mso-pagination:none;tab-stops:17.25pt 295.0pt;
mso-layout-grid-align:none;text-autospace:none'><span lang=EN-GB
style='font-family:Arial;mso-bidi-font-family:"Times New Roman";mso-ansi-language:
EN-GB'><span style='mso-tab-count:1'>      </span></span><b><i><span
lang=EN-GB style='font-size:8.0pt;font-family:Tahoma;color:black;mso-ansi-language:
EN-GB'>Minimum-Energy Broadcast Using Practical Directional </span></i></b><span
lang=EN-GB style='font-family:Arial;mso-bidi-font-family:"Times New Roman";
mso-ansi-language:EN-GB'><span style='mso-tab-count:1'>               </span></span><span
class=GramE><b><i><span lang=EN-GB style='font-size:8.0pt;font-family:Tahoma;
color:black;mso-ansi-language:EN-GB'>Efficient</span></i></b></span><b><i><span
lang=EN-GB style='font-size:8.0pt;font-family:Tahoma;color:black;mso-ansi-language:
EN-GB'> quarantining of scanning worms: optimal detection</span></i></b><b><i><span
lang=EN-GB style='font-size:11.0pt;font-family:Tahoma;color:black;mso-ansi-language:
EN-GB'><o:p></o:p></span></i></b></p>

<p class=MsoNormal style='mso-pagination:none;tab-stops:17.25pt 295.0pt;
mso-layout-grid-align:none;text-autospace:none'><span lang=EN-GB
style='font-family:Arial;mso-bidi-font-family:"Times New Roman";mso-ansi-language:
EN-GB'><span style='mso-tab-count:1'>      </span></span><b><i><span
lang=EN-GB style='font-size:8.0pt;font-family:Tahoma;color:black;mso-ansi-language:
EN-GB'>Antennas in All-Wireless Networks</span></i></b><span lang=EN-GB
style='font-family:Arial;mso-bidi-font-family:"Times New Roman";mso-ansi-language:
EN-GB'><span style='mso-tab-count:1'>                                    </span></span><b><i><span
lang=EN-GB style='font-size:8.0pt;font-family:Tahoma;color:black;mso-ansi-language:
EN-GB'> and coordination</span></i></b><b><i><span lang=EN-GB style='font-size:
9.5pt;font-family:Tahoma;color:black;mso-ansi-language:EN-GB'><o:p></o:p></span></i></b></p>

<p class=MsoNormal style='margin-top:7.8pt;mso-pagination:none;tab-stops:17.25pt 295.0pt;
mso-layout-grid-align:none;text-autospace:none'><span lang=EN-GB
style='font-family:Arial;mso-bidi-font-family:"Times New Roman";mso-ansi-language:
EN-GB'><span style='mso-tab-count:1'>      </span></span><span lang=EN-GB
style='font-size:8.0pt;font-family:Tahoma;color:black;mso-ansi-language:EN-GB'>Sabyasachi
Roy (Purdue University, US); Y. Charlie Hu (Purdue </span><span lang=EN-GB
style='font-family:Arial;mso-bidi-font-family:"Times New Roman";mso-ansi-language:
EN-GB'><span style='mso-tab-count:1'>                </span></span><span
lang=EN-GB style='font-size:8.0pt;font-family:Tahoma;color:black;mso-ansi-language:
EN-GB'>Ayalvadi Ganesh (Microsoft Research, UK); Dinan Gunawardena </span><span
lang=EN-GB style='font-size:11.0pt;font-family:Tahoma;color:black;mso-ansi-language:
EN-GB'><o:p></o:p></span></p>
```

Abb. 9.6: Beispiel eines 'zerstückelten' Tabellenaufbaus: Ausschnitt der Ansicht im Browser (oben) sowie des zugehörigen HTML-Quellcodes (unten). Die markierten Bereiche zeigen den gleichen Titel, der innerhalb des Quellcodes jedoch nicht zusammenhängend auftritt.
Quelle: `http://www.ieee-infocom.org/2006/technical_program.htm`

Kapitel 10

Fusion mit einer unstrukturierten Quelle

10.1 Fusion mit einem HTML-Konferenzprogramm

Nachdem wir in Kapitel 9 einen Eindruck der Problemstellung der Fusion einer strukturierten Quelle (d.h. in unserem Falle einer BHT_c oder BHT_{cite}-Datei) mit einer unstrukturierten HTML-Seite in Form eines Konferenzprogramms erhalten haben, soll nun die praktische Umsetzung innerhalb der Software erläutert werden. Wir werden uns hierbei auf die Gewinnung zusätzlicher Informationen bgzl. der Autorennamen beschränken und die Suche nach Zwischenüberschriften in Abschnitt 10.1.4 lediglich theoretisch erläutern. Wir werden sehen, warum sich die Suche nach Zwischenüberschriften erheblich schwieriger gestaltet und warum hier eine Lösung, die nur einen Teil der Daten hervorbringt, völlig indiskutabel ist.

Zur Extraktion der Daten aus der unstrukturierten Quelle benötigen wir zunächst eine geeignete Strategie. Hier gilt es, die aus der primären Quelle bekannten Informationen möglichst geschickt einzusetzen, um die entsprechenden Stellen der sekundären Quelle zu identifizieren, an welcher die gesuchten Informationen zu finden sind. Im Rahmen der vorliegenden Arbeit wurden drei derartige Strategien entwickelt, die in den Abschnitten 10.1.1, 10.1.2 und 10.1.3 erläutert werden.

Haben wir die Informationen extrahiert, so können wir den eigentlichen Fusionsvorgang gänzlich auf bereits bekannte Vorgänge zurückführen. Hierzu generieren wir wie zuvor bei der Informationsextraktion (siehe Kapitel 4.3.1) eine Record-Liste, die zunächst leer ist. Aus jedem extrahierten Datensatz werden wir sodann ein Record erzeugen und dieses jener Liste hinzufügen. Nach Abschluss des Extraktionsvorgangs liegen uns somit intern zwei Record-Listen vor: Eine enthält die Daten der primären Quelle, welche wir in gleicher Art wie zuvor bei der Fusion zweier strukturierter Quellen einlesen können, die andere enthält die neu extrahierten Daten der sekundären Quelle, die innerhalb der Liste nun ebenfalls in strukturierter Form vorliegen. Das Problem der Fusion reduziert sich demnach auf die Fusion zweier *strukturierter* Quellen, so dass wir sämtliche zuvor beschriebenen Algorithmen zur Partnersuche und Fusion einzelner Attribute in vollem Umfang anwenden können.

Es bleibt also zu zeigen, auf welche Weise die Daten extrahiert werden können. Da es, wie wir in Kapitel 9 gesehen haben, kaum zwei Konferenzseiten gibt, deren Aufbau gleich ist, ergibt hier eine manuelle Codierung der Extraktionsregeln keinen Sinn; die Software muss in der Lage sein, selbständig entsprechende Regeln zu generieren. Hierzu werden im Folgenden drei Strategien vorgestellt, welche auch in Java implementiert wurden. Wie zuvor wird auch hier von regulären Ausdrücken Gebrauch gemacht.

Da es das Ziel jener Strategien ist, die bestehenden Daten aufzubessern (engl. 'to enhance'), sollen jene Strategien auch als *Enhance-Strategien* bezeichnet werden. Die Klassen innerhalb der Software, in welchen die jeweiligen Strategien umgesetzt wurden, heißen entsprechend.

10.1.1 Enhance-Strategie 1: Simple Namenssuche

Eine recht einfach anmutende Strategie ist die folgende: Da uns aus der primären Quelle zumindest Teile eines jeden Autorennamens bekannt sind, nutzen wir diese, um innerhalb der HTML-Seite ähnliche Strings ausfindig zu machen. Wir verwenden also die Informationen der Autorennamen der primären Quelle, um entsprechende Namen innerhalb der sekundären Quelle zu finden. Dies soll das folgende Beispiel verdeutlichen:

Beispiel 10.1
In unserer primären Quelle laute der Name eines Autors "H. H. Hurtig" (= $aname_1$). In der sekundären Quelle wird der gleiche Autor mit vollständigem Vornamen angegeben: "Heinz Herbert Hurtig" (= $aname_2$). Unsere Aufgabe ist es nun, einen regulären Ausdruck unter Zuhilfenahme von $aname_1$ zu generieren, welcher in der Lage ist, $aname_2$ aufzuspüren. In diesem konkreten Fall ist das sehr einfach: Wir erwarten den Buchstaben des ersten Initials ("`H`") gefolgt von einer Reihe weiterer Buchstaben ("`\w+`") und eine anschließende Leerstelle ("`\s`"). Diese Kombination soll zweimal auftreten, da beide Vornamen mit dem gleichen Initial beginnen. Anschließend erwarten wir den uns bekannten Nachnamen ("`Hurtig`"). Der – zugegeben noch recht triviale – reguläre Ausdruck in Java[1] lautet demnach:

```
H\w+\sH\w+\sHurtig
```

Natürlich ist ein solcher Ausdruck wie in Beispiel 10.1 zu restriktiv. Mit dessen Hilfe kann zwar der im Beispiel genannte Name gefunden werden, doch müssen für eine allgemeine Suche eine Reihe weiterer Besonderheiten Beachtung finden. Die folgende Liste soll einen Überblick über die Schwierigkeiten bei der Generierung der Extraktionsregeln bieten.

Initiale Auch innerhalb der sekundären Quelle können Initiale auftreten, beispielsweise in "Heinz H. Hurtig". Die Extraktionsregel sollte in jedem Fall in der Lage sein, auch diese zu finden.

[1] Detaillierte Informationen zu regulären Ausdrücken in Java bietet beispielsweise `http://java.sun.com/docs/books/tutorial/essential/regex/`.

Zusätzliche Namensteile Der Name der sekundären Quelle kann zusätzliche Namensteile enthalten. Würde in obigem Beispiel der Name der ersten Quelle nur "H. Hurtig" lauten, so müsste sicher gestellt werden, dass eine Extraktion den vollständigen Namen ergäbe und nicht lediglich "Herbert Hurtig".

Fehlende Namensteile Natürlich kann es sich auch andersherum verhalten. Stünde in der sekundären Quelle lediglich "Heinz Hurtig", so müsste auch dieser Name gefunden werden.

Bindestriche Statt eines Leerzeichens könnte auch ein Bindestrich verwendet werden. Tritt dieser in der primären Quelle auf, z.B. in "H.-H. Hurtig", so müsste die Regel bei der Extraktion auch eine simple Leerstelle akzeptieren. Umgekehrt könnte jedoch auch in der sekundären Quelle ein Bindestrich auftreten ("Heinz-Herbert Hurtig"). Wir müssen also davon ausgehen, dass an *jeder* Position eines Leerzeichens auch ein Bindestrich vorkommen könnte.

Reihenfolge der Namensteile Wie wir in der Studie in Kapitel 9 gesehen haben, herrscht meist die Konvention, dass die Vornamen vor dem Nachnamen angezeigt werden. Dennoch sollten die Regeln auch in der Lage sein, die Form 'Nachname, Vorname(n)' korrekt zu erfassen.

Neben den genannten Fällen stellen auch hier die Entities ein besonderes Problem dar. Da die Daten der primären Quelle in normalisierter Form vorliegen, können hier lediglich die in der DBLP-DTD definierten Entities auftreten. Die sekundäre Quelle dagegen kann in einem nahezu beliebigen Zeichensatz codiert sein. Daher muss die gesamte HTML-Seite vor Beginn der Extraktion ebenfalls normalisiert werden. Hierzu werden sämtliche Zeichen, die *nicht* zu HTML-Tags gehören, ebenfalls entsprechend der DTD codiert. HTML-Tags werden unverändert belassen; diese werden wir in der Weiterentwicklung der Extraktionsstrategie (siehe Enhance-Strategie 3 in Abschnitt 10.1.3) in eben jener Form benötigen. Außerdem werden Zeilenumbrüche, ebenso wie mehrfach hintereinander auftretende Leerzeichen, durch ein einzelnes Leerzeichen ersetzt.

Treten Entities in der primären Quelle auf, so müssen wir davon ausgehen, dass sie in der sekundären Quelle fehlen könnten. Wir müssen also nach dem Entity oder einem alternativen Buchstaben suchen. Andererseits könnte es jedoch auch sein, dass in der sekundären Quelle ein Entity auftritt, welches in der primären Quelle fehlt, beispielsweise bei "M. Muller" und "Martin Müller". Gerade ein solcher Umlaut kann jedoch auch in einer alternativen Schreibweise dargestellt werden: "Mueller". Weiterhin können die Ligaturen "ß" und "æ" bzw. "Æ" auftreten, die ebenso mittels "ss" oder "ae" bzw. "AE" dargestellt werden können.[2] Unsere Regeln müssen daher in der Lage sein, all jene Formen zu finden, weshalb jeder auftretende Buchstabe "c" beispielsweise mittels "`(c|&c[^;]+;)`", jedes auftre-

[2]Weitere Ligaturen wie beispielsweise "œ" für das Zeichen "œ" können nicht auftreten, da diese nicht zum Latin-1 Zeichensatz gehören und daher in der DTD nicht definiert sind. Solche Zeichen wurden daher bereits bei der Normalisierung in eine alternative Darstellung umgewandelt, in obigem Falle in ein "oe".

tende “u” sogar durch “`(u|ue|&u[^;]+;)`“ codiert wird, um auch Entities wie “`&ccedil;`”, “`&uuml;`” etc. zu finden.[3]

Wir sehen, dass die eingangs recht simpel erscheinende Strategie bereits einige Überlegung erfordert, damit möglichst viele Namen gefunden werden können. Innerhalb der Software wird in mehreren Schritten vorgegangen, indem zunächst mittels restriktiver Regeln nach möglichst exakten Treffern gesucht wird; im Falle eines Misserfolges werden die entsprechenden Regeln dann etwas ‘aufgeweicht’, so dass alle möglichen oben genannten Sonderfälle gefunden werden können. Der reguläre Ausdruck, der alle beschriebenen Sonderfälle berücksichtigt, sieht für den in Beispiel 10.1 genannten Namen “H. H. Hurtig” dann wie folgt aus:[4]

```
(H(\.|(\w|\&[^;]+;)+)(\s*|-)((\w|\&[^;]+;)+\.?(\s*|-))?|H(\.|(\w|\&[^;]+;)+)){
1,}(\s*|-)((\w|\&[^;]+;)+\.?(\s*|-))?H(u|ue|\&u[^;]*;)r(t|\&t[^;]*;)(i|\&i[^;]
*;)g
```

Haben wir auf diese Weise die Namen der Autoren einer Publikation extrahiert, so erstellen wir ein neues Record und fügen dieses unserer Record-Liste hinzu. Diese wird, wie eingangs erläutert, nach Abschluss der Extraktion als sekundäre (strukturierte) Quelle zur Fusion mit den ohnehin strukturiert vorliegenden Daten der primären Quelle genutzt. Hierbei wird der in Kapitel 7.2 vorgestellte Partnersuche-Algorithmus zum Einsatz kommen, der daraufhin versuchen wird, Records beider Listen zu Paaren zusammenzufassen, welche anschließend fusioniert werden sollen. Da wir bei der Wahl obiger Strategie ohnehin die primäre Liste durchlaufen und gezielt nach den Autorennamen eines jeden Records suchen, wissen wir jedoch bereits, welchem Partner ein neu erstelltes Record zugeordnet werden soll. Wir reichern dieses daher geschickt mit weiteren Informationen an, welche wir aus dem primären Record entnehmen, damit der Algorithmus beide Records sofort als Partner erkennt:

Stellen wir uns vor, der mehrfach zitierte Autor “H. H. Hurtig” habe einen Artikel verfasst, dessen Abstract-Seite unter einem im EE-Attribut eingetragenen DOI oder URL online verfügbar sei. Da wir aus Abschnitt 7.1.1 wissen, dass mittels des Vergleichs zweier EE-Attribute direkt auf eine Ähnlichkeit der jeweiligen Records von `1.0` geschlossen werden kann, wenden wir hier einen kleinen Trick an. Wir fügen dem neu generierten Record R_2 nicht nur den aus der sekundären Quelle extrahierten Autorennamen ($authors_2$ =“Heinz Herbert Hurtig”) hinzu, sondern setzen zudem dessen EE-Attribut auf den Wert des EE-Attributs der primären Quelle ($ee_2 = ee_1$). Auf diese Weise werden beide Records bei der späteren Fusion sofort als Partner identifiziert, was zum einen eine mögliche Fehlerquelle (die fälschliche Zuordnung zweier Records) völlig ausschließt und zudem – da die Records beider Listen in gleicher Reihenfolge vorliegen – eine lineare Komplexität, wie sie in Beispiel 7.4 auf Seite 123 gezeigt wurde, bedingt.

[3] Wir nutzen hier aus, dass der Name jedes in der DBLP-DTD definierten Entities stets mit dem alternativen Buchstaben beginnt: **e**acute (é), **e**circ (ę), **e**grave (è) und **e**uml (ë) beispielsweise allesamt mit einem ‘e’.

[4] Auf eine Erläuterung des Ausdrucks soll an dieser Stelle verzichtet werden; das Beispiel soll lediglich dazu dienen, einen Eindruck von der Komplexität der Regeln zu erhalten.

Verfügt der Eintrag der primären Record-Liste nicht über ein EE-Attribut (ee_1 = *null*), so kann statt dessen auch das *key*-Attribut, welches im Falle, dass es sich bei der primären Quelle um eine BHT_{cite}-Datei handelt, gesetzt ist, verwendet werden. Ansonsten wird zumindest der Titel übernommen ($title_2$ = $title_1$), was ebenfalls eine eindeutige Zuordnung ermöglicht, die Zeitkomplexität jedoch erhöht, da hier keine Ergebnisse von `1.0` erzielt werden können und somit, wie in Kapitel 7.2 beschrieben, ein paarweiser Vergleich aller Records erfolgen muss.

10.1.2 Enhance-Strategie 2: Namenssuche mit Hilfe der Titel und Autorennamen

Die zuvor geschilderte Enhance-Strategie 1 liefert in der Praxis bereits recht ansehnliche Ergebnisse (vgl. Abschnitt 10.1.5), doch besitzt sie zwei entscheidende Nachteile. Zum einen werden sehr ähnliche Namen (beispielsweise "Abendrot" und "Abendbrot") nicht gefunden – ein Problem, welchem wir uns in Abschnitt 10.1.3 annehmen werden. Zum anderen birgt obige Methode jedoch eine große Gefahr der falschen Zuordnung von Namen, da die Informationen über einen Autorennamen stets ohne Betrachtung des Kontextes gewonnen werden, wie das folgende Beispiel verdeutlicht:

Beispiel 10.2
Stellen wir uns vor, unsere primäre Quelle enthielte 200 Records. Das 10. Record besäße dabei die Attribute

$$title_1^{10} = \text{"Neue Methoden der Software-Entwicklung" und}$$
$$authors_1^{10} = \text{"G. Schmidt, H. Wurst"}$$

(wobei die hochgestellte ^{10} lediglich auf die Position des Records innerhalb der primären Quelle hinweisen soll), während die Attribute der gleichen Typen im 192. Record

$$title_1^{192} = \text{"BiVis - ein neues Tool zur Visualisierung binärer Bäume" und}$$
$$authors_1^{192} = \text{"G. Schmidt, A. Abendrot"}$$

lauteten. Während man anhand jener Daten nicht entscheiden kann, ob es sich bei den erstgenannten Autoren "G. Schmidt" um ein und die selbe Person handelt, liefere die sekundäre Quelle in Form eines Konferenzprogramms eine klare Antwort auf jene Frage: Autoren des erstgenannten Artikels seien hier "Gustav Schmidt" und "Hans Wurst", während letzterer von "Gertrud Schmidt" und "Alfred Abendrot" verfasst sei.

Durchsuchen wir nun jene sekundäre Quelle mit Hilfe der Enhance-Strategie 1, so erhalten wir jedoch in beiden Fällen für den ersten Autoren das Ergebnis "Gustav Schmidt", da dies stets die erste Fundstelle unserer Extraktionsregel ist. Entgegen korrekter Angaben innerhalb des Konferenzprogramms können sich in einem solchen Sonderfall bei Anwendung der Enhance-Strategie 1 also derart üble Fehler ins Ergebnis einschleichen. Wir dürfen diese Strategie daher niemals direkt auf die gesamte Seite anwenden, können sie jedoch durch Einschränkung des Suchbereichs derart modifizieren, dass die Gefahr solcher Fehler extrem minimiert wird.

Stellen wir uns vor, in obigem Beispiel seien die Artikel des Konferenzprogramms in gleicher Reihenfolge angeordnet wie die Records der primären Quelle, so lägen die Informationen 181 anderer Artikel zwischen jenen der beiden genannten. Ziel ist es nun, die Stellen, an welchen sich die direkten Informationen zu einem einzelnen Artikel befinden, zu identifizieren, möglichst eng einzugrenzen und anschließend nur in diesen Bereichen mittels der Extraktionsregeln der Enhance-Strategie 1 zu suchen.

Um einen derartigen Suchbereich zu finden, nutzen wir weitere uns bekannte Informationen aus. So können wir beispielsweise fordern, dass die Namen *aller* Autoren eines Artikels in direkter Umgebung auftreten müssen. Suchen wir beispielsweise nach einer Stelle innerhalb des Konferenzprogramms, an welcher die Namen "Schmidt" und "Abendrot" in einem Abstand von nicht mehr als n Zeichen ($n \in \mathbb{N}$) auftreten, so würde bei Wahl eines geeigneten Wertes (beispielsweise $n = 100$) nur jener Bereich gefunden, in welchem beide Namen in unmittelbarem Kontext genannt würden. Damit erhöht sich die Chance, dass wir den Bereich der HTML-Seite finden, welcher die Daten des obigen 192. Artikels beinhaltet, gewaltig. Entsprechend würde bei der Extraktion nach Enhance-Strategie 1 innerhalb dieses Teilbereichs der korrekte Name "Gertrud Schmidt" gefunden.

Die genannte Vorgehensweise versagt natürlich, wenn ein Artikel über keine weiteren Autorenangaben verfügt. Besäße obiges 192. Record das Attribut

$$authors_1^{192} = \text{"G. Schmidt"},$$

welches lediglich den einen mehrdeutigen Namen enthielte, könnte jene Strategie nicht angewandt werden. Zudem könnte es auch sein, dass der Name "Abendrot" zufällig auch in der Nähe des 10. Artikels aufträte. Die Identifikation eines entsprechenden Bereichs nur mit Hilfe der Autorennamen ist daher noch immer mit zu vielen Fehlern behaftet.

Daher werden wir eine weitere Information nutzen, um einen solchen Bereich ausfindig zu machen: den Titel der Publikation. Wie wir im Beispiel sehen, unterscheiden sich $title_1^{10}$ und $title_1^{192}$ stark voneinander. Wenn wir also zusätzlich fordern, dass jene Titel in unmittelbarer Nähe – für deren Beschreibung wir ebenfalls obiges n nutzen können – der Autorennamen auftreten müssen, so können wir falsche Fundstellen mit großer Wahrscheinlichkeit ausschließen.

Doch müssen wir davon ausgehen, dass die Titel im Konferenzprogramm nicht exakt gleich lauten wie in unseren primären Records. Während der Studie der Konferenzseiten sind viele kleine Modifikationen von Titeln aufgefallen, hin und wieder treten auch stärkere Abweichungen auf. So lautet der Titel eines Artikels der Konferenz "HST '09" (Testdatensatz [14]) in Xplore beispielsweise "Evaluating drinking water early warning systems", während der entsprechende Partner innerhalb des Konferenzprogramms mit "Evaluation Methods for Water Distribution Network Early Warning Systems for Use on Military Bases" betitelt ist. Hier ist daher eine unscharfe Suche notwendig, die wir erreichen können, indem wir fordern, dass nicht *alle*, sondern lediglich eine ausreichend große Anzahl an *aussagekräftigen* Begriffen gefunden werden soll. Natürlich stellt sich hier sofort die Frage, welche Begriffe als 'aussagekräftig' bezeichnet werden können.

Es ist klar, dass Wörter, die in sehr vielen Titeln auftreten (beispielsweise Artikel, Präpositionen etc, aber auch im Kontext der Informatik oft verwendete Begriffe wie “database” oder “network”), hier nicht geeignet sind. Daher muss eine aus dem IR bekannte Eliminierung dieser so genannter Stoppwörter durchgeführt werden. Hierzu nutzt die Software eine einfache Heuristik: Zunächst werden die Titel aller derzeit in DBLP eingetragener Publikationen aus der Datei `dblp.xml` ausgelesen. Hieraus wird die Häufigkeit aller auftretenden Wörter gezählt, wobei die Wörter entsprechend auf ihre Stammform reduziert werden. Dies geschieht mittels des ebenfalls aus dem IR bekannten ‘Porter-Stemmers’ ([Por80]), eines Algorithmus’, der Wörter der englischen Sprache recht zuverlässig auf eine – wenn auch nicht immer grammatikalisch korrekte, dafür aber zum Vergleich nützliche – Grundform reduziert (beispielsweise “authors” zu “author”, “happy” und “happyness” zu “happi”). Für andere Sprachen wie das Deutsche ist er zwar prinzipiell weniger gut geeignet, doch da der Großteil der in DBLP erfassten Artikel in englischer Sprache verfasst ist, soll uns dies nicht weiter stören. Wurden die entsprechenden Häufigkeiten errechnet, so definieren wir die 2500 am häufigsten auftretenden Begriffe kurzerhand als Stoppwörter – wobei jener konkrete Wert nicht zwingend ist, jedoch bei manueller Kontrolle ein gutes Ergebnis zu liefern scheint. Auf diese Weise gelingt es uns, eine mehrsprachige, der direkten Praxis entnommene Liste von Stoppwörtern für den Bereich der Publikationen innerhalb der Informatik zu generieren, mittels derer wir die geforderten ‘aussagekräftigen’ Begriffe definieren können, welche im Folgenden als Schlüsselwörter eines Titels ($keywords_t$) bezeichnet werden.

Nehmen wir zu jenen Schlüsselwörtern nun auch noch die Namensteile der in Kapitel 8.3.3 definierten Typen PRENAME und SURNAME hinzu, die wir als $keywords_a$ bezeichnen werden, so erhalten wir eine Liste von Schlüsselwörtern ($keywords$), mittels derer die Identifikation eines Bereiches der HTML-Seite, welche die Daten des entsprechenden Artikels enthält, möglich ist. Für die in Beispiel 9.1 beschriebenen Artikel mit

$$\begin{aligned} title_1^{10} &= \text{“Neue Methoden der Software-Entwicklung”}, \\ authors_1^{10} &= \text{“G. Schmidt, H. Wurst”}, \\ title_1^{192} &= \text{“BiVis - ein neues Tool zur Visualisierung binärer Bäume” und} \\ authors_1^{192} &= \text{“G. Schmidt, A. Abendrot”} \end{aligned}$$

erhalten wir, wenn wir zudem die Groß- und Kleinschreibung ignorieren, demnach die Schlüsselwörter

$$\begin{aligned} keywords_t^{10} &= \text{“neue methoden software”}, \\ keywords_a^{10} &= \text{“schmidt wurst”}, \\ \Rightarrow keywords^{10} &= \text{“neue methoden software schmidt wurst”}, \end{aligned}$$

sowie

$$\begin{aligned} keywords_t^{192} &= \text{“bivis neues tool visualisierung”}, \\ keywords_a^{192} &= \text{“schmidt abendrot”}, \\ \Rightarrow keywords^{192} &= \text{“bivis neues tool visualisierung schmidt abendrot”}. \end{aligned}$$

Fordern wir nun, dass zumindest ein Teil jener Schlüsselwörter (beispielsweise mindestens aufgerundet $\frac{3}{4}$ von diesen) in unmittelbarer Nähe zueinander auftreten muss, so lassen sich auf diese Weise die entsprechenden Bereiche identifizieren, die dann mittels der ersten Enhance-Strategie weiterverarbeitet werden können.

Natürlich kann es hierbei vorkommen, dass Bereiche nicht gefunden werden, beispielsweise weil die Titel beider Artikel zu stark voneinander abweichen. Dieser Nachteil wird jedoch durch den immensen Vorteil aufgewogen, dass wir die bei der Enhance-Strategie 1 zu erwartenden gravierenden Fehler nun fast gänzlich ausschließen können.

10.1.3 Enhance-Strategie 3: Generierung der Regeln mittels Trainingsdaten

Beiden zuvor beschriebenen Strategien ist jedoch der bereits zuvor angesprochene Nachteil gemeinsam: Die nach obiger Beschreibung generierten Extraktionsregeln liefern kein Ergebnis, wenn die Schreibweisen eines Namens auch nur minimal voneinander abweichen, wie beispielsweise bei “Alfred Abendrot” und “Alfred Abendbrot”, “Schulze” und “Schultze” oder auch “Mayer” und “Maier”. All jene Namen haben lediglich eine Levenshtein-Distanz von 1 und würden bei der Fusion von unserer Ähnlichkeitsfunktion in jedem Fall erkannt werden. Mittels der zuvor definierten regulären Ausdrücke ist es uns jedoch nicht möglich, diese zu finden.

Um dies zu ändern, könnten Methoden des ‘approximate string matching’ wie beispielsweise der ‘Baeza-Yates-Gonnet-Algorithmus’ ([BYG92], auch bekannt als ‘Shift-or’ bzw. ‘Shift-and’-Algorithmus) verwandt werden, um eine unscharfe Suche nach jenen Namensteilen zu ermöglichen. Derartige Methoden wurden im Rahmen der erstellten Software jedoch nicht implementiert.

Eine andere Möglichkeit besteht darin, die Idee lernender Wrapper, wie sie in Kapitel 1.6.1 beschrieben wurde, in einer einfachen Form auf die konkrete Problemstellung anzuwenden. Dort wird u.a. der Ansatz von Arasu und Garcia-Molina beschrieben, in welchem die Tatsache, dass mehrere Webseiten mittels gleicher Templates erstellt wurden, zur automatischen Generierung von Extraktionsregeln genutzt wird ([AGM03]). Ein ähnlicher Versuch wurde ansatzweise in der 3. Enhance-Strategie umgesetzt.

Viele der HTML-Seiten wurden mittels ‘Generatoren’, d.h. mit Hilfe diverser Software erstellt. Da es sich bei den die Daten enthaltenden Programmblöcken stets um lange Listen oder Tabellen handelt, die i.d.R. ein gleiches Aussehen haben, kann davon ausgegangen werden, dass auch der HTML-Quellcode an dieser Stelle gleich oder zumindest ähnlich ist. Selbst bei manuell konstruierten Seiten kann davon ausgegangen werden, dass ähnliche Blöcke nicht stets neu eingegeben, sondern mittels ‘copy-and-paste’ vervielfältigt und anschließend angepasst wurden. Betrachten wir hierzu das folgende Beispiel:

Beispiel 10.3
Gegeben sei der folgende Ausschnitt des HTML-Quellcode einer Konferenz-Webseite.

```
<tr>
    <td width="120">
        <strong>Neue Methoden der Software-Entwicklung</strong><br />
        <em>Gustav Schmidt, Hans Wurst</em>
    </td>
</tr>
<tr>
    <td width="120">
        <strong>Konstruktion schneller Extraktionsverfahren</strong><br />
        <em>Horst Herbert Hurtig</em>
    </td>
</tr>
<tr>
    <td width="120">
```

Wir sehen dass die einzelnen Datensätze von jeweils gleichen HTML-Tags umschlossen werden, die wir als Template im Sinne von Arasu und Garcia-Molina auffassen können:

```
<tr>
    <td width="120">
        <strong>{TITLE}</strong><br />
        <em>{AUTHORS}</em>
    </td>
</tr>
```

Um ein solches Template automatisch zu konstruieren, gehen wir wie folgt vor: Zunächst identifizieren wir die Positionen der Titel und Autorennamen einiger Records ähnlich der Enhance-Strategie 2. Anschließend definieren wir einen Bereich des HTML-Quellcodes, der eben jene Daten enthält, indem wir beispielsweise vor und hinter den gefundenen Begriffen noch jeweils n Zeichen übernehmen. Auf diese Weise erhalten wir ein HTML-Codefragment, welches wir als "Trainingsdatensatz" bezeichnen wollen. Innerhalb dieses Datensatzes wissen wir, wo sich Titel und Namen befinden, da wir diese eindeutig identifizieren konnten.

Setzen wir im Beispiel einen Wert von $n = 20$ voraus (in der Praxis liegt er bei 200, da wir annehmen müssen, dass der Quelltext gerade von mittels Microsoft-Produkten erstellter Webseiten erheblich mehr Tags und somit mehr Zeichen enthält) und nehmen wir an, wir hätten im ersten Eintrag des obigen Beispiel-Quellcodes alle im vorherigen Abschnitt berechneten Schlüsselwörter gefunden, so ergibt sich – unter Eliminierung der Zeilenumbrüche – der Trainingsdatensatz

```
width="120"><strong>Neue Methoden der Software-Entwicklung</strong><
br /><em>Gustav Schmidt, Hans Wurst</em></td></tr><tr><
```

da wir 20 Zeichen vor dem zuerst auftretenden Schlüsselwort (“Neue”) sowie 20 Zeichen hinter dem zuletzt gefundenen Schlüsselwort (“Wurst”) mit übernommen haben. Entsprechend ergäbe sich für den zweiten Artikel der Trainingsdatensatz

```
width="120"><strong>Konstruktion schneller Extraktionsverfahren</str
ong><br /><em>Horst Herbert Hurtig</em></td></tr><
```

Wir sehen, dass beide Datensätze von den gleichen HTML-Codefragmenten

```
width="120"><strong>
```

und

```
</em></td></tr><
```

umschlossen sind, und dass zwischen Titel und Autoren ebenfalls der gleiche HTML-Code

```
</strong><br /><em>
```

zu finden ist. Daher können wir die Extraktionsregel

```
width=\"120\"><strong>(.*?)</strong><br /><em>(.*?)</em></td></tr><tr><
```

formulieren, die uns, angewandt auf obiges Codefragment, beide Datensätze liefern würde; die erste der sogenannten ‘Capturing Groups’ (der in Klammern stehende Ausdruck) liefert uns hierbei den Titel des Artikels, die zweite dessen Autoren.

Im allgemeinen Fall werden wir zehn derartige Trainingsdatensätze zu gewinnen versuchen und daraus, soweit möglich, eine entsprechende Extraktionsregel generieren. Gelingt uns dies, so wenden wir eben jene Extraktionsregel anschließend auf die gesamte HTML-Seite an und erhalten somit im günstigsten Fall sämtliche verfügbaren Datensätze. Diese transformieren wir wiederum in einzelne Records, die wir der Ergebnis-Liste hinzufügen und nach Ablauf des Extraktionsvorgangs zur Fusion mit der strukturierten Quelle verwenden können. In diesem Fall werden nun auch ähnliche Namen erkannt, da der Partnersuche-Algorithmus hierzu in der Lage ist.

Natürlich ist die Konstruktion der Regeln auch hier in der Praxis bei weitem umfangreicher als in obigem Beispiel 10.3. In den Bereichen vor, hinter oder zwischen Titel und Autorennamen befinden sich oftmals weitere variable Daten wie Uhrzeit, Veranstaltungsort oder Herkunft der Autoren. Letzteres (also Land, Stadt, Organisation, Universität etc.) ist oftmals auch direkt hinter den jeweiligen Autorennamen zu finden. Damit ergeben sich Blöcke der Form

```
<td width="120">
    12:00<br />
    <strong>Neue Methoden der Software-Entwicklung</strong><br />
    <em>Gustav Schmidt (University of Buxtehude),
        Hans Wurst (Microsaft GmbH)</em><br />
    Room B-102
</td>
```

Die Generierung allgemeingültiger Regeln ist hier offensichtlich erheblich schwieriger. Wurde die HTML-Seite zudem mit einem Generator erstellt, der zahlreiche unnötige Tags einfügte (beispielsweise die typischen leeren Tags `<o:p></o:p>`, die von Microsoft Produkten erstellt werden), so verkompliziert sich die Erstellung einer solchen Regel noch weiter; bekannte überflüssige Tags werden daher bei der Normalisierung der HTML-Seite entfernt. Die oftmals vorherrschende Unsitte, CSS-Definitionen in Form von `style`-Attributen direkt in die entsprechenden Tags einzufügen, wie beispielsweise in

```
<span style="text-indent:12.45pt;font-size:10.0pt;font-family:Arial">
```

kann sowohl problematisch als auch hilfreich sein – je nachdem, ob diese Angaben je nach Datenblock variieren oder gleich sind.

Leider ist es aus all diesen Gründen nicht in jedem Fall möglich, eine Extraktionsregel zu konstruieren. Unterscheiden sich die Datenblöcke innerhalb des Quellcodes zu sehr, so werden u.U. keine identischen Bereiche vor und hinter den bekannten Daten gefunden. Ebenso schlägt diese Strategie völlig fehl, wenn nicht genügend Fundstellen der *keywords* ausgemacht und somit nicht genug Trainingsdaten gewonnen werden können. Ob diese Strategie Erfolg verspricht, hängt in hohem Maße von der Beschaffenheit des HTML-Quellcodes ab.

Weitere Ergänzungen jener Strategie in Richtung lernender Wrapper wären daher durchaus denkbar. Mit den an dieser Stelle beschriebenen und in der Software umgesetzten Enhance-Strategien soll daher lediglich der Grundstein für weitere Forschung auf diesem Gebiet gelegt werden.

10.1.4 Probleme bei der Extraktion von Zwischenüberschriften

Wie eingangs erwähnt, stellt auch die Gewinnung von Zwischenüberschriften aus HTML-Konferenzprogrammen eine sinnvolle Anwendung der hier beschriebenen Verbindung zwischen Extraktion und Fusion dar. In der Studie der Konferenzprogramme in Kapitel 9 wurden jene Zwischenüberschriften daher ebenfalls gesondert berücksichtigt. Leider bringt die Identifikation jener Daten im Gegensatz zur Suche nach vollständigen Autorennamen eine Reihe erheblicher Schwierigkeiten mit sich.

Zum einen ist es erheblich schwieriger, entsprechende Bereiche ausfindig zu machen, die Zwischenüberschriften enthalten. Oftmals weist uns das Schlüsselwort “Session” auf eine solche Überschrift hin, und mit dessen Hilfe könnte versucht werden, entsprechend der vorgestellten Enhance-Strategie 3 auch für die Zwischenüberschriften eine Extraktionsregel zu generieren, was innerhalb der Software auch in rein experimenteller Form umgesetzt wurde. Fehlt dieses Schlüsselwort, so könnte man versuchen, den Bereich mittels einer Betrachtung des HTML-Codes auf Regelmäßigkeiten hin zu untersuchen und z.B. spezielle Tags zu identifizieren, die hin und wieder zwischen den einzelnen Artikeln auftauchen – aber eben nicht immer. Gerade die HTML-Tags zur Definition von Überschriften (`<h1>`, `<h2>`, ···, `<h5>`) werden oftmals zur Darstellung von Zwischenüberschriften verwendet. Hier könnte eine konsequente Anwendung lernender Wrapper Erfolg versprechen und stellt sicherlich eine Herausforderung zur weiteren Forschung dar.

Problematisch ist jedoch oftmals auch die korrekte Zuordnung gefundener Zwischenüberschriften zu den entsprechenden Artikeln. Liegt das Konferenzprogramm in Form einer Liste vor, so ist dies recht einfach möglich: Wurde eine Überschrift gefunden, so wird diese allen nachfolgenden Artikeln zugeordnet – so lange, bis die nächste Überschrift gefunden wird. In einigen Fällen, in denen das Konferenzprogramm als Tabelle vorliegt, ist dies jedoch nicht auf diese Art und Weise möglich. Abbildung 10.1 zeigt den Ausschnitt eines in Form einer Tabelle vorliegenden Konferenzprogramms (Testdatensatz [9]). Für den menschlichen Betrachter ist

	Session 1A (Salon E) Optimizing with MPI *Chair: Toni Cortes*	Session 1B (Salon ABC) Scheduling I *Chair: Box Leangsukun*
10:30am - 12:00pm	TCP Adaptation for MPI on Long-and-Fat Networks *Motohiko Matsuda, Tomohiro Kudoh, Yuetsu Kodama, Ryousei Takano, Yutaka Ishikawa* Extracting Critical Path Graphs from MPI Applications *Martin Schulz* Accelerating List Management for MPI *Keith D. Underwood, Arun Rodrigues, K. Scott Hemmert*	Search-based Job Scheduling for Parallel Computer Workloads *Sangsuree Vasupongayya, Su-Hui Chiang, and Bart Massey* A Case for Cooperative and Incentive-Based Coupling of Distributed Clusters *Rajiv Ranjan, Rajkumar Buyya and Aaron Harwood* Transparent Networked Checkpoint-Restart for Commodity Clusters *Oren Laadan, Dan Phung, Jason Nieh*

Abb. 10.1: Ausschnitt eines tabellarischen Konferenzprogramms: Die Zwischenüberschriften beziehen sich jeweils auf die darunter liegende Tabellenzelle.
Quelle: `http://cec2008.cs.georgetown.edu/schedule.html`

die Zuordnung der Zwischenüberschriften (“Optimizing with MPI” und “Scheduling I”) zu den darunter stehenden Artikeln kein Problem, doch anhand der dicken Randlinien lässt sich erkennen, dass Überschriften und Datenblöcke in verschiedenen Zellen einer HTML-Tabelle untergebracht sind. Da sich eine solche Tabelle im HTML-Quelltext zeilenweise aufbaut, finden wir hier zunächst zwei Zwischenüberschriften in direkter Folge und anschließend die beiden Zellen mit den Daten beider Sessions. Hier müsste eine Interpretation der Tabelle innerhalb des HTML-Quellcodes stattfinden, um die Sessions korrekt zuordnen zu können.

Die experimentelle Routine innerhalb der Software wurde gemeinsam mit der dritten Enhance-Strategie implementiert. Hier gelingt es in einigen Fällen, zumindest einen Teil der Zwischenüberschriften korrekt zu erfassen, indem ebenfalls eine Regel zur Extraktion der Überschriften generiert wird. Anschließend werden die Positionen der gefundenen Artikel innerhalb

der kompletten Seite mit den Positionen der Zwischenüberschriften vergleichen, wodurch bei einem listenartigen Aufbau eine Zuordnung der Artikel zu den Überschriften möglich ist.

Solange jedoch nicht mit großer Sicherheit *alle* auf der HTML-Konferenzseite enthaltenen Zwischenüberschriften erfasst werden können, ist von der Anwendung einer solchen Strategie abzuraten; hier gilt die Devise 'alles oder nichts', da die Zwischenüberschriften in direkter Abhängigkeit zu den übrigen Daten stehen. Gelingt es uns, nur jeden zweiten Autorennamen korrekt aufzuspüren, so ist das Ergebnis zwar nicht vollständig, aber dennoch richtig. Finden wir dagegen nur jede zweite Zwischenüberschrift, so ist das Ergebnis falsch: Ein Teil der Artikel wird einer falschen Section zugeordnet.

10.1.5 Ergebnisse der Enhance-Strategien

Alle drei zuvor beschriebenen Strategien wurden nach deren Implementierung in die der Arbeit beiliegende Software anhand der im vorherigen Kapitel beschriebenen Testdatensätze überprüft. Im Folgenden soll ein kurzer Überblick über die hierbei gewonnenen Erkenntnisse gegeben werden. Die Tabellen, die jener Studie zu Grunde liegen, sind in Anhang E zu finden.

Abbildung 10.2 zeigt die Ergebnisse der Anwendung der drei Enhancer-Strategien auf die Testdaten. Die Balken geben jeweils an, wie viel Prozent der Autorennamen, die in den jeweiligen BHT-Dateien der Testdatensätze von Xplore vorhanden waren, mittels der jeweiligen Strategie gefunden wurden. Dabei konnte keine Überprüfung der Korrektheit jener Daten durchgeführt werden, da hierzu eine manuelle Kontrolle notwendig gewesen wäre – was bei über 42.000 Autorennamen, die innerhalb der Testdatensätze zu finden sind, in einem sinnvollen Umfang schier unmöglich ist. Es muss jedoch davon ausgegangen werden, dass die Daten, welche mittels der Enhance-Strategie 1 gefunden wurden, aus den oben genannten Gründen diverse Fehler enthalten könnten.

Vielmehr dienen uns die Ergebnisse der Extraktion mittels Strategie 1 als Grundlage, die Güte der übrigen Enhance-Strategien zu messen (und auch als Messlatte für weitere Entwicklungen) bzw. um überhaupt festzustellen, ob eine Extraktion grundsätzlich möglich ist. Wir sehen, dass in 7 Fällen keine der Strategien erfolgreich war (Testfälle [14], [16], [18], [30], [33], [89] und [91]). Ein Blick auf die Struktur jener Seiten lässt jedoch keine Rückschlüsse auf eventuelle Schwierigkeiten ziehen. Es handelt sich bei einigen dieser Webseiten, nicht aber bei allen, um solche, die zuvor als schwierig eingestuft wurden, da sie große Mengen an Tags enthalten, die von Generatoren wie Microsoft Word oder Frontpage herrühren. Da jedoch viele jener Seiten – insbesondere der am Ende des vorherigen Kapitels als schwierige eingestufte Datensatz [23] – gute Ergebnisse lieferten (vgl. die Tabellen in Anhang E), müssen wir davon ausgehen, dass hier andere Gründe vorliegen, die wir nicht genau benennen können. Hierzu wäre eine eingehende Untersuchung der Testdaten nötig, auf welche an dieser Stelle verzichtet werden soll. Man erkennt jedoch auch, dass in einigen Fällen äußerst gute Ergebnisse erzielt werden konnten; in Testdatensatz [54] wurden beispielsweise über 91% der Namen gefunden, der Durchschnitt aller Datensätze liegt bei 47,82%.

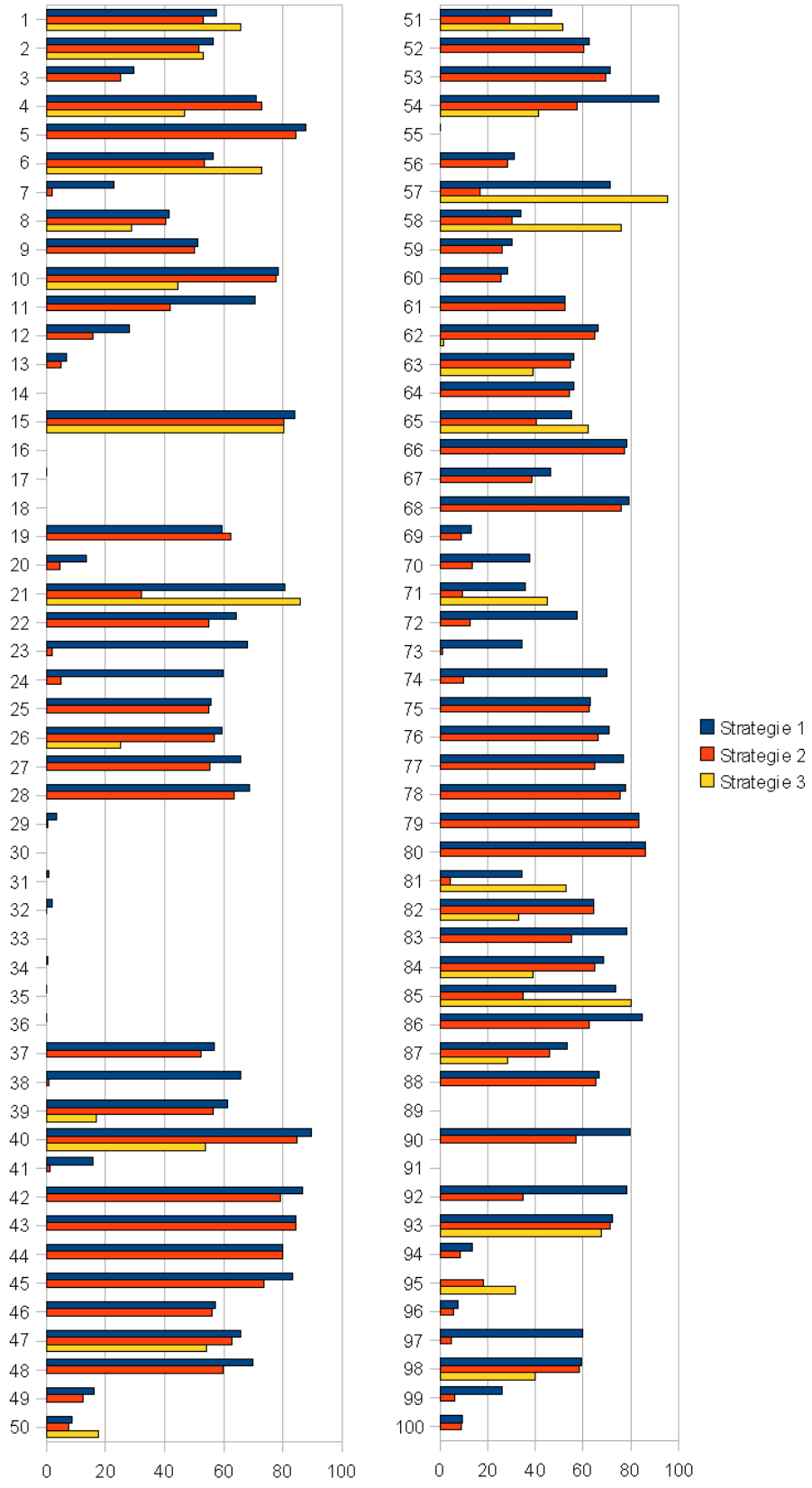


Abb. 10.2: Ergebnis der Anwendung der Enhancer-Strategien auf die Testdaten.
Quelle: eigene Erstellung

Vergleicht man nun die Ergebnisse der Strategien 1 und 2 miteinander, so fällt auf, dass Strategie 2 oftmals knapp an die Werte der ersten Strategie herankommt und in einigen Fällen sogar die gleiche Anzahl an Autorennamen findet ([79] und [80]); in [4] sogar einige mehr. In anderen Fällen werden nur äußerst wenige Autorennamen gefunden, was darauf hinweist, dass hier keine entsprechenden Blöcke identifiziert werden können. Dies ist oftmals bei obig genannten Webseiten der Fall, die viele zusätzliche HTML-Tags enthalten, wodurch sich der Abstand der Schlüsselwörter zueinander stark erhöht, so dass diese nicht mehr einander zugeordnet werden können. Hier könnte eine Erhöhung des Abstandes u.U. bessere Ergebnisse erzielen; denkbar wäre auch, die Software nach einer Voruntersuchung der Seite selbst einen geeigneten Abstandswert errechnen zu lassen. Fälle wie [85], in denen mittels der zweiten Strategie etwa halb so viele Treffer erzielt wurden, sind dagegen äußerst selten. Ein solcher Fall könnte darauf hinweisen, dass hier die Abstände der Schlüsselwörter in der Nähe des Grenzwertes liegen, weshalb es eher von Glück abhängig ist, ob eine Stelle noch gefunden wird oder nicht.

Strategie 3 schlägt, wie man unschwer erkennen kann, in vielen Fällen fehl. Insgesamt konnten nur in 29 Testdatensätzen Ergebnisse erzielt werden, was zeigt, dass diese Strategie noch stark verbesserungswürdig ist. Zumeist scheitert sie daran, dass nicht genügend einheitliche Trainingsdatensätze gefunden werden können, um eine Extraktionsregel zu generieren. Dennoch erkennen wir, dass mit dieser Strategie, sofern denn eine solche Regel gefunden wird, in vielen Fällen bessere Ergebnisse erzielt werden als mit Strategie 1. Hier spielen die oben genannten alternativen Schreibweisen bzw. Schreibfehler eine Rolle, die eine Extraktion nach den Strategien 1 und 2 verhindern, bei der Fusion jedoch keine Probleme verursachen. In Datensatz [71] erhalten wir so beispielsweise 251 Namen mehr als mittels der ersten Strategie.

Neben den Schwächen der bisherigen Methoden zeigt uns dieses Ergebnis jedoch auch, dass es in jedem Fall sinnvoll ist, verschiedene Strategien zu erproben und anzuwenden, da es hier sicher keine ‘Allround-Lösung’ geben wird. Diese Studie soll zu weiterer Forschung auf jenem Gebiet ermutigen.

10.2 Fusion mit einem Konferenzprogramm in PDF-Format

Nicht immer liegen die Konferenzprogramme in HTML-Format vor. Oftmals enthalten die Webseiten nur eine knappe Übersicht über den Programmablauf, bieten dafür aber das komplette Programm zum Download in einem anderen Format an – meist als PDF-, selten auch als DOC-Dateien. Dies hat den Vorteil, dass deren Aussehen klar definiert werden kann und i.d.R. auf allen Systemen gleich ist, während das Aussehen von HTML-Seiten in unterschiedlichen Browsern und auf unterschiedlichen Systemen u.U. variiert. Oftmals beinhalten jene Konferenzprogramme auch zahlreiche weiterführende Informationen; viele enthalten neben dem technischen Programm auch Informationen zu Unterkunft, Verpflegung und/oder lokalen Sehenswürdigkeiten. Neben dem reinen Konferenzverlauf sind in vielen Fällen ganze Abstracts der Artikel sowie Informationen und Biographien der Autoren zu finden. Wir haben es hier also meist mit großen Textmengen, die Bilder, Tabellen und große, irrelevante Textblöcke enthalten,

zu tun. Wir werden uns lediglich den PDF-Dateien widmen und exemplarisch untersuchen, ob die zuvor entwickelten Strategien auf jene anwendbar sind. DOC-Dateien werden wir aufgrund ihres seltenen Auftretens nicht behandeln.

Für die Suche nach Informationen sollen an dieser Stelle keine weiteren Algorithmen entwickelt werden. In der Software wurde jedoch die Möglichkeit umgesetzt, eine PDF-Datei als zweite Quelle anzugeben. Jene kann nach obigen Enhance-Strategien 1 und 2 – wobei in jedem Fall eine Anwendung der zweiten Strategie empfohlen wird – dazu dienen, zusätzliche Informationen über Autorennamen zu gewinnen. Die dritte Enhance-Strategie setzt dagegen die Existenz von HTML-Tags voraus, um die Bereiche, in welchen sich Autorennamen und Titel befinden, zu identifizieren und ist daher für PDF-Dokumente nicht geeignet.

Größtes Problem stellt zunächst das Einlesen einer PDF-Datei dar. Hierzu nutzt die Software das externe Linux-Programm `pdftotext`[5], welches eine PDF-Datei in reinen Plaintext verwandelt. Leider sind die Ergebnisse gerade bei kompliziertem Aufbau (Tabellen, Bilder, mittels Tabulatoren angeordnete Texte) oftmals nur von minderer Qualität. Dennoch kann versucht werden, aus jenen Textdateien anschließend die jeweiligen Autorennamen auszulesen.

In der Software wurde die Möglichkeit implementiert, als sekundäre Quelle eine PDF-Datei anzugeben. Die Erstellung von Testdatensätzen und eine Durchführung einer ähnlichen Studie wie für die HTML-Programme blieb jedoch aus, da der Hauptaugenmerk der vorliegenden Arbeit auf der Gewinnung von Daten aus HTML-Seiten liegt. Daher werden wir uns abschließend einem weiteren, letzten Fall widmen: dem 'Google-Orakel'.

10.3 Ausblick: Fusion mit dem WWW – Das 'Google-Orakel'

In den vergangenen Kapiteln haben wir gesehen, dass es möglich ist, Daten einer strukturierten Quelle mit zusätzlichen Informationen aus einer vorgegebenen zweiten Quelle – in unserem Fall handelte es sich um Konferenzprogramme – anzureichern. Dies setzt natürlich immer die Existenz einer derartigen Quelle voraus. Doch auch wenn uns keine derartige Quelle bekannt ist, eine solche u.U. gar nicht existiert, so steht uns doch eine zweite, äußerst umfangreiche Datenquelle zur Verfügung: das WWW. Der Informationsgehalt des 'World Wide Web' erscheint nahezu unendlich, und es ist realistisch anzunehmen, dass irgendwo, in den Tiefen des WWW, die gesuchten Informationen verborgen liegen könnten.

Im Falle unseres konkreten Problems, der Suche nach vollständig(er)en Autorennamen, können die gesuchten Informationen in digitalen Bibliotheken, wie sie in Kapitel 3 vorgestellt wurden, verborgen liegen, sie können auf privaten Homepages der entsprechenden Autoren zu finden

[5] `http://linux.die.net/man/1/pdftotext`

sein, oder auf Seiten sozialer Netzwerke wie 'XING'[6], 'facebook'[7] o.ä. versteckt sein. Selbst aus E-Mail-Adressen lassen sich hin und wieder vollständige Namen gewinnen. Die Möglichkeiten, an welchen gesucht werden kann, sind nahezu unbegrenzt, doch eine fleißige Suche ist oftmals lohnenswert.

Zum Abschluss der vorliegenden Arbeit soll nun versucht werden, eine Fusion einer strukturierten Quelle mit dem WWW durchzuführen. Hierzu bedienen wir uns der Internet-Suchmaschine 'Google'[8] und wollen versuchen, mittels geschickter Anfragen die gewünschten Informationen zu erhalten. Machen wir uns dies an dem folgenden Beispiel klar:

Beispiel 10.4
Betrachten wir die Konferenz INFOCOM 2009 (Testdatensatz [26]). Extrahieren wir die Daten jener Konferenz mittels unseres Wrappers aus Xplore, so erhalten wir eine BHT_c-Datei, deren erster Eintrag

```
<li>V. Konda, J. Kaur:
RAPID: Shrinking the Congestion-Control Timescale.
1-9
<ee>http://dx.doi.org/10.1109/INFCOM.2009.5061900</ee>
```

lautet. Betrachten wir das entsprechende HTML-Konferenzprogramm, so erfahren wir die vollständigen Namen der beiden Autoren: "Vishnu Konda" und "Jasleen Kaur". Stellen wir uns nun jedoch vor, jenes Konferenzprogramm sei uns nicht bekannt. Wir möchten daher Google befragen und versuchen, die Autorennamen auf diese Weise zu erfahren.

Um dies zu erreichen, müssen wir eine möglichst geschickte Anfrage stellen. Hierzu stehen uns die Namen beider Autoren sowie der Titel ihrer Publikation zur Verfügung. Die übrigen Angaben sind für unsere Zwecke leider nutzlos, da der DOI auf die Abstract-Seite in Xplore verweist, auf welcher wir keine weiteren Informationen zu erwarten haben. Die Seitenangaben helfen uns natürlich auch nicht weiter. Es wäre jedoch möglich, Autorennamen und Titel in einer Suchanfrage zu platzieren, in der Hoffnung, eben jenes o.g. Konferenzprogramm zu finden. Doch selbst wenn ein derartiger 'direkter Treffer' nicht auftritt, so könnte es sein, dass wir ähnliche Seiten finden. Es ist möglich, dass beide genannten Autoren noch andere Publikationen gemeinsam verfasst haben, wie eine Untersuchung sozialer Netzwerke, wie sie beispielsweise in [Reu07] nachzulesen ist, belegt. Ebenso ist es realistisch anzunehmen, dass jene Autoren weitere Publikationen in ähnlichen Themenbereichen verfasst haben. Deshalb scheint eine Suchanfrage, die sowohl die Nachnamen als auch einige aussagekräftige Schlüsselwörter enthält, hierfür äußerst geeignet.

Glücklicherweise haben wir hierzu bereits eine beachtliche Vorarbeit geleistet. In der Enhance-Strategie 2 (vgl. Abschnitt 10.1.2) standen wir vor der Aufgabe, Schlüsselwörter aus einem

[6] http://www.xing.com
[7] http://www.facebook.com
[8] http://www.google.com

Titel zu gewinnen, was wir mittels einer Stoppwort-Eliminierung anhand einer aus DBLP entwickelten Heuristik umgesetzt haben. Wenden wir jenen Algorithmus auf obigen Titel an, so erhalten wir die Begriffe “RAPID”, “Shrinking”, “Congestion”, “Control” und “Timescale” als Schlüsselwörter ($keywords_t$). Ebenso verwenden wir – analog zur Vorgehensweise bei obiger Enhance-Strategie – die verfügbaren Namensteile der Typen PRENAME und SURNAME als $keywords_a$. Da Google je nach Anordnung der Suchbegriffe unterschiedliche Ergebnisse liefert und man davon ausgehen kann, dass jene Begriffe, die weiter vorne in einer Suchanfrage auftauchen, stärker gewichtet werden, stellen wir die $keywords_a$ den $keywords_t$ stets voran. Zudem werden wir, je nachdem zu welchem Namen wir weitere Informationen wünschen, die Reihenfolge der Namen derart verändern, dass der gesuchte Name stets an erste Position der Suchbegriffe steht. Bei zwei Namen wie in obigem Fall mag dies noch keinen großen Unterschied machen, doch bei größeren Namenslisten ergibt diese Vorgehensweise in jedem Fall einen Sinn.

Wollen wir nun nach dem ersten Namen (“V. Konda”) suchen, so erhalten wir demnach

$keywords$ = (”Konda”, “Kaur”, “RAPID”, “Shrinking”, “Congestion”, “Control”, “Timescale”).

Aus diesen lässt sich ein URL generieren, wie auch Google ihn bei Eingabe der Begriffe in das dortige Suchfeld generiert:

```
http://www.google.com/search?q=konda+kaur+rapid+shrinking+congestion+control+timescale
```

Abbildung 10.3 zeigt einen Ausschnitt aus der Ergebnisliste, die mittels jenes URLs generiert wird. Bereits der erste Treffer liefert uns die gewünschten Informationen; es handelt sich bei diesem Treffer offensichtlich um eine im Web frei verfügbare PDF-Version des entsprechenden Artikels, der die vollständigen Namen enthält. Zweiter und dritter Treffer liefern keine weiteren Informationen, verweisen aber mit Sicherheit auf Einträge der gleichen realen Personen; letzterer verweist auf die Abstract-Seite des Ausgangsartikels bei IEEE Xplore. Der vierte Treffer entstammt ebenfalls Xplore, doch sehen wir, dass es sich hierbei um einen anderen Artikel handelt, der ebenfalls von beiden Autoren verfasst wurde und eines der Schlüsselwörter des Titels (“Timescale” ⇒ “timescales”) enthält – den Prozess des Stemmings, wie er in Abschnitt 10.1.2 angesprochen wurde, erledigt hier Google für uns. Dies bestätigt obige These der ‘Co-Autor-’ bzw. persönlichen Netzwerke, sowie die Vermutung, dass einzelne Autoren oftmals mehrere wissenschaftliche Publikationen zu ähnlichen Themen veröffentlichen.

In der Software wurde daher auch eine Google-Anfrage nach obigem Muster implementiert. Wird dem `merge`-Kommando (siehe Anhang A.3.1) nur eine Quelle übergeben, so unternimmt die Software den Versuch, fehlende oder unvollständige Namensteile mittels obiger Anfrage an Google aufzubessern. Google dient uns also praktisch als ‘Orakel’, welches wir immer dann befragen können, wenn uns keine besseren Alternativen zur Verfügung stehen. Hierbei wird stets nur die erste Ergebnisseite ausgewertet, die maximal zehn Ergebnisse enthält. Jene werden aus der Seite extrahiert und einzeln durchsucht. Da es sich nur um kleine Bereiche handelt, von denen wir wissen, dass sie viele der Begriffe der Titel und Autorennamen enthalten, entspricht

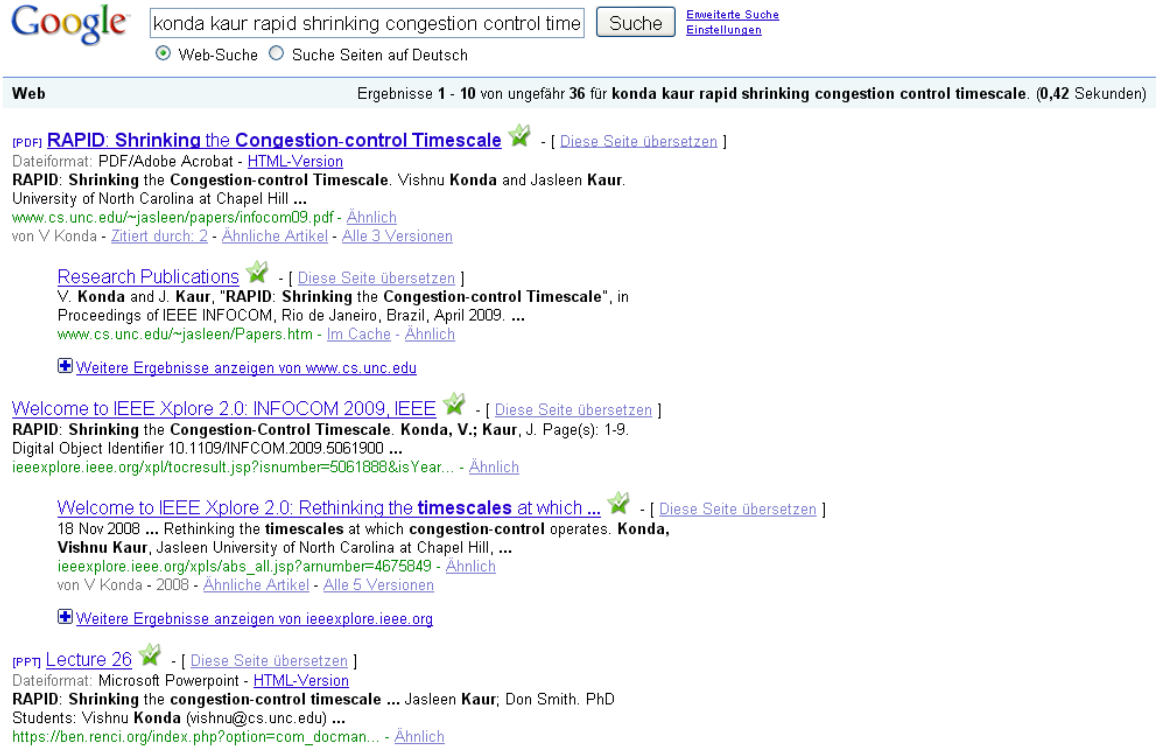


Abb. 10.3: Google-Ergebnis bei Eingabe der genannten Suchbegriffe: Bereits innerhalb der ersten 5 Treffer sind die vollständigen Vornamen beider Autoren zu finden.
Quelle: `http://www.google.com/search?q=konda+kaur+rapid+shrinking+congestion+control+timescale`

dies der Enhance-Strategie 2. Da wir jedoch nun bis zu zehn Teilergebnisse vorliegen haben, aus welchen unterschiedliche Namen extrahiert werden können, müssen wir entscheiden, welchen davon wir als Ergebnis in unsere zweite Record-Liste übernehmen. Uns liegt also ein Datenkonflikt vor, welchen wir mittels einer geschickten Konfliktbehandlungs-Strategie auflösen müssen.

Wir nutzen in diesem Fall eine Variante der in Kapitel 5.3.3 vorgestellten konfliktlösenden ("*Conflict resolution*") Strategie CRY WITH THE WOLVES, indem wir zunächst denjenigen Wert bevorzugen, der am häufigsten auftritt. Hier kann es jedoch passieren, dass wir Werte vergessen würden, die einen höheren Informationsgehalt tragen. Stellen wir uns vor, wir erhielten in obigem Beispiel 10.4 das folgende (konstruierte) Ergebnis:

```
V. Konda      (6x)
Vishnu Konda (3x)
Vishun Konda (1x)
```

Innerhalb der zehn Ergebnisse, die uns Google liefert, käme hiernach in sechs Fällen der Name "V. Konda", in dreien der Name "Vishnu Konda" und in einem der falsch geschriebene Name

“Vishun Konda” vor. Würden wir nur den am häufigsten auftretenden Namen wählen, so hätten wir die korrekte Information praktisch verschenkt. Daher geht die Software wie folgt vor: Der am häufigsten auftretende Name[9] wird als Basis genommen und solange mit weiteren Treffern fusioniert, bis jene Fusion fehl schlägt oder keine weiteren Treffer vorliegen. Im Beispiel würde demnach “V. Konda” als Basis genommen und dieser sodann mit “Vishnu Konda” fusioniert. Nach dem in Kapitel 8.3 erläuterten Algorithmus zur Fusion zweier Autorennamen erhalten wir

$$\text{“V. Konda”} \bowtie \text{“Vishnu Konda”} \longrightarrow \text{“Vishnu Konda”}.$$

Dieser fusionierte Name ist nun wiederum die Basis für die nächste Fusion, diesmal mit dem dritten Kandidaten. Wir erhalten

$$\text{“Vishnu Konda”} \bowtie \text{“Vishun Konda”} \longrightarrow \text{“Vishnu Konda”}.$$

Hier zeigt sich, dass jener Algorithmus recht gute Ergebnisse zu liefern in der Lage ist: Rein syntaktisch ist eine Entscheidung, welcher Vorname der ‘bessere’ ist, nicht möglich. Daher werden beide Namen an die DBLP-Namenssuche weitergeleitet, wodurch der Fehler im zweiten Namen aufgedeckt wird.

Die Software liefert, angewandt auf unsere Testdaten, bereits recht passable Ergebnisse. Gerade wenn ein Artikel über mehrere Autoren verfügt, können durch die kombinierte Anfrage mittels jener Namen zusätzliche, korrekte Ergebnisse gefunden werden. Auf einen vollständigen Test aller Datensätze incl. Auswertung der Ergebnisse wurde an dieser Stelle jedoch wegen der langen Laufzeit jenes Algorithmus’ verzichtet: Diese ist wegen der zahlreichen Anfragen, die an Google gestellt werden (eine pro Namen) recht hoch, zumal es die Software vermeidet, Google mit jenen Anfragen zu ‘bombardieren’ und stets einen respektvollen Abstand mehrerer Sekunden bis zur nächsten Kontaktierung einhält. Abbildung 10.4 zeigt die prozentualen Ergebnisse bei Anwendung auf die ersten zehn Testdatensätze. Wir erkennen, dass hier, selbst im Gegensatz zu Enhance-Strategie 1, in den meisten Fällen deutlich mehr Autorennamen gefunden werden konnten. Lediglich bei Testfall [6] ist das Ergebnis einer Suche mittels Strategie 2 höher. Natürlich sollten alle Ergebnisse auch hier stets einer manuellen Kontrolle unterzogen werden; bei einer oberflächlichen Betrachtung machten sie jedoch meist einen recht guten Eindruck.

Natürlich bestehen im Gebiet der Fusion der bibliographischen Daten zahlreiche Möglichkeiten der Verbesserung. Gerade die Anfrage an Google stellt ein ‘offenes Ende’ dar, hier bieten sich zahlreichen Möglichkeiten, durch Variation der Suchbegriffe die Qualität der Ergebnisse zu erhöhen. Bei einer Anfrage gefundene Kandidaten könnten durch weitere Suchanfragen bestätigt oder verworfen werden. Die Ausgabe der Ergebnisse könnte erweitert werden, um eine manuelle Kontrolle der gefundenen Ergebnisse zu begünstigen; beispielsweise, indem die URLs der von Google aufgespürten Seiten angezeigt würden, um dem Benutzer einen Besuch jener Seiten zu ermöglichen, bei welchem er die Korrektheit direkt überprüfen könnte.

Im Bereich der Enhancer sind, wie bereits zuvor angesprochen, ebenfalls zahlreiche Verbesserungen denkbar. Die anhand der Testdaten erzielten Ergebnisse sind leider noch nicht zufrieden

[9] Bei Gleichstand erfolgt eine Priorisierung nach alphabetischer Reihenfolge.

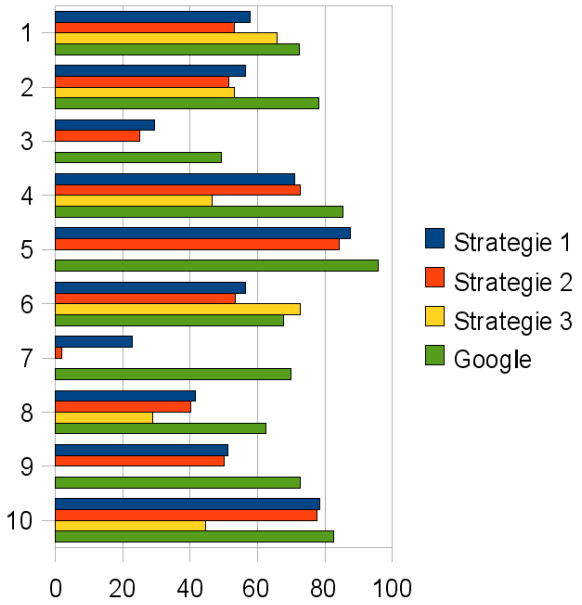


Abb. 10.4: Ergebnisse der Fusion mittels Google: Angewandt auf die ersten 10 Testfälle liefert Google stets eine äußerst hohe Trefferquote.
Quelle: eigene Erstellung

stellend, zumal eine ungewisse Anzahl falscher Datensätze vermutet werden muss. Für eine automatische Fusion mittels des `merge`-Modus eignen sich jene Strategien leider noch nicht. Die reine Erstellung eines Mergelogs mittels des Fusions-Modus `1`, welches anschließend manuell bearbeitet werden muss, kann jedoch bereits eine geringe Arbeitserleichterung darstellen.

Die Software bietet die Möglichkeit, jederzeit neue Enhance-Strategien zu implementieren und auf die Testdaten anzuwenden (siehe Anhang B.6.5). Die vorliegende Arbeit soll neben der konkreten Lösung einiger Probleme vor allem den Weg für weitere Forschung auf diesem Gebiet ebnen und das Interesse an jener wecken.

Literaturverzeichnis

[ACM09] ACM, Association for Computing Machinery: *What is ACM?* `http://www.acm.org/about`, Abruf: 15. September 2009

[ACT09] ACTAPress: *ACTA Press – A Scientific and Technical Publishing Company.* `http://www.actapress.com`, Abruf: 15. September 2009

[AGM03] Arasu, Arvind ; Garcia-Molina, Hector: Extracting Structured Data from Web Pages. In: *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, San Diego, California, USA, June 9-12, 2003*, ACM, 2003, S. 337–348

[AI99] Appelt, Douglas ; Israel, David: *Introduction to Information Extraction Technology. - A Tutorial Prepared for IJCAI-99, SRI International.* `http://www.ai.sri.com/~appelt/ie-tutorial/`, 1999

[BFG01] Baumgartner, Robert ; Flesca, Sergio ; Gottlob, Georg: Visual Web Information Extraction with Lixto. In: *VLDB 2001, Proceedings of 27th International Conference on Very Large Data Bases, September 11-14, 2001, Roma, Italy*, Morgan Kaufmann, 2001, S. 119–128

[Ble04] Bleiholder, Jens: Techniken des Data Merging in Integrationssystemen. In: *Grundlagen von Datenbanken*, 2004, S. 23–27

[BMC09] BMC, BioMed Central: *What is BioMed Central?* `http://www.biomedcentral.com/info`, Abruf: 15. September 2009

[BN06] Bleiholder, Jens ; Naumann, Felix: Conflict Handling Strategies in an Integrated Information System. In: *WWW Workshop in Information Integration on the Web (IIWeb) 2006, Edinburgh, UK.*, 2006

[BN08] Bleiholder, Jens ; Naumann, Felix: Data fusion. In: *ACM Computing Surveys* 41 (2008), Nr. 1

[BYG92] Baeza-Yates, Ricardo A. ; Gonnet, Gaston H.: A New Approach to Text Searching. In: *Communications of the ACM* 35 (1992), Nr. 10, S. 74–82

[Cam09] Cambridge University Press: *2009, a year of anniversaries...* `http://www.cambridge.org/about`, Abruf: 15. September 2009

[CHJ02] Cohen, William W. ; Hurst, Matthew ; Jensen, Lee S.: A flexible learning system for wrapping tables and lists in HTML documents. In: *Proceedings of the Eleventh International World Wide Web Conference, WWW2002, Honolulu, Hawaii, USA, 7-11 May 2002.*, ACM, 2002, S. 232–241

[CKGS06] Chang, Chia-Hui ; Kayed, Mohammed ; Girgis, Moheb R. ; Shaalan, Khaled F.: A Survey of Web Information Extraction Systems. In: *IEEE Transactions on Knowledge and Data Engineering* 18 (2006), Nr. 10, S. 1411–1428

[CSS99] Conrad, Stefan ; Saake, Gunter ; Sattler, Kai-Uwe: *Informationsfusion – Herausforderungen an die Datenbanktechnologie (Kurzbeitrag).* `http://infovis.uni-konstanz.de/papers/2000/ConSatSaa99.pdf`, 1999

[Cun06] Cunningham, Hamish: Information Extraction, Automatic. Elsevier, 2006, S. 665–677

[DBL09] DBLP: *Frequently Asked Questions (FAQ).* `http://dblp.uni-trier.de/db/about/faq.html`, Abruf: 15. September 2009

[DP08] Durme, Benjamin V. ; Pasca, Marius: Finding Cars, Goddesses and Enzymes: Parametrizable Acquisition of Labeled Instances for Open-Domain Information Extraction. In: *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, AAAI Press, 2008, S. 1243–1248

[EBS09] EBSCO: *MetaPress.* `http://www.ebscoind.com/groups-gs-mp.asp`, Abruf: 15. September 2009

[EHS04] Ehrig, Marc ; Hartmann, Jens ; Schmitz, Christoph: Ontologie-basiertes Web Mining. In: *INFORMATIK 2004 - Informatik verbindet, Band 2, Beiträge der 34. Jahrestagung der Gesellschaft für Informatik e.V. (GI), Ulm, 20.-24. September 2004* Bd. 51, GI, 2004 (LNI), S. 187–192

[Els09] Elsevier: *Elsevier at a Glance.* `http://www.elsevier.com/wps/find/intro.cws_home/ataglance`, Abruf: 15. September 2009

[FZ09] Faúndez-Zanuy, Marcos: Data Fusion at Different Levels. In: *Multimodal Signals: Cognitive and Algorithmic Issues, COST Action 2102 and euCognition International School Vietri sul Mare, Italy, April 21-26, 2008, Revised Selected and Invited Papers* Bd. 5398, Springer, 2009 (Lecture Notes in Computer Science), S. 94–103

[Goo09] Google: *Über Google Scholar.*
`http://scholar.google.de/intl/de/scholar/about.html`,
Abruf: 15. September 2009

[GS96] Grishman, Ralph ; Sundheim, Beth: Message Understanding Conference- 6: A Brief History. In: *COLING 1996, 16th International Conference on Computational Linguistics, Proceedings of the Conference, August 5-9, 1996, Center for Sprogteknologi, Copenhagen, Denmark.* Bd. 2, 1996, S. 466–471

[HFAN98] Huck, Gerald ; Fankhauser, Peter ; Aberer, Karl ; Neuhold, Erich J.: Jedi: Extracting and Synthesizing Information from the Web. In: *Proceedings of the 3rd IFCIS International Conference on Cooperative Information Systems, New York City, New York, USA, August 20-22, 1998, Sponsored by IFCIS, The Intn'l Foundation on Cooperative Information Systems*, IEEE Computer Society, 1998, S. 32–43

[HGE07] Holzinger, Andreas ; Geierhofer, Regina ; Errath, Maximilian: Semantische Informationsextraktion in medizinischen Informationssystemen. Springer, 2007, S. 69–78

[ICS09a] ICST, Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering: *EU Digital Library – Providing access to thousands of scientific articles from all fields of ICT research.*
`http://eudl.eu/About`,
Abruf: 15. September 2009

[ICS09b] ICST, Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering: *Leading the ICT ®evolution in Service of Humanity.*
`http://www.icst.org/about`,
Abruf: 15. September 2009

[IEE09a] IEEE, Institute of Electrical and Electronics Engineers: *About IEEE.*
`http://www.ieee.org/web/aboutus/home/index.html`,
Abruf: 15. September 2009

[IEE09b] IEEE, Institute of Electrical and Electronics Engineers: *IEEE Today.*
`http://www.ieee.org/web/aboutus/today/index.html`,
Abruf: 15. September 2009

[IGI09] IGI Global: *About IGI Global.*
`http://www.igi-global.com/about`,

Abruf: 15. September 2009

[Ind09] Inderscience: *About Inderscience Publishers.* `http://www.inderscience.com/mapper.php?id=11`, Abruf: 15. September 2009

[IOS09] IOS Press: *About IOS Press.* `http://www.iospress.nl`, Abruf: 15. September 2009

[IS06] Irmak, Utku ; Suel, Torsten: Interactive wrapper generation with minimal user effort. In: *Proceedings of the 15th international conference on World Wide Web, WWW 2006, Edinburgh, Scotland, UK, May 23-26, 2006*, ACM, 2006, S. 553–563

[KT04] Koike, Asako ; Takagi, Toshihisa: PRIME: automatically extracted PRotein Interactions and Molecular Information databas. In: *In Silico Biology* 5 (2004)

[Kus00] Kushmerick, Nicholas: Wrapper induction: Efficiency and expressiveness. Elsevier, 2000, S. 15–68

[KWD97] Kushmerick, Nicholas ; Weld, Daniel S. ; Doorenbos, Robert B.: Wrapper Induction for Information Extraction. In: *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, IJCAI, Volume 1*, Morgan Kaufmann, August 1997, S. 729–737

[Lev66] Levenshtein, Vladimir I.: Binary Codes Capable of Correcting Deletions, Insertions and Reversals. In: *Soviet Physics Doklady* 10 (1966), Februar, S. 707 ff.

[Ley02] Ley, Michael: The DBLP Computer Science Bibliography: Evolution, Research Issues, Perspectives. In: *String Processing and Information Retrieval, 9th International Symposium, SPIRE 2002, Lisbon, Portugal, September 11-13, 2002, Proceedings* Bd. 2476, Springer, 2002 (Lecture Notes in Computer Science), S. 1–10

[Ley09] Ley, Michael: DBLP - Some Lessons Learned. In: *PVLDB* 2 (2009), Nr. 2, S. 1493–1500

[LG09] Lee, Yeong S. ; Geierhos, Michaela: Business Specific Online Information Extraction from German Websites. In: *Computational Linguistics and Intelligent Text Processing, 10th International Conference, CICLing 2009, Mexico City, Mexico, March 1-7, 2009. Proceedings* Bd. 5449, Springer, 2009 (Lecture Notes in Computer Science), S. 369–381

[LGM08] Lam, Man I. ; Gong, Zhiguo ; Muyeba, Maybin K.: A Method for Web Information Extraction. In: *Progress in WWW Research and Development, 10th Asia-Pacific Web Conference, APWeb 2008, Shenyang, China, April 26-28, 2008.*

Proceedings Bd. 4976, Springer, 2008 (Lecture Notes in Computer Science), S. 383–394

[LR06] Ley, Michael ; Reuther, Patrick: Maintaining an Online Bibliographical Database: The Problem of Data Quality. In: *Revue des Nouvelles Technologies de l'Information* Bd. RNTI-E-6, Cépaduès-Éditions, 2006, S. 5–10

[MH09] Moens, Marie-Francine ; Hiemstra, Djoerd: Information Extraction and Linking in a Retrieval Context. In: *Advances in Information Retrieval, 31th European Conference on IR Research, ECIR 2009, Toulouse, France, April 6-9, 2009. Proceedings* Bd. 5478, Springer, 2009 (Lecture Notes in Computer Science), S. 810–813

[MMK99] Muslea, Ion ; Minton, Steven ; Knoblock, Craig A.: A hierarchical approach to wrapper induction. In: *Proceedings of the Third International Conference on Autonomous Agents*, ACM Press, New York, 1999, S. 190–197

[NBBW06] Naumann, Felix ; Bilke, Alexander ; Bleiholder, Jens ; Weis, Melanie: Data Fusion in Three Steps: Resolving Schema, Tuple, and Value Inconsistencies. In: *IEEE Data Engineering Bulletin* 29 (2006), Nr. 2, S. 21–31

[Neu01] Neumann, Günter: Informationsextraktion. In: *Computerlinguistik und Sprachtechnologie. Eine Einführung*, Spektrum, 2001, S. 448–455

[Pas06] Paskin, Norman: *The DOI Handbook - Version 4.4.1.* `http://dx.doi.org/10.1000/186`

[Por80] Porter, Martin F.: An algorithm for suffix stripping. In: *Program* 14 (1980), Nr. 3, S. 130–137

[PSRV05] Papadakis, Nikolaos ; Skoutas, Dimitrios ; Raftopoulos, Konstantinos ; Varvarigou, Theodora A.: STAVIES: A System for Information Extraction from Unknown Web Data Sources through Automatic Web Wrapper Generation Using Clustering Techniques. In: *IEEE Transactions on Knowledge and Data Engineering* 17 (2005), Nr. 12, S. 1638–1652

[Reu07] Reuther, Patrick: *Namen sind wie Schall und Rauch: Ein semantisch orientierter Ansatz zum Personal Name Matching*, Diss., 2007

[RL07] Ruser, Heinrich ; León, Fernando P.: Informationsfusion – Eine Übersicht. In: *Technisches Messen* 74 (2007), Nr. 3, S. 93–102

[RWL+06] Reuther, Patrick ; Walter, Bernd ; Ley, Michael ; Weber, Alexander ; Klink, Stefan: Managing the Quality of Person Names in DBLP. In: *Research and Advanced Technology for Digital Libraries, 10th European Conference, ECDL 2006, Alicante, Spain, September 17-22, 2006, Proceedings* Bd. 4172, Springer, 2006 (Lecture Notes in Computer Science), S. 508–511

[SALM01] Santos, Simone C. ; Angelim, Sérgio ; Lemos Meira, Silvio R.: Building Comparison-Shopping Brokers on the Web. In: *Electronic Commerce, Second International Workshop, WELCOM 2001 Heidelberg, Germany, November 16-17, 2001, Proceedings* Bd. 2232, Springer, 2001 (Lecture Notes in Computer Science), S. 26–38

[Sar92] Sarkowski, Heinz: *Der Springer-Verlag – Stationen seiner Geschichte. Teil I: 1842-1945*. Springer, 1992

[Sar08] Sarawagi, Sunita: Information Extraction. In: *Foundations and Trends in Databases* 1 (2008), Nr. 3, S. 261–377

[Sch06] Schubert, Lenhart K.: Turing's Dream and the Knowledge Challenge. In: *Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, USA*, AAAI Press, 2006

[Sch07] Schrenk, Michael: *Webbots, Spiders, and Screen Scrapers - A Guide to Developing Internet Agents with PHP/CURL*. No Starch Press, 2007

[SIA02] SIAM, Society for Industrial and Applied Mathematics: *Looking Back, Looking Ahead: A SIAM History*.
`http://www.siam.org/about/more/siam50.pdf`, September 2002

[SJ09] Steinberger, Josef ; Jezek, Karel: Text Summarization: An Old Challenge and New Approaches. In: *Foundations of Computational (6)*. Springer, 2009, S. 127–149

[SMB05] Scannapieco, Monica ; Missier, Paolo ; Batini, Carlo: Data Quality at a Glance. In: *Datenbank-Spektrum* 14 (2005), S. 6–14

[Spr09a] Springer: *LNCS Overview*.
`http://www.springer.com/computer/lncs?SGWID=0-164-6-73659-0`,
Abruf: 15. September 2009

[Spr09b] Springer: *Springer Science+Business Media: Unternehmensgeschichte*.
`http://www.springer-sbm.de/index.php?id=165&L=1`,
Abruf: 15. September 2009

[VDE08] VDE, Verband der Elektrotechnik Elektronik Informationstechnik e.V. (Hrsg.): Informationsfusion: Servolenkung für das Hirn. VDE, 2008, S. 8–9

[VG07] Vuong, Le Phong B. ; Gao, Xiaoying: Using Clustering for Web Information Extraction. In: *Australian Conference on Artificial Intelligence* Bd. 4830, Springer, 2007 (Lecture Notes in Computer Science), S. 415–424

[WG09] Wunder, Michael ; Grosche, Jürgen: *Verteilte Führungsinformationssysteme*. Springer, 2009

[Wor09] World Scientific Publishing Company: *Corporate Profile*. `http://www.worldscientific.com/profiles/corporates.shtml`, Abruf: 15. September 2009

[YRW08] Yao, Jingtao ; Raghavan, Vijay V. ; Wu, Zonghuan: Web information fusion: A review of the state of the art. Elsevier, 2008, S. 446–449

[ZLH08] Zhao, Xiaochuan ; Luo, Qingsheng ; Han, Baoling: Survey on robot multi-sensor information fusion technology. In: *Intelligent Control and Automation, 2008. WCICA 2008. 7th World Congress on*, 2008, S. 5019–5023

Anhang A

Anleitung zur Bedienung der Software

Dieses Kapitel soll eine kurze Einführung in die Bedienung der einzelnen Teile der zur vorliegenden Diplomarbeit gehörenden Softwarepakete liefern. Diese geschieht rein aus Sicht eines Anwenders und verbirgt sämtliche technischen Details.

Eine Kurzfassung dieser Anleitung liegt der Software auf der mitgelieferten CD-ROM in Form einzelner '`doku_X.txt`'-Dateien bei, die sich im Hauptverzeichnis befinden. Eine genauere Beschreibung der Arbeitsweise der einzelnen Softwareteile ist den Kapiteln 4 (Wrapper) und 10 (Merger) zu entnehmen. Eine detaillierte Beschreibung der einzelnen Java-Klassen findet sich in Anhang B. Noch spezifischere Informationen liefern die Kommentare innerhalb des auf der beiliegenden CD-ROM befindlichen Quelltextes.

Die gesamte Software arbeitet kommandozeilenorientiert, ein GUI wurde nicht implementiert. Da der Aufruf der einzelnen Klassen dem Aufruf von Shellkommandos ähnelt, sollen diese im Folgenden ebenfalls als *Kommandos* bezeichnet werden. Da es sich jedoch bei den Kommandos um Java-Programme handelt, ist stets das Kommando `java` voranzustellen.

Zur Syntax dieses Kapitels Kommandos werden durch eine `spezielle Schriftart` kenntlich gemacht. Ganze Kommandozeilen-Ausdrücke beginnen stets mit einem "`>`". Bei den Beschreibungen der Kommandos bezeichnet ein Ausdruck in `<`spitzen Klammern`>` einen Parameter, der anschließend näher erläutert wird. Parameter in `[`eckigen Klammern`]` sind optional.

A.1 Konfiguration der einzelnen Software-Pakete

Alle Pakete der beiliegenden Software greifen auf ein und dieselbe Konfigurationsdatei zu. Diese liegt im XML-Format vor und befindet sich unter '`config/config.xml`'. Die Datei ist in einzelne Abschnitte (sections) untergliedert, die für die jeweiligen Teile der Software (Handler, Wrapper, Merger) zuständig sind.

Im Default-Bereich werden solche Konfigurationen eingestellt, die allen Klassen dienen. Werden in einzelnen Klassen spezielle Werte eingestellt, so werden die entsprechenden Defaultwerte überschrieben.

Hier eine Liste der generellen Einstellungen:

loglevel
Hier muss ein Wert zwischen 1 und 9 eingegeben werden, je nachdem, welche Meldungen ausgegeben werden sollen (vgl. Tabelle A.1).

1	critical	Meldungen bei Programmabbruch
2	error	Fehlermeldungen
3	warning	Warnmeldungen
4	notice	Wichtige Meldungen
5	info	Informationen
6	detailed	Detaillierte Informationen
7	–	(nicht belegt)
8	debug	Debugging-Meldungen
9	all	Zahlreiche zusätzliche Meldungen

Tab. A.1: Detailstufen bei der Ausgabe der Logmeldungen

Die Level sind jeweils inklusive der darunter liegenden zu verstehen, d.h. bei Wahl eines Loglevels von 2 werden sowohl kritische Meldungen (1) als auch Fehler (2) angezeigt, bei Wahl von 5 (Standardeinstellung) werden alle Meldungen der Stufen 1 bis 5 ausgegeben.

logformat
Mögliche Werte sind "`long`" und "`short`". Im `long`-Modus werden Datum und Uhrzeit, loglevel und der Name der jeweiligen Klasse bei den Logmeldungen vorangestellt. Im `short`-Modus erscheint nur die eigentliche Meldung.

logtype
Mögliche Werte sind "`screen`" und "`file`". Bei der Wahl von "`screen`" werden alle Logmeldungen direkt auf dem Bildschirm ausgegeben; wählt man "`file`", so werden diese in eine Datei geschrieben, welche mittels des Elements "`logfile`" zu definieren ist.

missing_pages
Gibt den Standardwert an, der benutzt wird, falls das *pages*-Attribut leer ist. Derzeit ist dies stets "`0-`".

output_type
Mögliche Werte sind "`bht`" und "`xml`". Hier wird angegeben, in welchem Format das Ergebnis der Wrapper geschrieben werden soll. Die Merger nutzen diese Einstellung nicht, sondern geben das Ergebnis analog zur primären Quelle aus (als BHT-Datei oder in Form einzelner Records).

Diese o.g. Default-Werte können in jeder Section verändert werden. Beispielsweise kann der Loglevel erhöht werden, wenn ein neuer Wrapper getestet werden soll. Für einige Klassen gibt es spezielle Elemente, die im Folgenden jeweils erklärt werden.

A.2 Bedienung der Wrapper-Software

Die Wrapper-Software dient der Extraktion der bibliographischen Daten von einer der in Kapitel 3 vorgestellten Extraktionsquellen. Mittels des `get`-Kommandos lassen sich bequem einzelne Konferenzbände oder ganze Zeitschriftenserien extrahieren. Die Ausgabe erfolgt wahlweise in BHT-Dateien oder einem DTD-konformen XML-Format.

A.2.1 `get` – Extraktion der bibliographischen Daten

Das `get`-Kommando hat die Syntax

```
> java get [<options>] <url> [<start> [<end>]] [<options>]
```

Als einziger obligatorischer Parameter muss als `<url>` der URL einer der zu bearbeitenden Webseite angegeben werden. Dies kann eine Konferenz sein (in diesem Falle genügt die Angabe des URLs) oder ein Journal (in diesem Falle wird noch mindestens der `<start>`-Parameter benötigt). Der URL muss einer der Wrapper-Software bekannten Domain entstammen und einen der Domain entsprechenden Schlüssel beinhalten, mittels dessen die zu bearbeitende Publikation ermittelt werden kann (siehe A.2.2). Enthält ein URL Sonderzeichen wie z.B. das "&", so ist er in Anführungszeichen zu setzen.

Die Parameter `<start>` und `<end>` werden nur bei Journalen benötigt; dort ist erstgenannter jedoch verpflichtend. Sind beide Werte angegeben, so werden sie als erstes bzw. letztes zu erfassendes Volume bzw. Issue angesehen. Wurde nur der Startwert gesetzt, so werden nur die Daten dieses einen Volumes bzw. Issues extrahiert.

Die Werte müssen stets in der Form `<volume>[.<issue>]` angegeben werden, d.h. die Nummer des Bandes ist stets erforderlich, eine Heftnummer kann optional hinter einem Punkt angefügt werden. Bei beiden Werten werden normalerweise Integer-Zahlen erwartet, einzige Ausnahme ist die Erfassung so genannter "Supplements". Hier darf dem Issue ein kleines "s" vorangestellt werden, um nur ein bestimmtes Supplement zu erfassen.

Beispiele[1]

```
> java get <url> 55       // bearbeitet den kompletten Band 55
```

[1] Bei allen Beispielen sei vorausgesetzt, dass die entsprechenden Bände und Nummer auch existieren. Ansonsten würde eine leere Datei als Ergebnis geliefert.

```
> java get <url> 55 57    // bearbeitet die kompletten Bände 55, 56 und 57
> java get <url> 55.2     // bearbeitet Band 55, Heft 2
> java get <url> 55.2 56  // bearbeitet alle Hefte aus Band 55 bis auf das
                          // erste, sowie den kompletten Band 56
> java get <url> 55.s2    // bearbeitet das 2. Supplement des Bandes 55
```

Die Optionen (`<options>`) sind wahlfreie Parameter, die in beliebiger Reihenfolge auftreten können und den übrigen Parametern voran- oder nachgestellt sein dürfen. Optionen werden stets von einem "`-`"-Zeichen angeführt und fungieren entweder als simple Schalter (d.h. sie bewirken bereits durch ihre Anwesenheit eine Veränderung der Ausführung) oder als Schlüssel-Werte-Paare.

Mögliche Optionen sind:

`-q` bzw. `-quiet`
Unterdrückt die Ausgaben von Meldungen während der Programmausführung. Trotzdem werden weiterhin die Logmeldungen entsprechend ihres Levels ausgegeben, sofern in der Konfigurationsdatei (siehe A.1) der logtype 'screen' gewählt wurde.

`-output=[bht|xml]`
Legt das Ausgabeformat fest. Fehlt diese Option, wird der Wert der Konfigurationsdatei (siehe hierzu Anhang B.2) genommen.

`-filename=xxx`
Legt den Präfix der Ausgabedateien fest. Fehlt diese Option, wird ein Standardname verwendet.

`-d=xxx`
Absoluter oder relativer Pfad (allerdings sind `./` und `../` nicht erlaubt) des Verzeichnisses, in welches die Ausgabedateien geschrieben werden. Fehlt diese Angabe, so werden die Dateien ins aktuelle Verzeichnis geschrieben.

A.2.2 Unterstützte Verlage

Tabelle A.2 liefert eine Übersicht über die Verlage, zu denen ein Wrapper verfügbar ist, sowie den Parameter oder die Verzeichnisstruktur innerhalb des einzugebenden URLs, über den die zu extrahierenden Daten identifiziert werden.

Einige Buchserien (book series) sind hier gesondert aufgeführt, wenn deren URL (und somit auch deren Publikationsschlüssel) sich von den anderen Publikationen unterscheidet. Intern werden diese Serien jedoch unterschiedlich gehandhabt. Die Buchserien bei IOS benötigen eine Volume-Angabe und werden wie Journale behandelt, während es sich bei Wiley um eine Enzyklopädie handelt, die als $\mathrm{BHT_c}$-Datei ausgegeben wird.

book series	
IOS Press	`piid=<key>`
Wiley	`emrw/<key1>/<key2>`
conferences	
ACM	`idx=<key> | id=<key>`
ICST / EUDL	`eudlQuery=<key>`
IEEE / Xplore	`punumber=<key> | isnumber=<key1>&isYear=<key2>`
journals	
ACM	`id=<key> | idx=<key>`
ACTA Press	`journalID=<key>`
BMC	(aus URL, verschiedene Domains möglich)
Cambridge U. P.	`jid=<key>`
Elsevier	`/journal/<key>`
IEEE / Xplore	`punumber=<key>`
IEICE	`category=<key>`
IGI-Global	`id=<key> | ID=<key>`
Inderscience	`journalID=<key> | journalCode=<key>`
informs	`<key>.journal.informs.org`
IOS Press	`/content/<key>/`
MetaPress	`/content/<key>/`
MIT Press	`/loi|toc/<key>`
Oxford U. P.	`<key>.oxfordjournals.org | www.oup.co.uk/<key>`
Revues online	`<key>.revuesonline.com`
Sage	`<key>.sagepub.com`
SIAM	`KEY=<key>`
Springer	`/content/<key>/`
Taylor & Francis	`/title~content=<key>`
Wiley	`/journal/<key>/`
WorldScientific	`www.worldscinet.com/<key>/<key>.shtml`

Tab. A.2: Publikationsschlüssel der Verlagsserver

A.3 Bedienung der Merger-Software

Die Merger-Software dient der Fusion bibliographischer Daten und führt die in den Kapiteln 6 bis 10 beschriebenen Arbeitsschritte aus. Das Kommando `merge` bearbeitet hierbei, je nach Eingabeparameter, sowohl symmetrische als auch asymmetrische Quellen. Wegen der hohen Priorität des in Szenario F-2'$_{\text{LNCS}}$ beschriebenen Problems der Ersetzung von URL durch DOIs in tausenden von Bänden der Springer LNCS-Serie wurde hierzu ein weiteres Kommando (`fixLNCS`, siehe A.3.2) implementiert, welches diese Aufgabe äußerst komfortabel erfüllt.

A.3.1 `merge` – Fusion zweier Datenquellen

Mittels des Kommandos

```
> java merge [<options>] <source1> [<source2>] [<options>]
```

wird der Fusionsprozess der beiden Quellen gestartet. `<source1>` bezeichnet hierbei die primäre Quelle, bei welcher es sich um eine BHT$_{\text{c/j}}$- oder BHT$_{\text{cite}}$-Datei handeln muss. Die sekundäre Quelle (`<source2>`) dagegen kann entweder eine BHT$_{\text{c/j}}$-Datei sein, ein URL zu einer Konferenz-Website, oder eine PDF-Datei. Fehlt der Parameter völlig, so wird als sekundäre Quelle das WWW angesehen und Kapitel 10.3 entsprechend die Suchmaschine Google befragt.

Werden keine weiteren Optionen angegeben, so werden die Default-Einstellungen genutzt, die in der Konfigurationsdatei in der Sektion "BaseMerger" eingetragen sind.

Prinzipiell muss bei den `<options>` zwischen allgemeinen Optionen und solchen Optionen, die die Fusions-Modi bestimmen, unterschieden werden.

Allgemeine Optionen
`-q` bzw. `-quiet`
Unterdrückt die Ausgaben von Meldungen während der Programmausführung. Trotzdem werden weiterhin die Logmeldungen entsprechend ihres Levels ausgegeben, sofern in der Konfigurationsdatei (siehe A.1) der logtype 'screen' gewählt wurde.

`-filename=xxxxx`
Legt den Namen der Ausgabedateien fest. Fehlt diese Option, wird ein Standardname verwendet.

`-d=xxxxxx`
Absoluter oder relativer Pfad (allerdings sind `./` und `../` nicht erlaubt) des Ausgabeverzeichnisses, sofern das Ergebnis eine BHT$_{\text{c/j}}$-Datei ist. Werden bestehende DBLP-Records mit einer zweiten Quelle fusioniert (d.h. wird gemäß Szenario F-2' in Kapitel 6.1.3 eine BHT$_{\text{cite}}$-Datei

als primäre Quelle angegeben), so entsprechen die Verzeichnisse den in der Konfigurationsdatei angegebenen Werten, wie weiter unten beschrieben.

`-strategy=x`
Mit dieser Option kann bei Aufruf festgelegt werden, welche Enhance-Strategie (vgl. Kapitel 10.1) verwendet werden soll, um eine Fusion mit Daten einer unstrukturierten Quelle durchzuführen. Ist die sekundäre Quelle eine BHT-Datei, so wird diese Option ignoriert.

Die Fusions-Modi festlegende Optionen
Über diese Optionen können einzelne Fusions-Modi eingestellt werden, die angeben, aus welcher Quelle die Informationen übernommen werden sollen. Anzugeben ist jeweils ein Schlüssel-Wert-Paar. Tabelle A.3 bietet eine Übersicht der möglichen Schlüssel und Werte, die nach Belieben[2] kombiniert und wie normale Optionen angegeben werden können. Die Bedeutung der Fusions-Modi ist Kapitel 6.2.3 zu entnehmen.

Schlüssel		Werte	
-records	(oder kurz: -r)	=1	
-authors	(oder kurz: -a)	=2	
-title	(oder kurz: -t)	=merge	(oder kurz: =m)
-pages	(oder kurz: -p)	=ignore	(oder kurz: =i)
-ee	(oder kurz: -e)		
-section	(oder kurz: -s)		

Tab. A.3: Schlüssel und Werte zur Angabe der Fusions-Modi

Möchte man beispielsweise, dass die Seitenangaben aus der primären Quelle entnommen werden, die EE-Links aus der sekundären, und dass bei den Autorennamen stets der fusionierte Wert übernommen wird, so muss das `merge`-Kommando lediglich mit den Optionen "`-pages=1 -ee=2 -authors=merge`" bzw. in Kurzschreibweise "`-p=1 -e=2 -a=m`" aufgerufen werden.

Werden keine Werte angegeben, so gelten, wie bereits erwähnt, die Default-Werte, die in der `config.xml`-Datei definiert sind (s.u.). Derzeit gilt dort stets "1", d.h. die erste Quelle bleibt unverändert, aber es werden Informationen angezeigt, wenn bessere Werte verfügbar sind.

Beispiele zur Verdeutlichung der Fusions-Modi
`> java merge -ee=merge <source1> <source2>`
Hier wird versucht, bessere URLs aus der zweiten Quelle zu gewinnen und diese ins Ergebnis zu übernehmen.

`> java merge -ee=1 <source1> <source2>`
Hier werden in jedem Fall die ursprünglichen URLs übernommen, doch es wird eine Meldung ausgegeben, wenn der Wert der zweiten Quelle besser erscheint.

[2]Einzige Ausnahme bildet hier die Regel, dass dem Schlüssel 'records' nicht der Wert "2" zugeordnet werden darf (siehe Kapitel 6.2.3).

`> java merge -ee=2 <source1> <source2>`
Hier werden in jedem Fall die Werte der zweiten Quelle übernommen. Dabei ist äußerste Vorsicht geboten, denn hier ist es möglich, dass sich die Datensätze bei der Bearbeitung verschlechtern, falls in der zweiten Quelle schlechtere Informationen stehen. Der Modus "2" sollte nur im Ausnahmefall gewählt werden, wenn die Güte der Informationen der zweiten Quelle absolut sicher ist. Ansonsten sollte stattdessen immer der Modus "merge" gewählt werden, da hier nur im Fall, dass etwas besseres, also z.B. ein DOI statt eines normalen URLs, gefunden wurde, der Wert der primären Quelle überschrieben wird.

`> java merge -ee=ignore <source1> <source2>`
Mit dieser Option werden die EE-Attribute überhaupt nicht überprüft, und es werden demnach auch keine entsprechenden Meldungen ausgegeben.

Konfiguration
In der `config.xml`-Datei können einige Voreinstellungen getroffen werden. Wichtigste Parameter sind die Default-Fusions-Modi, die in den jeweiligen Sections "`BhtMerger`", "`HtmlMerger`", "`PdfMerger`" und "`GoogleMerger`" differenziert eingetragen werden können. Der Name des Mergers bezieht sich hierbei stets auf die sekundäre Quelle, da als primäre Quelle in allen Fällen sowohl $\text{BHT}_{\text{c/j}}$- als auch BHT_{cite}-Dateien erlaubt sind. Für letztere können zusätzlich im Abschnitt "`RecordsHandler`" die Quell-, Ziel- und Backup-Pfade angegeben werden. Tabelle A.4 zeigt eine Übersicht der Elemente, die hier gewählt werden können.

merge_records merge_title merge_authors merge_pages merge_ee merge_sections	Über diese Elemente können die Default-Fusions-Modi eingestellt werden. Mögliche Werte sind stets "1", "2", "merge" und "ignore". Die Werte haben die gleiche Bedeutung wie die entsprechenden Optionen (siehe dort).
records_base	Quellverzeichnis, aus dem die Records gelesen werden. Dies sollte i.A. '`/dblp/publ`' sein.
records_dest	Zielverzeichnis, in welches die Records geschrieben werden. Wird hier das gleiche Verzeichnis wie in *records_base* angegeben, so werden alte Einträge automatisch überschrieben. Ist ein anderes Verzeichnis angegeben, so werden die Records darin in entsprechenden Unterverzeichnissen abgelegt. (Liegen die Originale z.B. unter '`/dblp/publ/conf/testconf/`', und ist als *records_dest* das Verzeichnis '`/tmp`' angegeben, so werden die Ergebnisrecords unter '`/tmp/conf/testconf/`' zu finden sein.)
records_backup	Backup-Verzeichnis, in welches die Originalrecords unter Erhaltung ihres letzten Änderungsdatums (*mdate*) kopiert werden, um sie im Zweifelsfall wiederherstellen zu können.

Tab. A.4: Konfigurationsparameter des `merge`-Kommandos

Beispiele zum Aufruf des merge-Kommandos[3]

`> java merge ieee.bht eudl.bht`

Hier werden zwei $BHT_{c/j}$-Dateien mit Quellen unterschiedlicher Verlage gemäß Szenario F-1 (siehe Kapitel 6.1.1) fusioniert. Sinnvollerweise sollten diese natürlich bibliographische Einträge der gleiche Konferenz bzw. des gleichen Volumes und Issues eines Journals enthalten. Da keine Fusions-Modi angegeben wurden, werden die Default-Werte der Konfigurationsdatei genommen, welche bei Fertigstellung der vorliegenden Arbeit allesamt "1" lauten. Vermeintlich bessere Werte der sekundären Quelle werden daher angezeigt, aber nicht übernommen. Auch 'Single'-Records der sekundären Quelle werden nicht übernommen. Die Ergebnisdatei ist daher identisch zur primären Quelle (d.h. der Datei `ieee.bht`).

`> java merge /dblp/ht/db/journals/network/network20.bht network_new.bht`

Dieser Aufruf fusioniert bestehende Records mit einer neuen BHT_j-Datei. Die Speicherung der Ergebnis-Records erfolgt gemäß der Pfade, die in der `config.xml`Datei (s.o.) eingetragen wurden. Für die Fusions-Modi gelten die gleichen Aussagen wie im letzten Beispiel.

`> java merge -authors=1 -title=1 -ee=merge -pages=merge ieee.bht eudl.bht`

Hier werden die gleichen Dateien wie im ersten Beispiel bearbeitet. Erzeugt wird eine neue Datei, die, da keine '`-filename`'-Option angegeben wurde, den Standardnamen `mergeresult.bht` tragen wird, in welcher die jeweils fusionierten Werte für URLs (`-ee=merge`) und Seitennummern (`-pages=merge`) übernommen wurden. Autorennamen und Titel bleiben unverändert. Während der Fusion werden jedoch auch diese überprüft und eventuell ungleiche Werte mit einem Verbesserungsvorschlag ausgegeben. Zwischenüberschriften und 'Single'-Records werden gemäß den Angaben der Konfigurationsdatei behandelt.

`> java merge -a=1 -t=1 -e=m -p=m ieee.bht eudl.bht`

Dieses Beispiel ist völlig identisch zum vorherigen, jedoch in Kurzschreibweise.

`> java merge -a=i -t=i -p=i -e=m -s=i -r=i /dblp/ht/.../lncs.bht new_lncs.bht`

Hier werden lediglich URLs durch (eventuell in der zweiten Quelle vorhandene) DOIs ersetzt. Die Dateinamen deuten an, dass es sich bei den bibliographischen Daten um Artikel der LNCS handelt. Für diese Aufgabe kann das Kommando `fixLNCS` (siehe Abschnitt A.3.2) genutzt werden, welches erheblich komfortabler zu benutzen ist, da es automatisch die entsprechenden Daten mittels des Wrappers erfasst sowie ganze Zahlenränge bearbeitet.

A.3.2 `fixLNCS` – Austausch alter URLs gegen DOIs bei den Springer LNCS

Mittels `fixLNCS` können schnell ganze Bände von LNCS-Records um evtl. fehlende DOIs ergänzt werden. Die Records werden in einem zuvor in der `config/config.xml`-Datei festzu-

[3] Bei diesen Beispielen wollen wir stets voraussetzen, dass die entsprechenden Dateien existieren und deren Inhalt dem entspricht, was ihr Dateiname vermuten lässt. Eine Datei mit Namen "eudl.bht" soll demnach eine Liste von Artikeln in $BHT_{c/j}$-Format enthalten, die der EUDL (vgl. Kapitel 3.2.6) entnommen sind.

legenden Verzeichnis abgelegt. Dies kann entweder ein neues Verzeichnis, oder aber die Quelle (`/dblp/publ`) sein, sofern dort Schreibrecht existiert. Die tiefere Verzeichnisstruktur wird dabei innerhalb des angegebenen Ordners automatisch angelegt (also z.B. `conf/<confkey>/`).

Das Skript bearbeitet ausschließlich die *ee*-Links und verändert diese nur dann, wenn ein URL in der ersten Quelle existiert, in der zweiten Quelle aber ein DOI gefunden wurde. Sollen weitere Änderungen durchgeführt werden, so ist das `merge`-Kommando zu verwenden.

Die Syntax des `fixLNCS`-Kommandos lautet

```
> java fixLNCS [<options>] <start> [<end>] [<options>]
```

Der obligatorische `<start>`-Parameter gibt hierbei die Nummer des ersten zu bearbeitenden LNCS-Bandes an. Wird auch ein `<end>`-Wert gesetzt, so werden alle Bände innerhalb der angegebenen Spanne bearbeitet. Ist `<end>` kleiner als `<start>`, so werden die Angaben automatisch vertauscht; eine Bearbeitung erfolgt *immer* vom kleinsten zum größten Band, niemals rückwärts.

Die derzeit einzig mögliche Option (`<options>`) ist
`-q` bzw. `-quiet`
Diese Option unterdrückt die Ausgabe von Text.

Konfiguration
In der `config.xml`-Datei können im Abschnitt "`RecordsHandler`" einige globale Voreinstellungen für die verwendeten Pfade getroffen werden, welche Tabelle A.4 in Abschnitt A.3.1 zu entnehmen sind. Es ist zu beachten, dass sämtliche Fusions-Modi von diesem speziellen Kommando intern vergeben werden (der Schlüssel 'ee' wird auf "merge" gesetzt, während alle übrigen Schlüssel den Wert "ignore" erhalten) und die entsprechenden Einstellungen der Konfigurationsdatei für das `fixLNCS`-Kommando irrelevant sind.

Beispiele

```
> java fixLNCS 2500        // bearbeitet nur den LNCS-Band 2500
> java fixLNCS 2500 2600   // bearbeitet die insgesamt 101 Bände 2500 bis 2600
```

Je nach Größe der angegebenen Bearbeitungsmenge kann die Software erhebliche Zeit in Anspruch nehmen, da jeder Band zuvor mittels der Wrapper-Software eingelesen und anschließend mit Hilfe der Merger-Software fusioniert werden muss.

A.4 Hilfskommandos

Neben den zuvor beschriebenen Kommandos, die die in dieser Arbeit vorgestellten Aufgaben erledigen, existieren zwei weitere Kommandos, deren Aufgabe es ist, die Korrektheit der Wrapper und Merger nach einer Veränderung der Software zu überprüfen. Diese beiden Kommandos sollen nun abschließend vorgestellt werden.

A.4.1 `test_get` – Überprüfung der Wrapper-Software

Das `test_get`-Kommando hat die Syntax

```
> java test_get [<start> [<end>]]
```

und dient der Überprüfung der Wrapper. Wird es ausgeführt, so wird zunächst die Datei `config/testcases.txt` eingelesen. Diese enthält in jeder Zeile Parameter zum Aufruf des `get`-Kommandos (siehe Abschnitt A.2.1), wie beispielsweise:

```
http://www.sciencedirect.com/science/journal/09659978 18
```

Die Testsoftware konstruiert aus diesen Zeilen entsprechende `get`-Aufrufe und führt diese aus. Mittels `<start>` und `<end>` können numerische Werte angegeben werden, ab welcher bzw. bis zu welcher Zeile die Testfälle bearbeitet werden sollen.

Die Ergebnisse des Tests werden in ein Unterverzeichnis `testcases` geschrieben, das, sollte es nicht existieren, automatisch erstellt wird. Die einzelnen Dateien tragen jeweils den Dateinamen `result_XXX`, wobei `XXX` eine dreistellige Zahl mit führenden Nullen ist, die der Zeile in der Datei `testcases.txt` entspricht.

Zur besseren Übersichtlichkeit wurde die Datei `testcases.txt` zudem mittels Kommentarzeilen (beginnend mit einer Raute: `#`) in einzelne Abschnitte unterteilt, die mit dem jeweiligen Namen der Extraktionsquelle sowie laufenden Nummern versehen sind. Diese Zeilen werden, ebenso wie Leerzeilen, nicht mitgezählt und dienen lediglich der Übersichtlichkeit der Datei. Abbildung A.1 zeigt einen Ausschnitt aus dieser Datei. Möchte man hier beispielsweise den Wrapper für die Daten des BMC überprüfen, so lautet der Aufruf des `test_get`-Kommandos:

```
> java test_get 16 20
```

Die auf jene Weise gewonnenen Ergebnisse ($BHT_{c/j}$-Dateien) werden in einem temporären Verzeichnis (derzeit `/tmp/dblp-wrapper/compare`[4] gespeichert und anschließend mit dem neu

[4] Auf der beiliegenden CD-ROM befinden sich eben jene Dateien im Verzeichnis `_compare`.

```
# ACTAPRESS [14-15]
http://www.actapress.com/Content_of_Journal.aspx?journalID=65 26
http://www.actapress.com/Content_of_Journal.aspx?journalID=97#pages 4

# BMC [16-20]
http://www.biomedcentral.com/bmcbioinformatics/archive/ 9
http://www.biomedcentral.com/bmcbioinformatics/archive 9.S1
http://www.biomedcentral.com/bmcbioinformatics 9.S4
http://www.almob.org/ 2 3
http://www.jcheminf.com/ 1

# CAMBRIDGE [21-26]
http://journals.cambridge.org/action/displayJournal?jid=JFP 1
http://journals.cambridge.org/action/displayJournal?jid=JFP 3
http://journals.cambridge.org/action/displayJournal?jid=JFP 9 11
http://journals.cambridge.org/action/displayJournal?jid=JFP 18
http://journals.cambridge.org/action/displayJournal?jid=AIE 21.3 22
http://journals.cambridge.org/action/displayJournal?jid=CPC 14.5
```

Abb. A.1: Ausschnitt der Datei 'testcases.txt': Bei Aufruf des `test_get`-Kommandos werden aus den entsprechenden Zeilen dieser Datei Aufrufe des `get`-Kommandos konstruiert.
Quelle: eigene Erstellung

gewonnenen Ergebnis mittels des Unix/Linux-Shellkommandos '`diff`' verglichen. Treten hierbei Unterschiede auf, so wird eine entsprechende Datei mit Namen `diff_XXX` im Verzeichnis `testcases` erstellt, die das Ergebnis des Vergleichs enthält. Existiert noch keine Datei zum Vergleich, so wird diese aus dem gewonnenen Ergebnis erstellt.

Auf diese Weise ist es möglich, die Wrapper stets auf dem aktuellsten Stand zu halten und ständigen Prüfungen zu unterziehen. Wann immer in der Vergangenheit Probleme aufgetreten sind, wurde ein entsprechender Testfall erstellt und der `testcases.txt`-Datei hinzugefügt. Werden tief greifende Veränderungen an der Software vorgenommen (z.B. an Handler- oder DBLP-Klassen), so sollte anschließend ein kompletter Testlauf erfolgen. Wurden lediglich einzelne Wrapper bearbeitet, so kann man mittels der Parameter bestimmte Testfälle gezielt auswählen. Der Vergleich mit den vorherigen Dateien ist natürlich unabhängig von der Nummer des Testfalls – hier werden normalisierte Dateinamen zur Speicherung der Inhalte verwendet. Das Einfügen neuer Testfälle ist somit stets problemlos möglich.

A.4.2 `test_merge` – Überprüfung der Merger-Software

Ebenso wie die Wrapper-Software getestet werden muss, so sind auch entsprechende Testfälle für die Merge-Software nötig. Hier kann derzeit jedoch nur der Fall der Fusion einer $\mathrm{BHT}_{c/j}$-Datei mit einer Konferenzseite in HTML durchgeführt werden.

Das Kommando hat eine dem Wrapper-Test entsprechende Syntax:

```
> java test_merge [<options>] [<start> [<end>]]
```

Die XML-Datei `config/testcases_merge.xml` beinhaltet entsprechend der in Kapitel 9.2 durchgeführten Studie alle URLs der dort untersuchten Sites. Jeder Testfall wird innerhalb eines XML-Elements "`<case>`" dargestellt und beinhaltet ein Attribut "`key`", über welches er eindeutig identifiziert werden kann. Möchte man nur bestimmte Testfälle überprüfen, so kann dies durch Angabe der entsprechenden Keys als `<start>` und `<end>`-Werte erfolgen. Als `<options>` sind die in Abschnitt A.3.1 angegebenen Fusions-Modi sowie der `-strategy`-Parameter erlaubt.

Wird das Kommando gestartet, so wird entsprechend der erste zu bearbeitende Testfall untersucht. Hierzu wird zunächst ein `get`-Kommando konstruiert, um den Wrapper zu veranlassen, die entsprechende Konferenz aus IEEE Xplore zu extrahieren. Anschließend wird ein `merge`-Kommando erstellt, welches die zuvor erhaltene $\mathrm{BHT_c}$-Datei mit der Konferenz-Website fusioniert.

Beispiel
Der zehnte Testfall innerhalb der `testcases_merge.xml`-Datei lautet

```
<case key="10">
    <name>PerCom</name>
    <year>2006</year>
    <url_ieee>http://ieeexplore.ieee.org/xpl/tocresult.jsp?isnumber=33716&amp;
                                                        isYear=2006</url_ieee>
    <url_conf>http://cnd.iit.cnr.it/percom2006/</url_conf>
    <url_program>http://cnd.iit.cnr.it/percom2006/program.html</url_program>
</case>
```

und beinhaltet offensichtlich die Daten der Konferenz "PerCom" aus dem Jahre 2006. Möchten wir anhand dieses Datensatzes den `HtmlMerger` testen, so können wir dies mit folgendem Kommando tun:

```
> java test_merge 10 10
```

Der Aufruf dieses Kommandos entspricht dabei den hintereinander ausgeführten Aufrufen der beiden folgenden Kommandos:

```
> java get -d=/tmp/dblp-wrapper/htmlMerger/ -filename=ieeeconf_010 -q
  "http://ieeexplore.ieee.org/xpl/tocresult.jsp?isnumber=33716&isYear=2006"
> java merge /tmp/dblp-wrapper/htmlMerger/ieeeconf_010.bht
  http://cnd.iit.cnr.it/percom2006/program.html -a=m -s=m -t=i -p=i -e=i -r=i
```

Mit Hilfe des `test_merge` Kommandos konnten in der Entwicklungsphase der Software die jeweiligen Strategien zur Informationsfusion optimal ausgetestet werden. Ebenso wird es von großem Wert sein, um neue Strategien, die in Zukunft entwickelt werden mögen, auszutesten.

Anhang B

Klassenbeschreibung

Dieses Kapitel soll einen kurzen Überblick über die im Softwarepaket enthaltenen Klassen sowie deren Zusammenspiel bieten. Genauere Informationen sind dem auf der zugehörigen CD-ROM befindlichen Quellcode zu entnehmen.

Sämtliche Programme wurden in Java implementiert; zur Übersetzung der `.java`-Dateien ist ein Compiler einer Version ≥ 1.5 erforderlich. Prinzipiell ist die Software daher plattformunabhängig. Einige Klassen machen allerdings von verschiedenen Shellkommandos unter Unix/Linux-Systemen Gebrauch:

diff Das Kommando `test_get` (vgl. Kapitel A.4.1) benötigt zum Vergleich der Ergebnisse das Shellkommando `diff` und kann auf Windows-Systemen nicht ausgeführt werden.

mv Zur Erstellung der Kopien bestehender Records gemäß den Szenarien F-2, F-2' und F-2_{LNCS} (siehe Kapitel 6.1) benötigt der `FileHandler` (Abschnitt B.3.5) das Shellkommando `mv`, da mit den in Java verfügbaren Methoden lediglich Dateioperationen innerhalb des gleichen Dateisystems möglich sind und die *records* somit nicht unter Erhaltung des `mdates` aus `/dblp/publ` verschoben werden könnten. Unter Windows können daher nur $\text{BHT}_{\text{c/j}}$-Dateien als primäre Quelle verwandt werden.

pdftotext Hierbei handelt es sich nicht um ein Shellkommando, sondern um eine zusätzliche Software, die für Unix/Linux verfügbar ist. Jene verwandelt PDF-Dokumente in Plaintext und wird zur Fusion mit einem Konferenzprogramm in PDF-Format (siehe Kapitel 10.2) benötigt. Unter Windows steht diese Art der Fusion daher derzeit nicht zur Verfügung, könnte aber bei Bedarf durch nachträgliche Anpassung des `pdfHandlers` (Abschnitt B.3.8) hergestellt werden.

Abgesehen von diesen kleinen Einschränkungen ist die Software auch unter Windows vollständig funktionsfähig.

Insgesamt gliedert sich das Softwarepaket in vier Teile: Kommandos, Basis-(Base), Wrapper- und Merger-Klassen. Die Basis-Klassen unterteilen sich wiederum in Handler- und DBLP-Klassen. Abbildung B.1 liefert einen Überblick über die Struktur der Software. Der hierar-

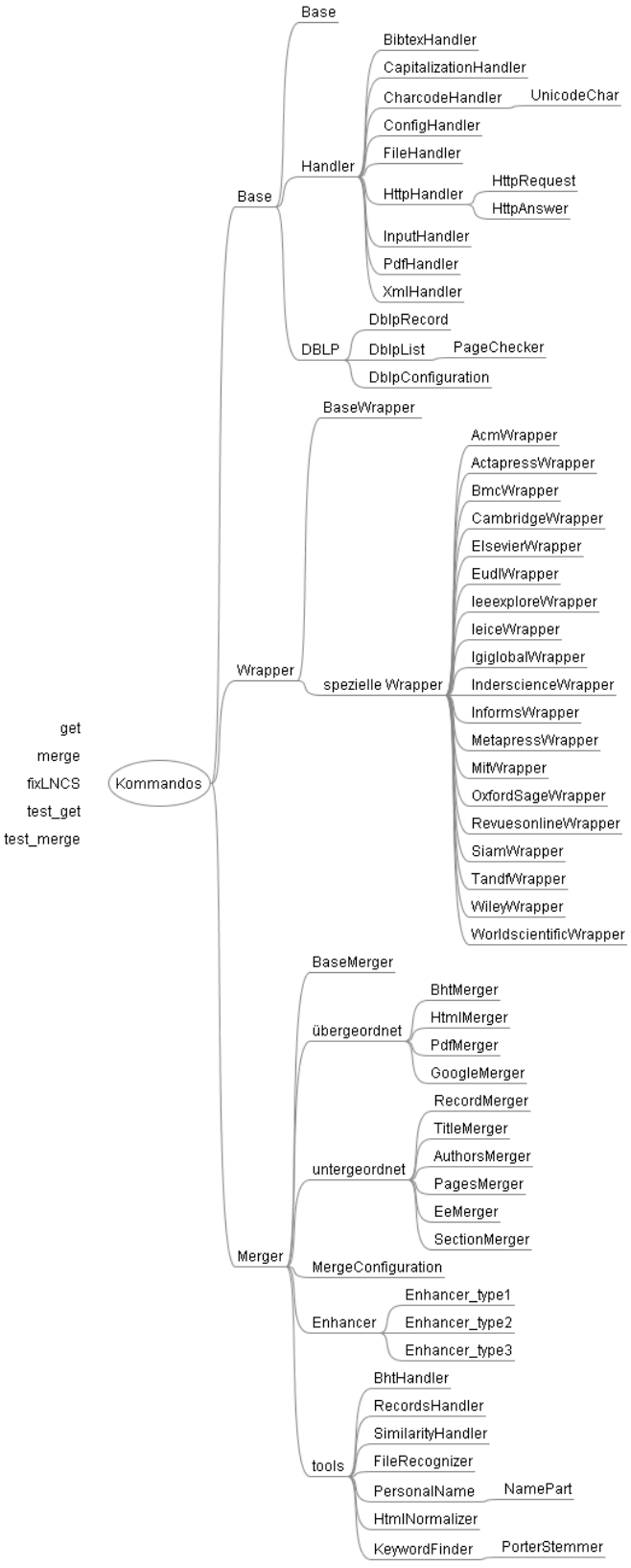


Abb. B.1: Struktur des Softwarepakets

chische Aufbau ist hierbei an die Verzeichnisstruktur der Software angelehnt, unterscheidet sich jedoch an einigen Stellen auch von jener, um einzelne Gruppen funktionell zusammengehöriger Klassen aufzuzeigen. Eine Erläuterung aller Klassen ist den folgenden Abschnitten zu entnehmen.

Zudem sind im Hauptverzeichnis der Software auf der CD-ROM zwei weitere Ordner zu finden:

config In diesem Verzeichnis sind diverse XML- und Textdateien abgelegt, die von den einzelnen Klassen benötigt werden. Wichtigste Datei ist hierbei die `config.xml` (vgl. Kapitel A.1), in welcher sämtliche Voreinstellungen der einzelnen Klassen vorgenommen werden können. Weiterhin findet sich hier die Datei `specialchars.xml` (vgl. Abschnitt B.3.3), die eine große Zahl an Zeichencodes unterschiedlicher Art enthält. `chinese.txt` beinhaltet eine Reihe gängiger chinesischer Nachnamen und wird zur Bestimmung der Namensteile gemäß der in Kapitel 8.3.2 erläuterten Vorgehensweise genutzt. Die beiden Dateien `testcases.txt` und `testcases_merge.xml` enthalten schließlich die Testfälle für die Kommandos `test_get` (Kapitel A.4.1) und `test_merge` (Kapitel A.4.2).

_compare Hier befinden sich zahlreiche Textdateien, bei welchen es sich um 'alte', überprüfte Versionen mittels der Wrapper erstellter $\text{BHT}_{\text{c/j}}$-Dateien handelt. Jene werden bei Ausführung des Kommandos `test_get` (Kapitel A.4.1) verwendet, um Abweichungen der neuen und alten Ergebnisse zu dokumentieren.

Neben den genannten Verzeichnissen benötigt die Software ein Cache-Verzeichnis, in welchem sie vollständige Schreibrechte erwartet. Dies ist derzeit auf `/tmp/dblp-wrapper/` voreingestellt, kann aber in der Konfigurationsdatei (vgl. Kapitel A.1) angepasst werden.

B.1 Die Kommando-Klassen

Die Kommando-Klassen stellen die Schnittstelle zum Benutzer dar. Als einzige aller Klassen besitzen sie eine `main`-Methode und lassen sich daher direkt ausführen. Eine detaillierte Beschreibung der Funktionalität und Aufrufparameter liefert Anhang A.

Jedes Objekt einer Kommandoklasse besitzt einen eigenen `InputHandler` (siehe B.3.7), an welchen es die Kommandozeilenparameter der Benutzereingabe weiterreicht. Entsprechend seiner Aufgabe erstellt es dann anhand der übergebenen Parameter einen Wrapper oder Merger, oder startet ein anderes Kommando; `test_get` beispielsweise ruft seinerseits das Kommando `get` mit entsprechend generierten Parametern, je nach zu überprüfendem Testfall, auf.

Weiterhin sind die Kommandoklassen für die Verarbeitung des jeweiligen Ergebnisses zuständig. So liefert ein Wrapper beispielsweise lediglich einen String zurück, welcher dann von der `get`-Klasse unter dem dort berechneten Dateinamen abgespeichert wird.

B.2 Die Klasse `Base`

Grundlage aller Wrapper, Merger und Handler ist die Klasse *Base*. Sämtliche Klassen – mit wenigen Ausnahmen – sind von dieser Basisklasse abgeleitet und erben somit folgende wichtige Funktionalität:

Konfiguration Die Basisklasse verwaltet einen `ConfigHandler` (siehe B.3.4), welcher zum Einlesen der jeweiligen Konfiguration aus der Datei `config/config.xml` zuständig ist. Jedes abgeleitete Objekt kann somit direkt auf seine eigenen Konfigurationsparamter zurückgreifen.

Ausgabe der Log-Meldungen Die Basisklasse beinhaltet zudem sämtliche Routinen zur Ausgabe der Log-Meldungen. Jede abgeleitete Klasse kann über die Methode `writeToLog`, welcher ein *loglevel* entsprechend Tabelle A.1 auf Seite 179, sowie die entsprechende Meldung zu übergeben ist, Nachrichten unterschiedlicher Priorität definieren. Diese Meldungen werden dann, je nach Konfiguration, während der Programmausführung (standardmäßig auf dem Bildschirm) ausgegeben.

Debugging Die Klasse verfügt zudem über die Methode `debug`, welche bei der Entwicklung der Software äußerst nützlich ist. Der ihr übergebene Wert wird auf dem Bildschirm ausgegeben, danach wird das Programm beendet. Durch Angabe des zusätzlichen bool'schen Parameters *false* lässt sich der Programmabbruch auch verhindern.

B.3 Die Handler-Klassen

Die Handler des Basispakets übernehmen eine Reihe wichtiger, an vielen Stellen benötigter Aufgaben. Gemäß dem Paradigma der Wiederverwendbarkeit kapseln sie wichtige Funktionalitäten und können auch in anderen Softwareprojekten Anwendung finden. In den folgenden Abschnitten soll jeder dieser Handler kurz vorgestellt werden.

B.3.1 `BibtexHandler`

Der `BibtexHandler` erledigt zweierlei Aufgaben. Zum einen ist er in der Lage, BibTeX-Records, wie sie von SIAM (vgl. Kapitel 3.2.11) zur Verfügung gestellt werden, einzulesen und die einzelnen Attributwerte gesondert auszugeben. Hierbei werden auch spezielle in TeX/LaTeX gebräuchliche Zeichencodierungen (beispielsweise "`\"{a}`" für ein "ä") in entsprechende Unicodezeichen transformiert, wozu der Handler den `CharcodeHandler` (siehe Abschnitt B.3.3) nutzt.

Zum anderen können dem `BibtexHandler` beliebige Strings übergeben werden, die häufige in TEX/LATEX verwendete Ausdrücke enthalten, wie beispielsweise "`$n_{\frac{5}{2}}$`". Diese Schreibweise bezeichnet den Term "$n_{\frac{5}{2}}$" und wird in eine in der DBLP-DTD erlaubte Form gebracht, die dem entsprechenden Term möglichst nahe kommt ("`n<sub>5/2</sub>`"). Natürlich ist es wegen der enormen Komplexität der TEX/LATEX-Befehle nicht möglich, alle in Frage kommenden Kombinationen korrekt umzuwandeln, doch i.d.R. können auf diese Weise äußerst zufrieden stellende Ergebnisse erzielt werden.

B.3.2 `CapitalizationHandler`

Der `CapitalizationHandler` ist für die Umwandlung von Strings, die in reiner GROSSSCHRIFT geschrieben sind, zuständig. Autorennamen werden hier mittels fester Regeln, wie sie in Kapitel 4.3.1 im Abschnitt "*normalisiere spezielle Daten*" beschrieben sind, umgewandelt. Für Titel und Zwischenüberschriften nutzt der Handler eine spezielle Heuristik, die aus den derzeit in der `dblp.xml`-Datei vorhandenen Titeln erstellt wird. Wurde jene Datei einmal gelesen und ausgewertet, so schreibt der `CapitalizationHandler` seine Ergebnisse in die Datei `adjustingTitle.txt`, welche im Cache (standardmäßig ist dies "`/tmp/dblp-wrapper/`"), abgelegt wird. Bei jedem nachfolgenden Aufruf wird nun lediglich diese Datei gelesen. Soll sie aus der neusten Version der `dblp.xml` generiert werden, so kann sie einfach gelöscht werden. Der Vorgang der Sammlung jener Daten dauert eine Weile; eine entsprechende Meldung erscheint im Log.

B.3.3 `CharcodeHandler`

Der `CharcodeHandler` ist für sämtliche Umwandlungsvorgänge einzelner Strings in andere Zeichencodierungen zuständig. Er bietet die öffentlichen Methoden `toUTF8`, `ToDBLP` und `toASCII` an, die entsprechende Transformationen durchführen.

Hierzu liest er die Datei `/config/specialchars.xml` ein, welche eine sehr große Anzahl an Einträgen zur Darstellung von Sonderzeichen enthält. Das folgende Fragment zeigt einen Ausschnitt jener Datei:

```
<char code="296">
    <bibcode>\~{I}</bibcode>
    <entity>Itilde</entity>
    <replace>I</replace>
</char>
<char code="297">
    <bibcode>\~{i}</bibcode>
    <bibcode>\~{\i}</bibcode>
    <entity>itilde</entity>
```

```
    <replace>i</replace>
</char>
<char code="298">
    <bibcode>\={I}</bibcode>
    <entity>Imacr</entity>
    <replace>I</replace>
</char>
```

Man erkennt, dass hier jedem Zeichen mögliche TEX/LATEX/BIBTEX-Codes, Entities, sowie alternative Darstellungen zugewiesen wurden. Aus diesen Daten wird eine Liste einzelner `UnicodeCharacter`-Objekte erstellt, die zur Umwandlung eines Strings in das jeweils gewünschte Format verwendet werden.

B.3.4 `ConfigHandler`

Der `CharcodeHandler` ist der einzige Handler, der nicht von der `Base`-Klasse abstammt, da er von jener genutzt wird. Er liest die `config.xml` ein und stellt die Konfigurationen der einzelnen Sections zur Verfügung. Weitere Informationen zur Konfiguration sind Anhang A.1 zu entnehmen.

B.3.5 `FileHandler`

Alle Datei-Operationen können über den `FileHandler` erledigt werden. Die einzelnen Methoden setzen die von den Wrappern und Mergern benötigten Operationen auf dem Filesystem um. Textdateien können gelesen oder geschrieben werden; eine besondere `move`-Funktion ist zudem in der Lage, Dateien innerhalb verschiedener Dateisysteme zu verschieben, indem das Unix/Linux-Shellkommando `mv` verwendet wird (siehe auch die Erklärung zu Beginn dieses Kapitels).

B.3.6 `HttpHandler`

Der `HttpHandler` implementiert den Abruf von Inhalten des WWW mittels des HTTP-Protokolls. Durch Aufruf der simplen `get`-Methode, der ein beliebiger URL übergeben werden muss, liefert dieser Handler den entsprechenden HTML-Quellcode (bzw. andere Daten wie etwa BIBTEX-Records) zurück. Hierzu bedient er sich zweier Hilfsklassen (`HttpRequest` und `HttpAnswer`).

Der `HttpHandler` ist in der Lage, auf zahlreiche in der Praxis auftretende HTTP-Statuscodes[1] zu reagieren. Insbesondere speichert er Cookies und überträgt diese bei Bedarf an den jeweiligen Server, da ohne diese Funktionalität eine große Zahl an Seiten nicht erreicht werden könnten (vgl. die Studie der Extraktionsquellen in Kapitel 3).

Gleichzeitig verwaltet der `HttpHandler` auch einen Cache, in welchem einmal referenzierte Seiten abgelegt und für (derzeit) zwei Tage (Anpassungen sind in der Konfigurationsdatei möglich) aufbewahrt werden. Erneute Anfragen jener Seiten werden entsprechend aus dem Cache bedient. Beim Abruf neuer Seiten von anderen Servern achtet der Handler zudem darauf, stets eine gebührende Zeitspanne (derzeit 5 Sekunden; auch dies kann in der `config.xml` angepasst werden) bis zum nächsten Request einzuhalten.

B.3.7 `InputHandler`

Alle Kommando-Klassen (siehe Abschnitt B.1) nutzen ein Objekt jener Klasse, dem sie die vom Benutzer übermittelte Eingabe weiterleiten. Der `InputHandler` bereitet diese entsprechend auf und liefert auf Anfrage die einzelnen Parameter (entsprechend der Reihenfolge ihrer Eingabe) und Optionen (entsprechend ihrem Namen) zurück. Durch diese Handhabung der Daten ist es dem Benutzer möglich, die Optionen in beliebiger Reihenfolge und an beliebiger Position zu übermitteln. Zudem können Aliase der Optionsnamen definiert werden, die der `InputHandler` automatisch umwandelt. Die folgenden Aufrufe des `get`-Kommandos (vgl. Anhang A.2.1) sind durch jene Behandlung identisch:

```
> java get -filename=abc -quiet http://www.abc.com 8.1
> java get http://www.abc.com 8.1 -filename=abc -quiet
> java get http://www.abc.com -quiet -filename=abc  8.1
> java get -q http://www.abc.com -f=abc  8.1
```

B.3.8 `PdfHandler`

Der `PdfHandler` liest eine Datei im PDF-Format ein. Hierzu transformiert er jene mit Hilfe der Software `pdftotext`[2], die zur Herstellung der Funktionalität des Handlers verfügbar sein muss, in eine reine Textdatei und gibt jenen Text anschließend zurück. Die Qualität des auf diese Weise erzeugten Texts ist, wie in Kapitel 10.2 erläutert, oftmals unzureichend. Der Handler kapselt die Funktionalität jener Umwandlung, so dass an Stelle der genannten Software jederzeit eine andere, die qualitativ hochwertigere Ergebnisse zu liefern in der Lage ist, eingebunden werden kann.

[1] `http://www.iana.org/assignments/http-status-codes`
[2] `http://linux.die.net/man/1/pdftotext`

B.3.9 `XmlHandler`

Der `XmlHandler` liest eine XML-Datei und liefert einen Knoten des DOM-Modells in der in Java üblichen Form zurück. Entsprechende SAX-Funktionen wurden vorbereitet, kommen jedoch im Rahmen der derzeitigen Software nicht zum Einsatz, da die besonderen Vorzüge von SAX (beispielsweise die Möglichkeit, XML-Dokumente nicht sequentiell einlesen zu müssen) derzeit keine Anwendung finden.

B.4 Die DBLP-Klassen

Die DBLP-Klassen bilden das in Kapitel 2.4 definierte Datenmodell innerhalb der Software ab. `DblpList` entspricht dabei einer Record-Liste R^L, `DblpRecord` einem Record R. Die Attribute a werden durch Attribute innerhalb der `DblpRecord`-Klasse repräsentiert.

B.4.1 `DblpRecord`

Bei Erstellung ist ein `DblpRecord` zunächst leer, d.h. alle Attribute haben den Wert *null*. Mittels der `set`-Methoden (`setTitle`, `setVolume`, `setEE` etc.) können beliebige Strings als Attributwerte zugewiesen werden. Das `DblpRecord`-Objekt wandelt jene Strings intern mittels eines `CharcodeHandlers` (siehe B.3.3) in reinen Unicode um, wodurch sämtliche Sonderzeichen ersetzt werden. Anschließend erfolgt eine Prüfung der Werte, sowie die Rückverwandlung in das DTD-konforme Format – ebenfalls durch den `CharcodeHandler`. Dieser Vorgang entspricht der in Kapitel 4.3.1 beschriebenen Normalisierung der globalen und speziellen Daten. Die überarbeiteten Daten werden anschließend in den Attributen des `DblpRecord`-Objektes gespeichert.

Mittels gleich lautender `get`-Methoden (`getTitle, getVolume, getEE etc.`) können jene Werte wieder abgerufen werden. Die Methoden sind dabei äußerst simpel gehalten und geben lediglich den Wert des entsprechenden Attributes aus. Diese Vorgehensweise trägt der Tatsache Rechnung, dass den Records in der Regel nur einmal ein entsprechender Attributwert zugewiesen wird, jener aber u.U. sehr oft referenziert wird.

Bei Erstellung eines neuen `DblpRecord`-Objekts kann diesem ein `DblpConfiguration`-Objekt (siehe Abschnitt B.4.3) übergeben werden, welches diverse Informationen, beispielsweise bzgl. der Reihenfolge der Autorennamen in den nachfolgend übergebenen Strings, enthalten kann.

B.4.2 DblpList

Ein DblpList-Objekt enthält im wesentlichen eine lineare Liste, in welche Objekte des Typs DblpRecord eingefügt werden können. Eine derartige Liste wird zunächst leer erstellt. Im Verlauf der Extraktion oder bei Einlesen einer strukturierten Quelle werden entsprechende Records hinzugefügt. Diese können entweder als Liste angefordert werden, oder aber gleich in einem passenden Ausgabeformat: Die Methode get liefert, je nach übergebenem bzw. in der Konfigurationsdatei eingetragenem Parameter einen String, der dem direkten Inhalt einer $\mathrm{BHT}_{\mathrm{c/j}}$-Datei entspricht.

Mittels der Übergabe eines DblpConfiguration-Objektes werden auch hier interne Parameter gesetzt. So wird auf diese Weise beispielsweise festgelegt, ob es sich bei den eingetragenen Records um Daten eines *journals* oder einer *conference* handelt, was wiederum das Ausgabeformat beeinflusst.

Vor Ausgabe der Daten überprüft das DblpList-Objekt eigenständig die Reihenfolge der enthaltenen Records mit Hilfe einer PageChecker-Klasse. Bei jener Klasse handelt es sich um ein Tool, welches die *pages*-Attribute der DblpRecords auf Linearität hin überprüft. Sind diese linear, jedoch unsortiert, so bringt der PageChecker sie in die korrekte Reihenfolge. Der Aufruf der get-Methode liefert also automatisch stets ein syntaktisch korrektes und, sofern möglich, nach Seitenangaben sortiertes Datenformat.

Eine Ausgabe des Typs xml ist ebenfalls bereits vorbereitet. Wird dies gewählt, so liefert die get-Methode die Daten in XML-Syntax, mit zur DBLP-DTD konformen Tags, zurück.

B.4.3 DblpConfiguration

Das Konfigurationsobjekt enthält eine Reihe fester Konstanten, die von den Record- und Listen-Objekten genutzt werden. Zudem lassen sich hier, je nach Wrapper, feste Vorgaben bzgl. der zu erwartenden Daten machen. Dabei sind folgende Angaben möglich:

```
capitalizedTitles
capitalizedSections
capitalizedAuthors
capitalizedAuthorParts
nametype
authorSeparator
coupledInitials
bibTexParts
removeBackslashes
```

Bei den meisten dieser Attribute – genauer gesagt allen außer nametype und authorSeparator – handelt es sich um bool'sche Variablen.

Die `capitalized`-Attribute können gesetzt werden, wenn die entsprechenden Daten häufig in Großbuchstaben vorliegen. Die Records werden dann jeden entsprechenden Datensatz einer eingehenden Prüfung unterziehen. `capitalizedAuthors` steht hier für einen komplett in Großbuchstaben geschriebenen Namen (z.B. "MING LEE"), während `capitalizedAuthorsParts` darauf hinweist, dass i.d.R. nur einzelne Namensteile in Großbuchstaben vorliegen (z.B. "Ming LEE"), wie dies beispielsweise bei Daten der EUDL (vgl. Kapitel 3.2.6) oftmals der Fall ist.

Als `nametype` wird eine der Konstanten `NAMETYPE_PN`, `NAMETYPE_NP`, `NAMETYPE_MIXED` oder `NAMETYPE_UNKNOWN` erwartet, die angibt, in welcher Reihenfolge Vor- und Nachnamen der Autorennamen auftreten (wobei '`P`' für "prename", '`N`' für "name", also den Nachnamen steht). Wird letzterer Wert angegeben, so wird die Software selbständig versuchen herauszufinden, in welcher Form die Namen vorliegen – was in einigen Fällen recht praktisch, oftmals jedoch auch fehlerträchtig sein kann. `authorSeparator` erwartet einen String, der angibt, welche(s) Zeichen zwischen zwei Autorennamen steht/stehen. Bleibt der Wert leer, so wird die Software auch hier versuchen, diesen selbständig zu ermitteln.

In einigen der in Kapitel 3 genannten Quellen werden Initiale zusammengefasst und ohne Punkt dargestellt, also beispielsweise "MJK Müller-Lindemann" statt "M. J. K. Müller-Lindemann". Die Software sollte in einem solchen Fall die Transformation derartiger Namensteile (mehrere Großbuchstaben hintereinander) automatisch durchführen. Bei anderen Quellen kann dies jedoch zu Fehlern führen, beispielsweise, wenn obiger "Ming LEE" entsprechend der asiatischen Konvention, zuerst den Nachnamen zu nennen, als "LEE Ming" geschrieben wird. Dieser Name soll natürlich nicht in "L. E. E. Ming" verwandelt werden. Daher kann das Konfigurations-Attribut `coupledInitials` gesetzt werden; hat es den Wert *true*, so wird obige Transformation durchgeführt, andernfalls nicht.

`bibTexParts` weist die Records an, die erhaltenen Titel und Zwischenüberschriften an einen `BibtexHandler` (siehe Abschnitt B.3.1) zu übergeben, um entsprechende Codierungen zu entfernen. `removeBackslashes` schließlich diente dazu, Daten, die JavaScript-Ausdrücken entnommen waren, aufzubereiten. Hier wurden einfache und doppelte Anführungszeichen mittels eines Backslashs maskiert (`\'` bzw. `\"`), der entsprechend eliminiert werden musste. Dies kam ausschließlich bei der alten Version des `WorldscientificWrappers` zum Einsatz, da dieser die bibliographischen Daten allesamt aus verwirrenden JavaScript-Variablen heraustüfteln musste. Seit der Verlag seinen Internetauftritt umgestaltet hat (vgl. Kapitel 3.2.12) ist dies jedoch nicht mehr notwendig.

B.5 Die Wrapper-Klassen

Die Wrapper-Klassen setzen die Informationsextraktion wie in Kapitel 4 beschrieben um. Jeder spezielle Wrapper erbt dabei allgemeine Methoden des abstrakten `BaseWrapper`s und implementiert eine Reihe eigener Funktionen.

B.5.1 `BaseWrapper`

Die abstrakte Klasse `BaseWrapper` stellt eine Vielzahl häufig verwendeter Methoden zur Verfügung. Eine ausführliche Darstellung der Funktionalität soll an dieser Stelle unterbleiben, da Kapitel C ausführlich die Konstruktion eines neuen Wrappers und die hierzu verwendeten Methoden der Klasse `BaseWrapper` beschreibt.

B.5.2 spezielle Wrapper

Zur Extraktion der bibliographischen Daten aus den in Kapitel 3 vorgestellten Extraktionsquellen (Verlagsseiten und DLs großer Gesellschaften) kommen jeweils spezielle Wrapper zum Einsatz, wie in Kapitel 4.2 nachzulesen ist. Das dieser Arbeit beiliegende Softwarepaket verfügt derzeit über 19 spezielle Wrapper, die in Tabelle B.1 aufgelistet sind.

Name der Klasse	Verwendung für die Quelle(n)	siehe Kapitel
`AcmWrapper`	ACM	3.2.1
`ActapressWrapper`	ACTA Press	3.2.2
`BmcWrapper`	BMC	3.2.3
`CambridgeWrapper`	Cambridge University Press	3.2.4
`ElsevierWrapper`	ScienceDirect (Elsevier)	3.2.5
`EudlWrapper`	EUDL (ICST)	3.2.6
`IeeexploreWrapper`	IEEE Xplore	3.2.7
`IeiceWrapper`	IEICE	3.2.13
`IgiglobalWrapper`	IGI-Global	3.2.8
`InderscienceWrapper`	Inderscience	3.2.9
`InformsWrapper`	informs	Anhang C
`MetapressWrapper`	MetaPress, IOS Press, Springerlink	3.2.10
`MitWrapper`	MIT Press	3.2.13
`OxfordSageWrapper`	Oxford University Press, SAGE	3.2.13
`RevuesonlineWrapper`	Revues online	3.2.13
`SiamWrapper`	SIAM	3.2.11
`TandfWrapper`	Informaworld (Taylor & Francis)	3.2.13
`WileyWrapper`	Interscience (Wiley)	3.2.13
`WorldscientificWrapper`	WorldScientific	3.2.12

Tab. B.1: Übersicht der speziellen Wrapper-Klassen

B.6 Die Merger-Klassen

Mittels der Merger-Klassen ist die Fusion zweier Quellen gemäß den in Kapitel 6.1 beschriebenen Szenarien möglich. Die einzelnen Klassen lassen sich hier, je nach ihrer Aufgabe, in übergeordnete (Abschnitt B.6.3) und untergeordnete (Abschnitt B.6.4) Merger, Enhancer (Abschnitt B.6.5) sowie Merge-Tools (Abschnitt B.6.6) untergliedern. Zudem existiert auch hier – analog zu den Wrappern (siehe Kapitel B.5) – eine abstrakte Elternklasse (`BaseMerger`, Abschnitt B.6.1), von welcher jedoch nur die *übergeordneten* Merger abstammen. Eine Konfigurations-Klasse (`MergeConfiguration`, Abschnitt B.6.2) wird ähnlich der Vorgehensweise bei den DBLP-Objekten (siehe Kapitel B.4) verwendet.

B.6.1 `BaseMerger`

Die abstrakte Basisklasse, von welcher alle übergeordneten Merger (Abschnitt B.6.3) abstammen, stellt grundlegende Funktionen wie die Fusion zweier `DblpList`-Objekte oder die Ausgabe des 'Mergelogs' bereit. Zur Fusion beider Listen wurde hier der entsprechende Partnersuche-Algorithmus (siehe Kapitel 7.2) auf Ebene der Records implementiert, mit dessen Hilfe Paare von `DblpRecord`-Objekten beider Listen gefunden werden können. Die direkte Fusion jener beiden Objekte erledigt der `RecordMerger` (Abschnitt B.6.4).

Das *Mergelog* beinhaltet Informationen zu allen gefundenen Unterschiedenen der verglichenen Daten – sofern für das entsprechende Attribut nicht der Fusions-Modus `ignore` gewählt wurde (zu den Fusions-Modi siehe Kapitel 6.2.3). Es wird stets auf dem Bildschirm angezeigt und wird von den beiden auffälligen Blöcken

```
++++++++++++
+ Mergelog +
++++++++++++
```

und

```
+++++++++++++++++
+ Mergelog end +
+++++++++++++++++
```

umschlossen. Das Mergelog wird immer erst am Ende des Fusionsvorgangs ausgegeben; die Sortierung der Meldungen entspricht der Reihenfolge der Records der primären Quelle. Es dient sowohl der Information über den vollzogenen Fusionsvorgang, als auch der manuellen Nachbearbeitung des Ergebnisses.

Die einzelnen Einträge des Mergelogs haben einen Aufbau, welcher am folgenden Beispiel erklärt werden soll:

```
#25
[1]S. Prakash Ponnaluri / [Satya Prakash Ponnaluri]
[2]S. G. Wilson / Stephen Wilson -> [Stephen G. Wilson]
p:  [123-127] / 0-

#26
[Multi-path Self-routing Switching Structure.]
 Multi-Path Self-Routing Switching Structure.
       ^          ^
[3]Hui-yao An / [Hui-Yao An]
[5]Bin-qiang Wang / [Bin-Qiang Wang]
[7]- / [Xi Chen]
p:  [129-133] / 0-
```

Die Nummer vor jedem Block (`#25`, `#26`) gibt die Position des Records innerhalb der primären Quelle – und somit auch innerhalb des Ergebnisses – an. Sodann erfolgt eine Auflistung der Attribute, welche sich innerhalb beider Partner-Records unterscheiden. Autoren werden mit einer voranstehenden Zahl (im ersten Block `[1]` und `[2]`, im zweiten Block `[3]`, `[5]` und `[7]`) gekennzeichnet, die die Position des Namens innerhalb der Autorenliste angibt. Der erstgenannte Name tritt hierbei in der primären Quelle, der zweite in der sekundären Quelle auf. Der Name, welcher ins Ergebnis übernommen wurde, ist in eckige Klammern gesetzt. Wurden beide Namen zu einem neuen Namen fusioniert, so wird dies wie im Beispiel "`[2]S. G. Wilson / Stephen Wilson -> [Stephen G. Wilson]`)" dargestellt.

Ein `p` zu Beginn der Zeile weist auf unterschiedliche Seitenangaben hin, ein `ee` (nicht im Beispiel zu sehen) auf verschiedene EE-Attribute. Unterscheiden sich die Titel, so werden sie wie im Beispiel untereinander geschrieben, wobei die Stelle der ersten Differenz – bei gleich langen Titeln auch die jeder weiteren – markiert ist. Ebenso wird bei unterschiedlichen Zwischenüberschriften verfahren. Der ins Ergebnis übernommene Wert ist auch hier mittels eckiger Klammern markiert.

Je nach Fusions-Modus kann es sein, dass ein Wert ins Ergebnis übernommen wird, der nach Berechnung der Fusion als nicht optimal erscheint. In einem solchen Fall wird eine entsprechende Meldung im Mergelog ausgegeben:

```
[2][J. Homer] / John Homer <---- second name looks better!
```

Auch eine fehlgeschlagene Fusion wird derart angezeigt:

```
[2][David Homer] / John Homer <---- fusion failed!
```

B.6.2 `MergeConfiguration`

Jeder übergeordnete Merger (siehe B.6.3) erwartet die Übergabe eines derartigen Konfigurationsobjekts, welche vom aufrufenden `merge`-Kommando erzeugt wird. Das Konfigurationsobjekt beinhaltet die Fusions-Modi (siehe Kapitel 6.2.3) gemäß der Eingabe durch den Benutzer oder die Voreinstellungen innerhalb der Konfigurationsdatei. Alle Modi, die nicht explizit auf einer dieser beiden Arten gesetzt wurden, sind standardmäßig auf `ignore` voreingestellt. Auch die Wahl der Enhance-Strategie (siehe Kapitel 10.1) wird hier festgelegt.

B.6.3 Merger (übergeordnet)

Als *übergeordnete* Merger sollen jene bezeichnet werden, die auf der Ebene der beiden Quellen arbeiten, d.h. für das Einlesen der Daten und deren Ausgabe zuständig sind. Hierzu bedienen sie sich entsprechender Handler (sowohl allgemeine Handler, siehe Abschnitt B.3, als auch speziell zur Fusion benötigte Handler, siehe Abschnitt B.6.6), mit deren Hilfe $\text{BHT}_{\text{c/j}}$- oder BHT_{cite}-Dateien eingelesen, HTML-Seiten abgerufen oder Anfragen an Google gestellt werden können.

Jeder übergeordnete Handler erstellt auf diese Weise intern zwei Objekte des Typs `DblpList`, welche dann mit Hilfe der Funktionalität der Elternklasse (`BaseMerger`, siehe Abschnitt B.6) fusioniert werden. Die Namen der übergeordneten Merger deuten jeweils auf die sekundäre Quelle hin; als primäre Quellen sind stets nur BHT-Dateien ($\text{BHT}_{\text{c/j}}$ oder BHT_{cite}) erlaubt. Tabelle B.2 liefert eine Auflistung der verschiedenen übergeordneten Merger sowie der von ihnen behandleten Sachverhalte.

Name der Klasse	Formate der Quellen	siehe Kapitel
`BhtMerger`	BHT ⋈ BHT	6.1
`HtmlMerger`	BHT ⋈ HTML (URL)	10.1
`PdfMerger`	BHT ⋈ PDF	10.2
`GoogleMerger`	BHT ⋈ WWW (Google)	10.3

Tab. B.2: Übersicht der übergeordneten Merger-Klassen

B.6.4 Merger (untergeordnet)

Im Gegensatz zu den Mergern des vorangegangenen Abschnittes liegt die Aufgabe der *untergeordneten* Merger darin, bestimmte bibliographische Objekte des Datenmodells aus Kapitel 2.4 – mit Ausnahme der Record-Listen, denn dies erledigt die Klasse `BaseMerger` (Abschnitt B.6.1) – nach den in den Kapiteln 7 und 8 festgelegten Verfahren miteinander zu fusionieren. Der `RecordMerger` erstellt hierzu Objekte der übrigen untergeordneten Merger-Klassen, da die Fusion zweier Records auf die Fusion ihrer Attribute zurückzuführen ist (vgl. Kapitel 6.2). Jede dieser Klassen enthält eine öffentliche Methode mit dem Namen `merge`, die das entsprechende

Fusionsergebnis, oder, falls die beiden Objekte nicht miteinander fusioniert werden können, den vordefinierten String "`<+++failed+++>`" zurückliefert.

Zur Ähnlichkeitsbestimmung nutzen diese Klassen einen entsprechenden `SimilarityHandler`, der zu den Merge-Tools (Abschnitt B.6.6) gerechnet wird. Der `AuthorsMerger` verwendet zudem die Tool-Klasse `PersonalName`, die an gleicher Stelle erläutert wird. Tabelle B.3 bietet einen Überblick über die untergeordneten Merger sowie Verweise auf die mit deren Hilfe umgesetzten Konzepte. Für alle übrigen Attribute sind keine Merger notwendig, da diese bei der Fusion zweier Records unberücksichtigt bleiben (siehe Kapitel 8.1.6).

Name der Klasse	Formate der Quellen	siehe Kapitel
`RecordMerger`	$R \bowtie R$	8.2
`TitleMerger`	$title \bowtie title$	8.1.2
`AuthorsMerger`	$authors \bowtie authors$	7
	$aname \bowtie aname$	8.3
	$n \bowtie n$	8.3.5
`PagesMerger`	$pages \bowtie pages$	8.1.3
`EeMerger`	$ee \bowtie ee$	8.1.4
`SectionMerger`	$section \bowtie section$,	
	$subsection \bowtie subection$,	
	$subsubsection \bowtie subsubection$	8.1.5

Tab. B.3: Übersicht der untergeordneten Merger-Klassen

B.6.5 Enhancer

Die Enhancer-Klassen setzen die in Kapitel 10.1 beschriebenen Enhance-Strategien um. Je nach Wert des entsprechenden Attributes innerhalb des `MergeConfiguraton`-Objekts wird hier eine der drei verfügbaren Klassen ausgewählt.

Jedes Enhance-Objekt verfügt hierbei über eine Methode `enhance`, welcher ein `DblpList`-Objekt (die primäre Quelle) sowie ein String (der Quellcode einer HTML-Seite, ein in Plaintext transformiertes PDF-Dokument oder die HTML-Fragmente einzelner Google-Ergebnisse) übergeben wird, aus welchem mittels der in der Liste enthaltenen `DblpRecord`-Objekte zusätzliche Informationen extrahiert werden sollen. Die Methode liefert ein `DblpList`-Objekt mit entsprechend der Strategie modifizierten `DblpRecord`s zurück.

B.6.6 Merge-Tools

Zu dieser Gattung zählen all jene Klassen, die von anderen Merge-Klassen genutzt werden, um wiederkehrende Aufgaben zu erledigen:

BhtHandler Dieser Handler transformiert den Inhalt einer $\text{BHT}_{\text{c/j}}$-Datei in eine Record-Liste unseres internen Datenmodells. Der umgekehrte Vorgang ist bereits innerhalb der Klasse `DblpList` (Abschnitt B.4.2) implementiert.

RecordsHandler Entsprechend sorgt jener Handler dafür, dass *records* gelesen und wieder gespeichert werden können. Hierzu wurden die in Szenario F-2' (Kapitel 6.1.3) erläuterten Funktionen implementiert. Der Handler erwartet als Eingabe eine BHT_{cite}-Datei, aus welcher er selbständig die zugehöriten *records* ermittels, einliest und in `DblpRecord`-Objekte transformiert, welche einem `DblpList`-Objekt hinzugefügt werden. Umgekehrt übernimmt er auch die Rücktransformation eines `DblpList`-Objektes in die entsprechenden *records*, wobei die Möglichkeit einer Kopie oder des Überschreibens besteht.

SimilarityHandler Diese Klasse besitzt verschiedene Funktionen zur Bestimmung der Ähnlichkeit zweier bibliographischer Objekte gemäß Kapitel 7.1 und wird entsprechend von den jeweiligen Mergern genutzt. Intern wurden eine Reihe entsprechender Algorithmen implementiert, u.a. der Levenshtein-Algorithmus ([Lev66]), welcher in leicht modifizierter Form der Quelle `http://www.merriampark.com/ldjava.htm` entnommen wurde.

FileRecognizer Diese Klasse liefert den Datentyp einer übergebenen Datei zurück. Bei allgemeinen Formaten wie PDF-, HTML- oder TXT-Dateien wird hierzu lediglich die Endung des Dateinamens untersucht. Bei den BHT-Formaten genügt dies nicht; hier wird ermittelt, um welche Art (BHT_{c}, BHT_{j} oder BHT_{cite}) es sich handelt. Ebenso können *records* korrekt als solche identifiziert werden.

PersonalName / Namepart Ein Objekt des Typs '`PersonalName`' erwartet einen Personennamen in Form eines Strings und zerlegt jenen intern in einzelne `Namepart`-Objekte, denen der entsprechende Typ gemäß Kapitel 8.3.3 zugeordnet wird. Objekte dieser beiden Klassen werden durch den `AuthorsMerger` (Abschnitt B.6.4), aber auch von den verschiedenen Enhancern (Abschnitt B.6.5) genutzt.

HtmlNormalizer Diese Klasse stellt Methoden zur Verfügung, mit welchen einen HTML-Quellcode in Form eines Strings entsprechend der in Kapitel 10.1 skizzierten Vorgehensweise normalisiert wird.

KeywordFinder / PorterStemmer In der Klasse `KeywordFinder` wurde die Funktionalität implementiert, bei Eingabe eines Titels in Form eines Strings die darin enthaltenen Schlüsselwörter zu erhalten, wie in Kapitel 10.1.2 beschrieben. Intern nutzt diese Klasse hierzu den `PorterStemmer`, dessen Quellcode `http://www.ils.unc.edu/~keyeg/java/porter/PorterStemmer.java` entnommen wurde. Der `KeywordFinder` kommt auch bei der Generierung geschickter Suchanfragen an Google (siehe Kapitel 10.3) zum Einsatz.

Anhang C

Tutorial: Konstruktion eines Wrappers

Die Konstruktion eines neuen Wrappers ist dank des vorhandenen Software-Frameworks eine recht einfache Aufgabe, die in wenigen Stunden möglich ist. In diesem Tutorial werden alle hierzu notwendigen Arbeitsschritte anhand eines konkreten Beispiels erklärt.

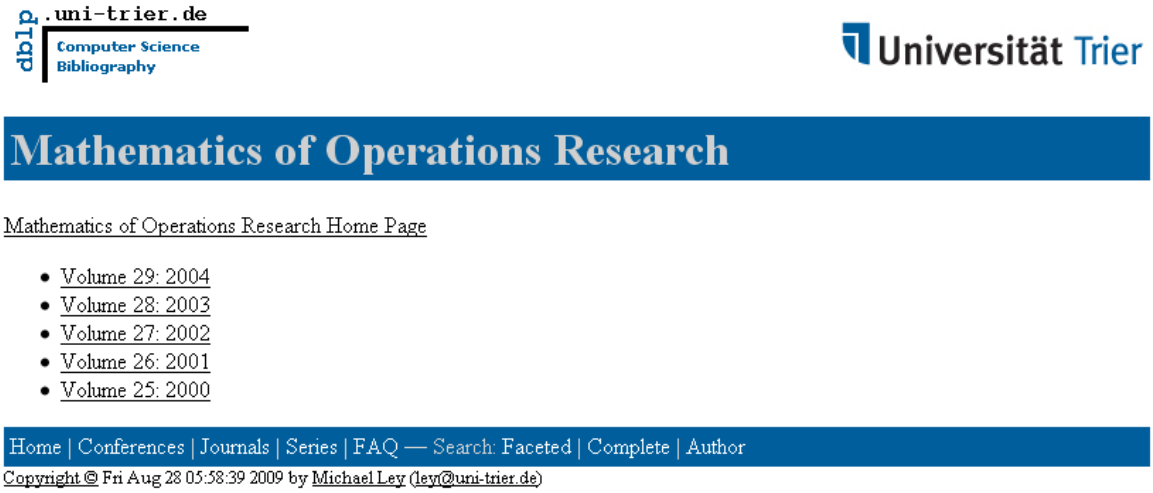


Abb. C.1: In DBLP verfügbare Publikationen des Journals "MOR"
Quelle: `http://dblp.uni-trier.de/db/journals/mor/`, 28.08.2009

Ausgangspunkt ist das Journal "Mathematics of Operations Research", von welchem zum derzeitigen Stand (28. August 2009) die Bände 25-29 bereits in DBLP erfasst sind.[1] Wie man Abbildung C.1 entnehmen kann, ist letztgenannter Band bereits aus dem Jahre 2004, d.h. seither wurden keine neuen Daten hinzugefügt. Wir möchten dies beheben, indem wir einen Wrapper für jenes Journal konstruieren.

[1] siehe `http://dblp.uni-trier.de/db/journals/mor/`

C.1 Studie des Verlagsservers

Folgt man dem Link zur Homepage, so gelangt man zum URL
`http://mor.pubs.informs.org/` (vgl. Abb. C.2).
Man erkennt, dass das entsprechende Journal unter einer Subdomain liegt, die dem "Institute for Operations Research and the Management Sciences" (kurz "informs") gehört. Bevor wir uns auf die Konstruktion des Wrappers stürzen, möchten wir uns die entsprechenden Websites jener Organisation näher betrachten um herauszufinden, ob dort evtl. noch weitere für unsere Zwecke interessante Zeitschriften oder Konferenzen gefunden werden können.

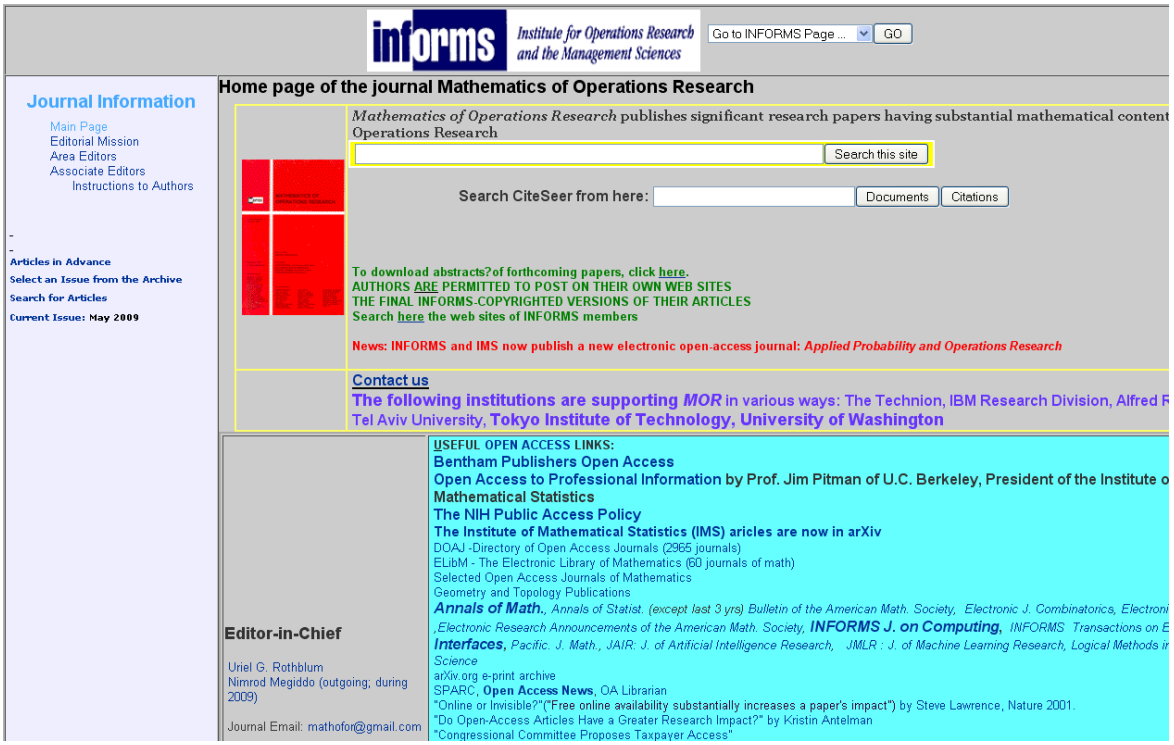


Abb. C.2: Homepage des Journals "Mathematics of Operations Research"
Quelle: `http://mor.pubs.informs.org/`

Ein Klick auf das Banner im Kopf der Seite führt uns zur Startseite des *informs*.[2] Dort findet man im Hauptmenü einen Punkt "PUBLICATIONS", über welchen man zu einer Übersichtsseite publizierter Journale gelangt[3], auf welcher man erfährt, dass die Organisation derzeit zwölf wissenschaftliche Journale publiziert.

Der erste Eintrag des dort befindlichen Menüs ("PubsOnLine (Institutions)") leitet uns nun zu einer Seite, auf welcher all jene Journale, auch das MOR, welches uns zu diesem Server führte, aufgelistet sind (siehe Abb. C.3). Die meisten dieser Journale sind für die Informatik unbedeutend, doch beispielsweise das "INFORMS Journal on Computing" (JOC) ist ebenfalls

[2] `http://www.informs.org/`
[3] `http://www.informs.org/index.php?c=57&kat=PUBLICATIONS`

in DBLP erfasst.[4] Es liegt also nahe, den Wrapper so zu konstruieren, dass er in der Lage ist, alle diese Journale – und somit auch solche, die in Zukunft hinzu kommen könnten – zu erfassen.

Abb. C.3: Die zwölf derzeit verfügbaren Journale des *informs*
Quelle: `http://journal.informs.org/`

Durch Klicken auf eines der Journal-Cover gelangt man auf die jeweilige Startseite des Journals (Siehe Abb. C.4). Wir erkennen, dass sich diese von der zuvor über den Link aus DBLP gefundenen Seite (Abb. C.2) unterscheidet. Jene ist jedoch unter dem dritten Menüpunkt ("Editor-in-Chief") zu erreichen.

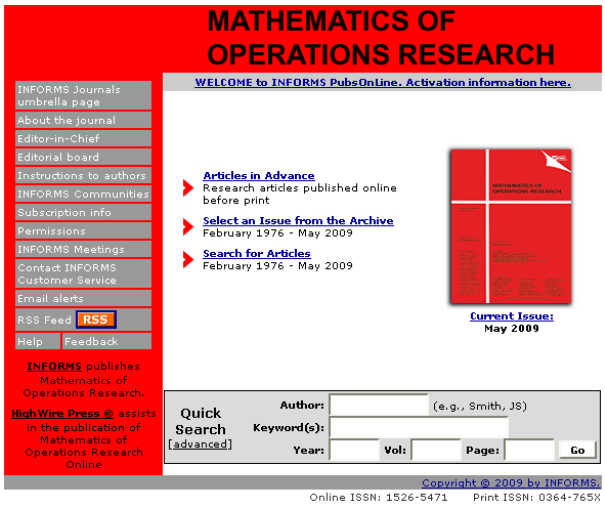


Abb. C.4: Startseite des Journals
Quelle: `http://mor.journal.informs.org/`

[4] `http://dblp.uni-trier.de/db/journals/informs/index.html`

Wählen wir nun den zweiten Punkt des mittleren Menüs (“Select an Issue from the Archive”), so gelangen wir zur Übersicht des Archivs (Abb. C.5). Betrachten wir dessen URL
`http://mor.journal.informs.org/archive/`,
so stellen wir fest, dass

- das Journal unter einer Subdomain von `journal.informs.org` liegt, deren Name dem Akronym des Journals (`mor`) entspricht
- das Archiv in einem Unterverzeichnis `/archive/` untergebracht ist.

Eine Untersuchung der anderen Journale ergibt, dass es sich bei jenen ebenso verhält. Kennen wir also das Akronym eines der Journale, so kennen wir demnach auch den URL von dessen Archivübersicht. Diese Erkenntnis werden wir bei der Konstruktion des Wrappers nutzen, um die Suche stets gleich auf eben jener Seite beginnen zu können.

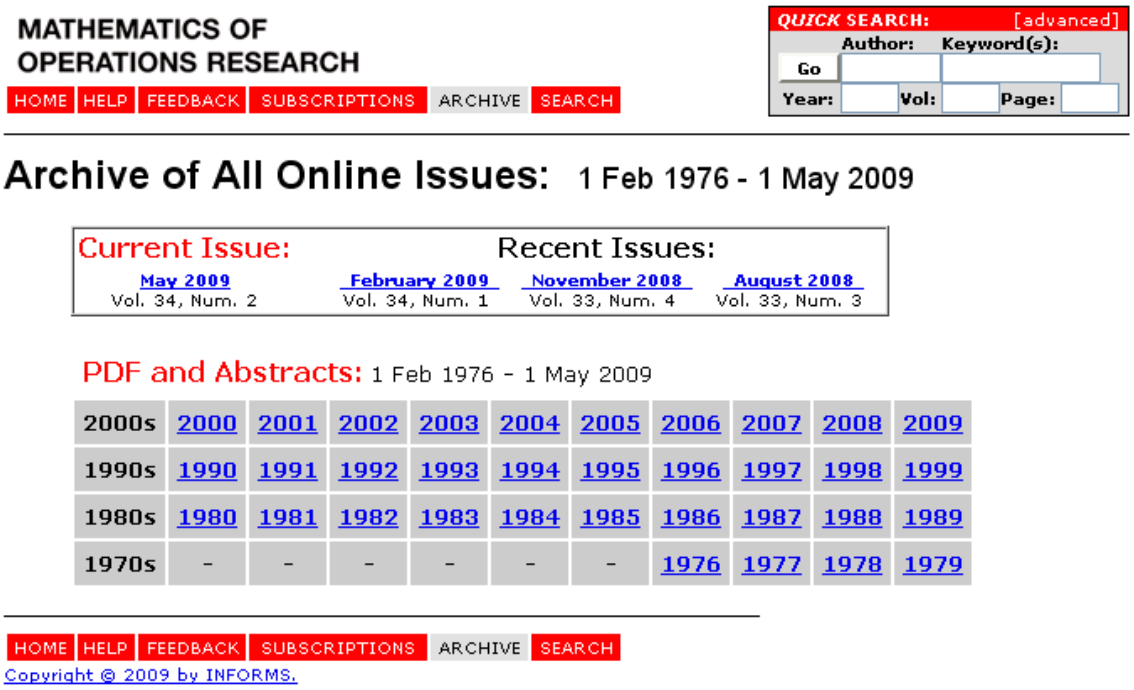


Abb. C.5: Archivübersicht des Journals
Quelle: `http://mor.journal.informs.org/archive/`

Untersuchen wir das Archiv (und auch die Archive anderer Journale) genauer, so können wir einige weitere wichtige Feststellungen machen:

- Jedes Volume entspricht einem Jahrgang (also z.B. Volume 34 $\widehat{=}$ 2009, Volume 33 $\widehat{=}$ 2008,$\cdots$ Volume 1 $\widehat{=}$ 1976).
- Die jeweils aktuellste Nummer ist als “Current Issue” gesondert ausgewiesen (in Abb. C.5 beispielsweise “May 2009 – Vol. 34, Num. 2”).

Hieraus lässt sich schließen, dass wir die Seite, die die für uns relevanten Volumeinformationen enthält, auf einfachste Weise mittels der Formel

$$\text{Jahrgang}_{\text{gesucht}} = \text{Jahrgang}_{\text{aktuell}} - (\text{Volume}_{\text{aktuell}} - \text{Volume}_{\text{gesucht}})$$

errechnen können. Möchten wir beispielsweise Volume 20 bearbeiten, so berechnen wir:

$$\left.\begin{aligned} \text{Jahrgang}_{\text{aktuell}} &= 2009 \\ \text{Volume}_{\text{aktuell}} &= 34 \\ \text{Volume}_{\text{gesucht}} &= 20 \end{aligned}\right\} \Rightarrow \text{Jahrgang}_{\text{gesucht}} = 2009 - (34 - 20) = 1995.$$

Ein Klick auf das entsprechende Jahr führt uns zur Seite
`http://mor.journal.informs.org/archive/1995.dtl`
und bestätigt die Richtigkeit obiger Formel: Dort sind alle Nummern des 20. Volumes aufgelistet (Abb. C.6). Doch wir sehen eine weitere Regelmäßigkeit: Der URL der gesuchten Seite besteht aus dem URL der Archivübersicht und einer Seite, welche sich aus dem gesuchten Jahr (1995) und einer Dateiendung (.dtl) zusammensetzt. Es liegt nahe zu vermuten, dass jeder URL einer Jahrgangsseite sich derart konstruieren lässt, und eine manuelle Überprüfung mehrerer Stichproben bestätigt unsere Vermutung.

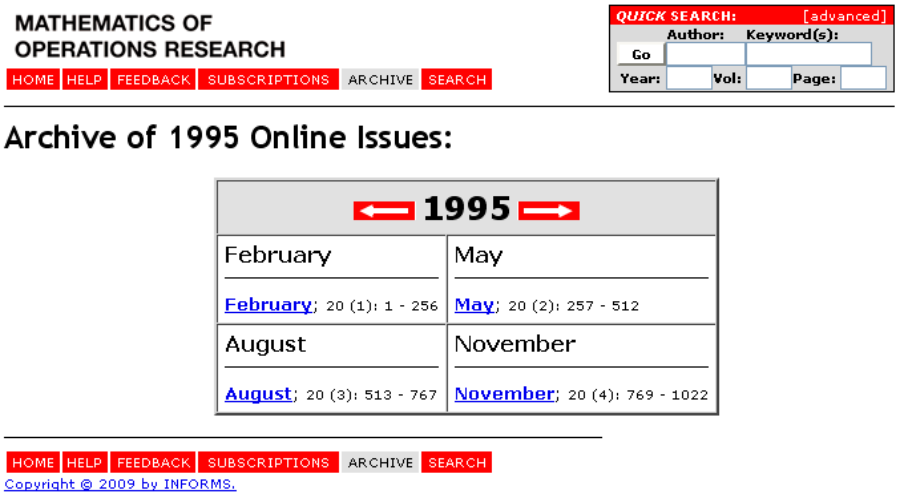


Abb. C.6: Archivseite eines Jahrgangs: Jeder Jahrgang entspricht einem Volume
Quelle: `http://mor.journal.informs.org/archive/1995.dtl`

Das von uns betrachtete Journal 'MOR' erscheint seit 1976 mit vier regelmäßigen Heften (d.h. Issues) pro Jahr, jeweils im Februar, Mai, August und November. Klickt man auf einen der Monatsnamen, so gelangt man zur entsprechenden TOC-Seite, auf der die von uns gesuchten bibliographischen Daten zu finden sind (vgl. Abb. C.7). Im Kopf der Seite sehen wir die Daten für Erscheinungsmonat und -jahr nochmals aufgeführt. Diese sind uns zwar bereits bekannt, für die Bearbeitung mittels des Wrappers ist es jedoch äußerst praktisch, dass wir die Informationen an dieser Stelle wiederfinden. Danach folgt die Liste der Artikel, in welcher die Titel und Autorennamen – meist vollständig und mit einigen Sonderzeichen, Akzenten etc. versehen, was auf eine hohe Datenqualität schließen lässt – sowie die jeweiligen Seitennummern vorhanden sind. Autorennamen werden in der Form 'Vorname(n) Nachname' angegeben. Zwischenüberschriften scheinen nicht zu existieren und konnten auch bei anderen Heften nicht gefunden werden.

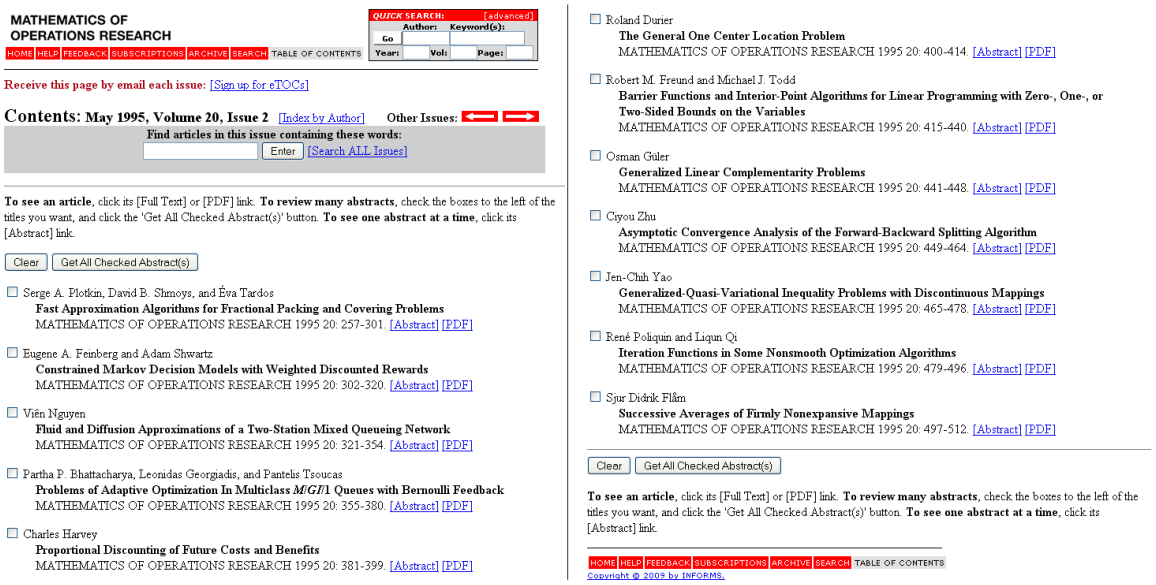


Abb. C.7: TOC-Seite eines Issues: Sämtliche bibliographischen Daten des zweiten Heftes des 20. Bandes von 1995 (links die obere Hälfte der Seite, rechts die untere)
Quelle: `http://mor.journal.informs.org/content/vol20/issue2/index.dtl`

Da wir auf der TOC-Seite jedoch keine DOIs entdecken, überprüfen wir, ob solche auf den Abstract-Seiten verfügbar sind. Und tatsächlich werden wir dort fündig: Abbildung C.8 zeigt den Kopf der Abstract-Seite des ersten Eintrags der zuvor genannten TOC-Seite und enthält den DOI-Eintrag "DOI: 10.1287/moor.20.2.257".

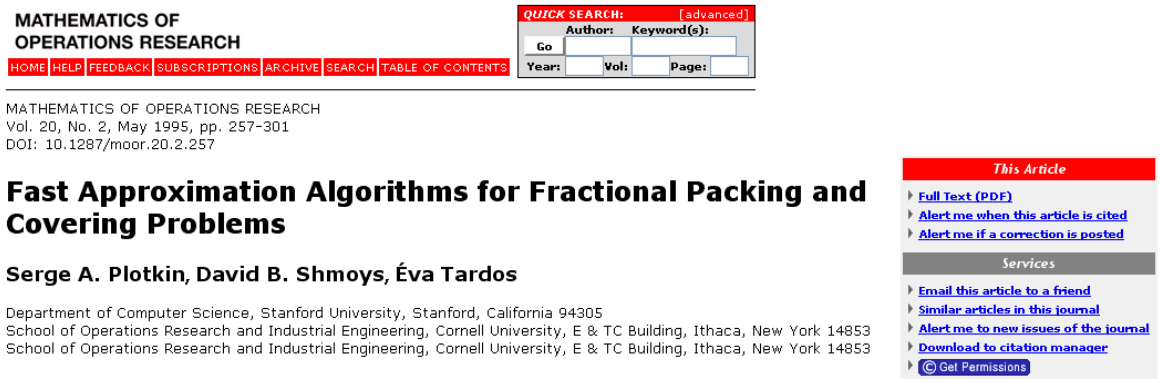


Abb. C.8: Kopfbereich der Abstract-Seite eines Artikels: Hier finden wir einen DOI.
Quelle: `http://mor.journal.informs.org/cgi/content/abstract/20/2/257`

Vergleichen wir nun jenen DOI mit dem URL der Abstract-Seite (`http://mor.journal.informs.org/cgi/content/abstract/20/2/257`), so können wir einige Feststellungen zur Konstruktion des DOIs machen:

- '10.1287' ist ein DOI-Präfix der Organisation (vgl. Aufbau eines DOIs)
- 'moor' ähnelt dem Akronym des Journals (MOR)

- '20.2' scheint für das aktuelle Volume (20) und dessen Issue (2) zu stehen
- '257' ist wahrscheinlich eine fortlaufende Artikelnummer

Wir stellen fest, dass die drei letztgenannten Zahlen unabhängig von ihrer vermuteten Bedeutung auch im URL auftreten. Wenn wir nun noch annehmen, dass der DOI-Präfix sich bei allen Journalen des *informs* nicht ändert – was einer weiteren manuellen, stichprobenartigen Untersuchung anderer Journale/Volumes/Issues bedarf – und die Zeichenfolge 'moor' tatsächlich das Journal bezeichnet – was sich leicht nachprüfen lässt – sind wir in der Lage, einen DOI aus den Informationen, die wir auf der TOC-Seite finden können, zu konstruieren.[5]

Leider ist unsere Freude jedoch nur von kurzer Dauer, denn wir müssen schon bald feststellen, dass diese Methode nicht immer Erfolg verspricht. Bei neueren Bänden (z.B. dem derzeit aktuellen Volume 34, Issue 2) kann der DOI eben *nicht* auf jene Art gewonnen werden. So besitzt beispielsweise der Artikel, dessen Abstract-Seite unter dem URL
`http://mor.journal.informs.org/cgi/content/abstract/34/2/257`
gefunden wird, den DOI
`http://dx.doi.org/10.1287/moor.1080.0372`,
welcher sich in keiner Weise automatisch konstruieren lässt. Wir werden daher beim Extraktionsvorgang überprüfen, ob eine automatische Konstruktion möglich ist und müssen andernfalls stets die Abstract-Seite laden, um den Wert dort direkt auszulesen.

Bisher haben wir noch keine einzige Zeile unseres Wrappers programmiert, jedoch eine Menge nützlicher Informationen gesammelt. Es ist stets äußerst wichtig, die zu erfassenden Seiten zunächst eingehend zu untersuchen, um Regelmäßigkeiten oder Eigenheiten aufzudecken, die uns bei der Datenerfassung behilflich oder hinderlich sein können, um diese entsprechend zu nutzen bzw. zu bewältigen. Nun jedoch können wir mit der eigentlichen Programmierung des Wrappers beginnen.

C.2 Konstruktion der Wrapper-Klasse

Sämtliche Wrapper befinden sich im Unterverzeichnis '`Wrapper`' und tragen den Namen des entsprechenden Informationsanbieters im Namen. Unser neuer Wrapper soll daher den Namen 'InformsWrapper' erhalten. Am einfachsten ist es, wenn wir einen bestehenden Wrapper kopieren und dessen Programmcode anschließend modifizieren, deshalb erstellen wir eine Java-Datei mit dem Namen '`Wrapper/InformsWrapper.java`' und kopieren den Code eines beliebigen anderen Wrappers (in unserem Falle beispielsweise des 'Inderscience'-Wrappers, da dieser in der alphabetischen Liste genau darüber steht) hinein. Natürlich müssen zunächst Klassen- und

[5] Wir benötigen einzig die Information, wie das Journal identifiziert wird, in diesem Falle mittels 'moor'. Diese muss zunächst von einer der Abstract-Seiten gewonnen werden, erst dann besitzen wir alle Informationen. Wichtig ist jedoch, dass es hierzu genügt, nur *eine* Abstract-Seite zu lesen, und nicht die Abstract-Seite jedes Artikels.

Konstruktorname angepasst werden. Danach verfügt unser Wrapper-Gerüst über die folgenden Methoden:

```
public InformsWrapper()
public String get(String source, String range, String out)
protected boolean getKeyByUrl(String source)
protected boolean validateKey()
private boolean setAllVolumes(String range)
private boolean setAllIssues(String vol)
```

Zusätzlich verfügt er über zwei konstante Attribute:

```
private final String URL_BASE
private final String DOI_BASE
```

C.2.1 Definition der URL-Präfix-Konstanten

Zunächst möchten wir letztgenannte Konstanten anpassen. Ihre Werte sollen jeweils dem Präfix sämtlicher auf der Seite gewonnener URLs bzw. DOIs entsprechen. Da wir jedoch aus C.1 wissen, dass die Daten stets unter einer Subdomain von `journal.informs.org` liegen, müssen wir in die `URL_BASE` eine Variable einfügen, die wir stets ersetzen müssen:

```
private final String URL_BASE = "http://[key].journal.informs.org/";
```

Hierzu erstellen wir uns einfach eine Methode '`url_base()`', die jene Ersetzung vornimmt und uns den Präfix mit korrektem Schlüssel zurück liefert. Als Basis der DOIs wählen wir die Standard-Domain:

```
private final String DOI_BASE = "http://dx.doi.org/";
```

C.2.2 Anpassung des Konstruktors

Nun beschäftigen wir uns mit dem Ablauf des Extraktionsvorgangs. Zunächst wird, sobald die Wrapper-Klasse instanziiert wird, wie üblich der Konstruktor aufgerufen. Hier werden verschiedene Initialisierungsaufgaben ausgeführt (z.B. das Logging eingestellt, Konfigurationsdateien gelesen etc.), sowie einige grundlegenden Definitionen getroffen:

```
dblpc.nametype = DblpConfiguration.NAMETYPE_PN;
sourceType = DblpConfiguration.DBLPTYPE_JOURNAL;
```

Die erste Zeile setzt ein Attribut des Konfigurationsobjekts und definiert, dass die Reihenfolge der Autorennamen 'PN' ('prename(s) name', also 'Vorname(n) Nachname') lautet. Die zweite Zeile definiert, dass es sich bei den Daten stets um Journale handelt. Diese beiden Einstellungen genügen uns für den zu konstruierenden Wrapper. Weitere mögliche Werte, die hier gesetzt werden könnten, sind beispielsweise Informationen über Großschreibung von Autorennamen oder Titeln. Für Details siehe Anhang B.4.3).

C.2.3 Ermittlung des Publikationsschlüssels

Die einzige öffentliche Methode eines Wrappers ist die `get()`-Methode. Diese wird mit den folgenden drei Parametern aufgerufen:

- `source`: Der URL, den der Benutzer auf der Kommandozeile angegeben hat
- `range`: Werte für Start- und End-Volume bzw. -Issue
- `out`: Der Typ des Outputs, derzeit stets 'BHT'

Da wir den Wrapper durch 'copy & paste' eines vorhandenen Wrappers erstellt haben, sind auch die wichtigsten Methodenaufrufe und die generelle Struktur der `get()`-Methode bereits enthalten und können unverändert beibehalten werden. Zunächst wird hier die Methode `setPublicationKey` der Elternklasse (`BaseWrapper.java`) aufgerufen. Diese wiederum benötigt die Methoden `getKeyByUrl()` und `validateKey()`, welche wir daher zunächst anpassen möchten.

getKeyByUrl Diese Methode erhält den URL, der im Parameter `source` gespeichert ist. Aus jenem müssen wir den Publikationsschlüssel extrahieren, d.h. einen Wert, der das vom Benutzer gewünschte Journal eindeutig identifiziert. Bei der Studie der Website haben wir gesehen, dass in unserem Fall die Subdomain entscheidend ist. Nehmen wir also an, der Benutzer hätte einen beliebigen URL des MOR-Journals angegeben, z.B.

`http://mor.journal.informs.org/cgi/content/abstract/20/2/257`.

Dann benötigen wir hieraus lediglich den Namen der Subdomain ('`mor`'), d.h. wir generieren einen regulären Ausdruck in Java, der uns eben jenen Wert liefert und setzen auf diese Weise den `publKey`, ein obligatorisches Attribut der Elternklasse.

```
protected boolean getKeyByUrl(String source) {
    String regex = "http://(.*?)\\.";
    Pattern p = Pattern.compile(regex);
    Matcher m = p.matcher(source);
    if (m.find()) {
        publKey = m.group(1);
        return true;
    }
    return false;
}
```

Sollte der reguläre Ausdruck nicht fündig werden, so liefert die Methode `false` zurück, was zu einem Programmabbruch mit entsprechender Fehlermeldung führt.

Nun kennen wir zwar den Schlüssel, den der URL enthielt, doch wissen wir nicht, ob dieser tatsächlich zu einem existierenden Journal gehört. Es wäre ja auch möglich, dass der Benutzer einen falschen Schlüssel angegeben hat. Daher folgt auf die Extraktion des Schlüssels stets dessen Validierung.

validateKey In dieser Methode wird, ebenfalls durch einen Aufruf durch die Elternklasse, der zuvor gewonnene Schlüssel validiert. Hierzu wird die entsprechende Seite des mittels unserer zuvor definierten `url_base()`-Methode gewonnenen URLs geladen. Da es sich in unserem Fall um eine Subdomain handelt, werden wir von der Klasse, welche den HTTP-Dialog mit dem Server bearbeitet (der `HtmlHandler`, welchen wir über das Elternattribut `httph` ansprechen können), eine entsprechende Fehlermeldung erhalten, falls der Schlüssel invalide ist, d.h. die Subdomain nicht existiert. In anderen Fällen – oftmals wird der `publKey` als Parameter im URL übertragen, beispielsweise als `http://www.xyz.com/?volume_id=abc`, wobei hier `abc` dem `publKey` entspricht – erhalten wir im Falle eines invaliden Schlüssels i.d.R. eine Fehlerseite, auf der eine entsprechende Meldung ausgegeben wird, welche wir korrekt interpretieren müssen.

Damit jedoch auch der Benutzer die Möglichkeit hat, die Korrektheit der erkannten Eingabe zu überprüfen, lesen wir in der gleichen Methode zudem einen Publikationstitel (`publTitle`) aus, der bei der Bearbeitung in den Log-Nachrichten angezeigt wird. Da der Name des Journals in unserem Falle stets im `<TITLE>`-Tag des HTML Quellcodes steht, extrahieren wir einfach diesen Wert und speichern ihn im o.g. Attribut – sämtliche weitere Verarbeitung erfolgt innerhalb der Elternklasse und braucht uns nicht weiter zu interessieren.

C.2.4 Sammlung aller URLs des gewünschten Volumes

Nun folgt der erste etwas schwierige Schritt, in welchem wir die URLs sämtlicher Seiten erfassen möchten, von denen nachfolgend die bibliographischen Informationen extrahiert werden sollen.

Dies hat den Vorteil, dass wir uns zunächst rein mit der Navigation über die Seiten beschäftigen werden und uns hiernach lediglich um die Extraktion kümmern müssen.

Zunächst definieren wir die private Methode `setAllVolumes`, welcher wir den vom Benutzer übermittelten Wert der zu erfassenden Volumes/Issues als Parameter `range` übergeben. Dieser darf nicht gleich `null` sein, da bei *journals* stets eine Angabe benötigt wird. In Ausnahmefällen wäre es jedoch zu überlegen, ob auch die Möglichkeit gegeben werden soll, eine einzige Journalseite durch Eingabe des URLs *ohne* entsprechende Bandangaben zu gestatten, doch derzeit ist dies nicht vorgesehen. Die Methode `setAllVolumes` ruft ihrerseits eine Methode `setAllIssues` auf und übergibt dieser das aktuelle Volume. Da in einem Extraktionsvorgang stets nur die Daten eines Volumes erfasst werden (siehe Kapitel 4.3) können wir uns also auf die Erfassung der entsprechenden Seiten eines einzigen Volumes beschränken.

Innerhalb der `setAllIssues`-Methode müssen wir nun Links zu den Seiten der Issues ausfindig machen und diese einer Methode der Elternklasse, `addLink`, übergeben. Auf diese können wir dann beim eigentlichen Extraktionsvorgang über eine verkettete Liste zugreifen.

Um die URLs zu finden, nutzen wir die Erkenntnisse, die wir beim Studium der Website in Abschnitt C.1 gesammelt haben. Wir weisen den Wrapper an, die HTML-Seite der Archivübersicht (`http://[key].journal.informs.org/archive/`) zu laden, wobei wir `[key]` durch den `publKey` ersetzen lassen. Von dieser Seite extrahieren wir die Information des ak-

```
<FONT SIZE="+2" FACE="arial,helvetica"><STRONG>Archive of All Online Issues:</STRONG></FONT>   
<FONT SIZE="+1" FACE="arial,helvetica">1 Feb 1976 - 1 May 2009</FONT>
<DL><DD>
        <TABLE CELLPADDING="0" CELLSPACING="0" BORDER="1"><TR><TD ALIGN="CENTER" VALIGN="MIDDLE">
        <TABLE CELLPADDING="2" CELLSPACING="0" BORDER="0">
        <TR>
                <TH ALIGN="CENTER" VALIGN="BOTTOM"><FONT FACE="verdana,arial,helvetica" COLOR="#FF0101">Current Issue:</FONT></TH>
                <TD WIDTH="15"></TD>
                <TH COLSPAN="5" ALIGN="CENTER" VALIGN="BOTTOM"><FONT FACE="verdana,arial,helvetica">Recent Issues:</FONT></TH>
        </TR><TR>
                <TD ALIGN="CENTER" VALIGN="BOTTOM" NOWRAP>

                        <FONT FACE="verdana,arial,helvetica" SIZE="-2">
                        <STRONG><A HREF="/content/vol34/issue2/index.dtl">May 2009</A></STRONG><BR>
                        Vol. 34, Num. 2
                </FONT></TD>
                <TD WIDTH="15"></TD>
```

Abb. C.9: Ausschnitt des HTML-Quellcodes der Archivübersicht (vgl. Abb. C.5)
Quelle: `http://mor.journal.informs.org/archive/`

tuellen Volumes und dessen Jahrgangs (es könnte ja sein, dass die Publikation eines Journals mittlerweile eingestellt wurde und der Jahrgang des letzten Bandes nicht dem aktuellen Jahr entspricht) mittels des folgenden regulären Ausdrucks, den wir nach Betrachtung des entsprechenden HTML-Codes (vgl. Abb. C.9) ermittelt haben:[6]

```
regex = "Current Issue:.*?"
        + "<A HREF=\"/content/vol(\\d+)/[^>]*>\\w*\\s+(\\d{4})</A>"
```

Dieser reguläre Ausdruck besagt:

[6]Im Folgenden wird stets nur der reguläre Ausdruck angegeben; der gesamte Code zur Suche innerhalb des HTML-Quelltextes ähnelt hierbei stets dem zuvor in Abschnitt C.2.3 gezeigten Java-Code.

- Suche die Stelle im Quelltext, an welcher '`Current Issue:`' steht
- Suche das nächste hierauf folgende `<A>`-Tag
- Lese aus dessen `HREF`-Attribut die Nummer des Volumes
- Lese aus dem Text, der innerhalb von `<A>` und `</A>` steht eine vierstellige Zahl

Durch die runden Klammern entstehen somit zwei Gruppen, deren erste die Volumenummer, die zweite den Jahrgang enthält.

Nun berechnen wir nach der zuvor aufgestellten Formel den Jahrgang des gesuchten Volumes. Jene Seite laden wir, indem wir den Key und die Zahl des entsprechenden Jahrgangs im URL `http://[key].journal.informs.org/archive/[jahrgang].dtl` ersetzen.

```
<TD VALIGN="TOP" WIDTH="25%">
        <STRONG><FONT FACE="verdana,arial,helvetica">February</FONT></STRONG>
        <HR WIDTH="100%" NOSHADE SIZE="1" COLOR="#000000">


                        <NOBR><FONT SIZE="-1" FACE="verdana,arial,helvetica">
                        <STRONG><A HREF="/content/vol20/issue1/index.dtl">February</A></STRONG>;</FONT>
                        <FONT SIZE="-2" FACE="verdana,arial,helvetica">20 (1): 1 - 256</FONT>
                        </NOBR>


        </TD>
```

Abb. C.10: Ausschnitt des HTML-Quellcodes einer Jahrgangsseite (vgl. Abb. C.6)
Quelle: `http://mor.journal.informs.org/archive/1995.dtl`

Wieder müssen wir den HTML-Quellcode der entsprechenden Jahrgangsseite betrachten und finden dort Einträge wie den in Abbildung C.10 gezeigten. Demnach können wir den folgenden regulären Ausdruck nutzen, um die URLs zu extrahieren:

```
regex = "<A HREF=\"/?content([^\"]+)\".*?" + vol + "\\s*\\((\\d+)\\)";
```

Wir suchen also nach einem `<A>`-Tag, dessen `HREF`-Attribut mit '`content`', dem evtl. ein Slash vorausgehen darf, beginnt. Natürlich müssen wir hierzu sicher gehen, dass nur die von uns gewünschten Informationen diese Eigenschaft aufweisen, was manuell zu überprüfen ist. Darauf folgend suchen wir die nächste Stelle, an welcher die aktuelle Volume-Nummer, evtl. gefolgt von Leerzeichen, und darauf folgend eine in Klammern stehende Zahl auftritt. Durch die Gruppierung innerhalb des Ausdrucks erhalten wir entsprechend in der ersten Gruppe den relativen Link, in der zweiten Gruppe die Issue-Nummer.

Doch Vorsicht! Hier gibt es hin und wieder Spezialfälle, wie uns Abbildung C.11 zeigt. Die Mai-Ausgabe (Issue 3) des 1980er Jahrgangs des Journals "Operations Research" besteht aus zwei Teilen. Hier zeigt sich das große Problem bei der Erstellung der Wrapper: Während man den Großteil der Seiten recht einfach bearbeiten kann, treten hin und wieder Spezialfälle auf, bei denen die Extraktion dann fehl schlägt. Hier liegt es daran, dass hinter der Issue-Nummer (3) innerhalb der Klammern ein weiterer Text steht, der für uns zwar irrelevant ist, die Extraktion

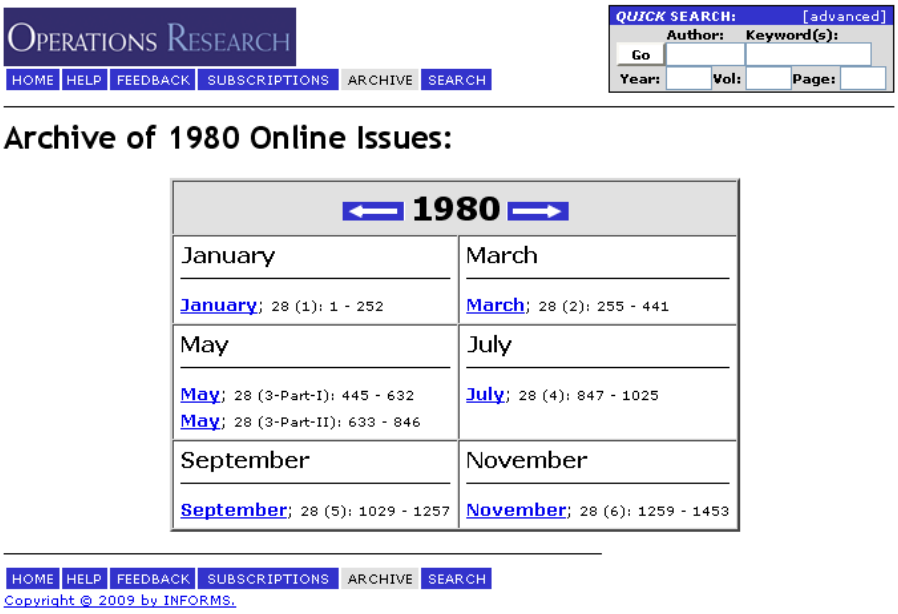


Abb. C.11: Spezialfall: Ein Issue besteht aus zwei Teilen
Quelle: `http://or.journal.informs.org/archive/1980.dtl`

mittels des obigen regulären Ausdrucks jedoch verhindert. Da wir diesen Fall gefunden haben, verändern wir den Ausdruck minimal und haben das Problem gelöst:

```
regex = "<A HREF=\"/?(content[^\"]+)\".*?" + vol + "\\s*\\((\\d+)";
```

Die Veränderung ist auf den ersten Blick kaum erkennbar: Es wurde lediglich die letzte schließende Klammer gelöscht. Damit erfasst der Ausdruck nun alle Zahlen korrekt, besteht jedoch nicht mehr auf die Klammer. Beide Teile werden demnach als ‘Issue 3’ erkannt und die Links extrahiert. Dass nunmehr bei der Bearbeitung von Issue 3 zwei Seiten untersucht werden müssen, stellt kein Problem dar; dies regelt die Basisklasse automatisch.

Auf diese Weise werden nun sämtliche URLs entsprechend ihres Issues gespeichert. Zu debugging-Zwecken kann die Elternmethode `volumeInfo()` genutzt werden, um alle erfassten Daten auszugeben. In unserem Fall liefert diese Methode das folgende Ergebnis:

```
Volume: 20
    Issue 1
        > http://mor.journal.informs.org/content/vol20/issue1/index.dtl
    Issue 2
        > http://mor.journal.informs.org/content/vol20/issue2/index.dtl
    Issue 3
        > http://mor.journal.informs.org/content/vol20/issue3/index.dtl
    Issue 4
        > http://mor.journal.informs.org/content/vol20/issue4/index.dtl
```

C.2.5 Extraktion der bibliographischen Daten

Nachdem wir alle relevanten URLs gesammelt haben, folgt nun der eigentliche Extraktionsprozess der bibliographischen Daten. Hierzu generieren wir mittels

```
LinkedList<HashMap<String, String>> l
        = getLinks(getVolumePart(range), getIssuePart(range));
Iterator<HashMap<String, String>> itl = l.iterator();
```

eine verkettete Liste, die alle für uns relevanten Links enthält, sowie einen Iterator über jene Liste, mit welchem wir die Daten nun verarbeiten möchten. Hierbei ist zu beachten, dass die `getLinks`-Methode die entsprechenden Volume- bzw. Issue-Teile unseres zu scannenden Ranges als Parameter erhält. Dies hat folgenden Grund: Während wir im vorherigen Schritt *alle* verfügbaren Issues aufgenommen haben, werden nun jene Issues ausgewählt, die laut Benutzereingabe auch tatsächlich erfasst werden sollen. Dies können alle verfügbaren sein, – wenn der Benutzer den entsprechenden Rang eingegeben oder aber völlig auf jenen Parameter verzichtet hat – es kann sich jedoch auch bloß um eine Teilmenge jener handeln. Falls die betreffenden Issues gar nicht verfügbar sind (oder nicht gefunden wurden, was auf einen Fehler im Wrapper hinweisen würde), so bleibt die Liste entsprechend leer – und damit auch das Ergebnis.

Zu Beginn der großen `while`-Schleife erfolgen einige Initialisierungen, die einfach so belassen werden sollten. Interessant wird es erst an der Stelle, an welcher die ersten regulären Ausdrücke definiert werden. Gemäß der in Kapitel 4.3 beschriebenen Vorgehensweise werden wir zunächst die globalen Daten erfassen, bevor wir unsere Aufmerksamkeit den eigentlichen Artikeln widmen.

```
<TR>
<TD ALIGN=LEFT VALIGN=TOP NOWRAP>
<STRONG>
<FONT SIZE=+2>Contents:</FONT> <FONT SIZE=+1>February 1995, Volume 20, Issue 1</STRONG></FONT>
          <A HREF="/content/vol20/issue1/aindex.dtl">[Index by Author]</A> 
```

```
<TR>
<TD ALIGN=LEFT VALIGN=TOP NOWRAP>
<STRONG>
<FONT SIZE=+2>Contents:</FONT> <FONT SIZE=+1>May-June 1980, Volume 28, Issue 3-Part-I</STRONG></FONT>
          <A HREF="/content/vol28/issue3-Part-I/aindex.dtl">[Index by Author]</A> 
```

Abb. C.12: Ausschnitte der HTML-Quellcodes zweier TOC-Seiten: Sämtliche Bandinformationen sind leicht auffindbar
Quellen: `http://mor.journal.informs.org/content/vol20/issue1/index.dtl`
`http://or.journal.informs.org/content/vol28/issue3-Part-I/index.dtl`

Wie wir bereits bei der Studie der Site festgestellt haben, werden sämtliche Bandinformationen (also Volume, Issue, Erscheinungsmonat und -jahr) im Seitenkopf noch einmal aufgeführt. Dies ist äußerst praktisch, da wir ansonsten erneut die Übersichtsseite aufrufen und die Daten dort extrahieren müssten. Im Quelltext (vgl. Abb. C.12) sind diese Daten hinter dem zwischen `<FONT>`-Tags stehenden Wort “Contents:” leicht zu finden. Im zweiten Fall (untere Hälfte von Abb. C.12) sieht man, dass auch hier bei der Gewinnung der Issue-Information auf Hefte geachtet werden muss, die aus mehreren Teilen bestehen. Wir können daher nicht den internen

Issue-Wert (in diesem Falle '3') einfach übernehmen, sondern müssen ihn aus der Seite extrahieren (und erhalten in diesem Falle '3-Part-I'). Zudem sehen wir, dass das genannte Heft sich über zwei Monate erstreckt ('May-June'), d.h. wir den gesamten Wert erfassen müssen. Die Normalisierung der Monatsinformationen übernimmt der Wrapper.

Unser regulärer Ausdruck sieht daher wie folgt aus:

```
regex = "<FONT[^>]*>\\s*Contents:\\s*</FONT>\\s*"
    + "<FONT[^>]*>([^\\s]+)\\s+(\\d{4})"
    + ",?\\s*Volume\\s*(\\d+),?\\s*Issue\\s*([^<]+)<";
```

An vielen Stellen wurden hier gewisse 'Unschärfe-Faktoren' eingebaut, um den Ausdruck möglichst flexibel zu halten. So werden beispielsweise durch '`\\s*`' oftmals Leerzeichen erlaubt, obwohl im untersuchten Quelltext an diesen Stellen keine solchen vorhanden sind, oder solche, die vorhanden sind, können in anderen Fällen auch fehlen. Die Kommata sind ebenfalls durch das nachgestellte Fragezeichen nicht verpflichtend. Hinter dem Schlüsselwort `Issue` wird sämtlicher Text bis zum Beginn des nächsten Tags (`<`) erfasst. Es wurde darauf verzichtet, den Namen des nächsten Tags anzugeben, da dieser zum einen nicht relevant, zum anderen im derzeitig erhaltenen HTML-Code auch fehlerhaft ist: Wie man Abbildung C.12 entnehmen kann, sind die Tags `<STRONG>` und `<FONT>` fehlerhaft geschachtelt, die HTML-Dokumente sind daher nicht wohlgeformt.

Da wir die bibliographischen Daten nach dem Prinzip eines HLRT-Wrappers (siehe Kapitel 1.6.2) erfassen möchten, werden wir nun zunächst den Bereich, der die Artikelinformationen enthält, extrahieren. Wir betrachten den Quellcode und stellen fest, dass unmittelbar über und unter dem Datenbereich ein Submit-Button

```
<INPUT TYPE="submit" NAME="sendit" VALUE="Get All Checked Abstract(s)">
```

definiert ist, den wir als Zeichenkette der Abgrenzung benutzen möchten. Zur eindeutigen Identifizierung genügt der `VALUE`, was folgenden regulären Ausdruck bedingt:

```
regex = "VALUE=\"Get All Checked Abstract\\(s\\)\"(.*)"
    + "VALUE=\"Get All Checked Abstract\\(s\\)\"";
```

Nun können wir endlich beginnen, die einzelnen Artikel zu extrahieren. Wir stellen fest, dass wir auch in diesem Fall leichtes Spiel haben: Jeder einzelne Artikel ist zwischen `<DL>`-Tags eingeschlossen (vgl. Abb. C.13).

Wir können die Daten eines einzelnen Artikels daher mit einem äußerst simplen regulären Ausdruck extrahieren:

```
<DL>
<DT><INPUT TYPE="checkbox" NAME="gca" VALUE="20/1/1" ID="hw_mathor_toc_20_1_1">
Martin Schweizer

        <DD><STRONG><label for="hw_mathor_toc_20_1_1">Variance-Optimal Hedging in Discrete Time</label></STRONG>
        <BR>MATHEMATICS OF OPERATIONS RESEARCH 1995 20: 1-32.
        <NOBR>
                <A HREF="/cgi/content/abstract/20/1/1">[Abstract]</A>

                <A HREF="/cgi/reprint/20/1/1">[PDF]</A>


         </NOBR>
<P>
</DL>
<DL>
<DT><INPUT TYPE="checkbox" NAME="gca" VALUE="20/1/33" ID="hw_mathor_toc_20_1_33">
Avi Mandelbaum and William A. Massey

        <DD><STRONG><label for="hw_mathor_toc_20_1_33">Strong Approximations for Time-Dependent Queues</label></STRONG>
        <BR>MATHEMATICS OF OPERATIONS RESEARCH 1995 20: 33-64.
        <NOBR>
                <A HREF="/cgi/content/abstract/20/1/33">[Abstract]</A>

                <A HREF="/cgi/reprint/20/1/33">[PDF]</A>


         </NOBR>
<P>
</DL>
```

Abb. C.13: HTML-Quelltext einzelner Artikel
Quelle: `http://mor.journal.informs.org/content/vol20/issue2/index.dtl`

```
regex = "<DL>(.*?)</DL>";
```

Für die Extraktion der einzelnen bibliographischen Attribute könnten wir nun entweder einen einzigen regulären Ausdruck angeben, oder aber für jedes Attribut einen separaten Ausdruck. Welche Methode wir wählen, ist größtenteils Geschmackssache. In einigen Fällen bietet sich die erste Version an, vor allem wenn die Daten nicht anhand unterschiedlicher Tags, sondern nur durch ihre Position identifiziert werden können. In unserem Fall werden wir drei separate reguläre Ausdrücke konstruieren:[7]

```
regex = "<INPUT[^>]*>\\s*(.*?)\\s*<DD>";                 // authors
regex = "<label[^>]*>\\s*(.*?)\\s*</label>";              // title (siehe C.4.1!)
regex = "<BR>.*?\\s+" + volume + ":\\s*(\\d+(-\\d+)?)"; // pages
```

Anschließend extrahieren wir den URL der Abstract-Seite. Diesen benötigen wir, da wir auf der Abstract-Seite einen DOI suchen und dessen Aufbau mit dem des URLs vergleichen werden. Anschließend werden wir, soweit möglich, die folgenden DOIs direkt konstruieren können.

```
regex = "<A HREF=\"/?([^\"]+)\">\\s*\\[Abstract\\]\\s*</A>";
```

Wie wir aus unserer Studie wissen, benötigen wir zur Konstruktion des DOIs – sofern dies möglich ist – lediglich noch die Zeichenfolge, die das Journal definiert (in unserem Falle lautete

[7] Der reguläre Ausdruck zur Erfassung der Titel unterscheidet sich an dieser Stelle noch von jenem, der letztendlich in der Software implementiert wurde. In Abschnitt C.4.1 werden wir uns noch einmal mit diesem beschäftigen.

diese ‘moor’). Hierzu definieren wir ein entsprechendes Attribut `doi_key`, welches wir zu Beginn mit dem Wert `null` initialisieren. Falls dieses nun also einen `null`-Wert besitzt, so laden wir die entsprechende Abstract-Seite und versuchen, aus dieser den DOI zu extrahieren. Eine Betrachtung des Quelltextes verrät uns, dass der DOI auch in den Metadaten innerhalb des `<head>`-Bereiches angegeben ist. Dort lässt er sich sehr leicht mittels des regulären Ausdrucks

```
regex = "<meta name=\"citation_doi\" content=\"([^\"]+)\"";
```

erfassen. Nun versuchen wir, aus diesem eine entsprechende Zeichenkette zu extrahieren. Gelingt uns dies und entsprechen die nachfolgenden Zahlen dem Verzeichnis des URLs, so speichern wir diese Zeichenkette in `doi_key` und benötigen fortan keine weitere Abstract-Seite. Andernfalls wird die Software jede einzelne Abstract-Seite nach dem jeweiligen DOI absuchen, was natürlich länger dauert, aber dennoch ein absolut zufriedenstellendes Ergebnis liefert.

Wir wir sehen, ist an diesen Stellen oftmals etwas Fingerspitzengefühl notwendig, um die Wrapper effizient und dennoch korrekt zu konstruieren. Oftmals vermutet man Regelmäßigkeiten, die sich jedoch nicht allgemeingültig bestätigen lassen. Wie weit man solche Besonderheiten berücksichtigt, liegt stets im eigenen Ermessen, inwiefern eine sinnvolle Verbesserung möglich ist. Sollten in späteren Testläufen Probleme auftreten, können einzelne Strategien (beispielsweise die hier beschriebene automatische Konstruktion der DOIs) auch völlig verworfen oder noch weiter verfeinert werden. Eine allgemeingültige Strategie existiert hier nicht.

Ein weiteres, bei den Daten des *informs* anscheinend sehr selten auftretendes Problem, ist in Abb. C.14 zu sehen: Ein Titel enthält ein Sonderzeichen, welches mittels eines Bildes angezeigt wird. Auch in einem solchen Fall muss der Wrapper geeignet reagieren. Prinzipiell wird er dieses Bild automatisch gegen ein Warn-Tag ersetzen, welches bei einer manuellen Bearbeitung ins Auge fallen sollte, ansonsten aber spätestens von den weiter verarbeitenden Programmen aufgedeckt würde:

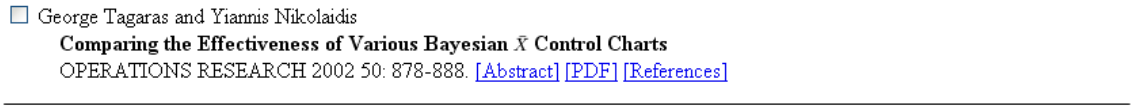
```
<ACHTUNG ! + + + + + + ! Hier stand ein Bildchen ! + + + + + + ! ACHTUNG>
```

In unserem Fall ist es jedoch möglich, das ALT-Attribut des IMG-Tags auszulesen und das Bild gegen den dortigen Wert (ein “X”) zu ersetzen. Hierzu definieren wir eine Methode `replaceImageTags()`, in welcher wir für jeden Titel und jede Zeile mit Autorennamen nach IMG-Tags suchen und diese ggf. ersetzen.

Weitere Probleme, die in einigen Ausnahmefällen gefunden werden können, sind Titel, die mit einer hochgestellten Zahl enden, was wohl auf eine – wenn auch auf der Seite nicht vorhandenen – Fußnote hinweisen soll.[8] Auch diese müssen geeignet verarbeitet werden.

Es ist zu erwarten, dass in der Praxis noch weitere Probleme auftreten werden. Eingehende Tests (vgl. Abschnitt C.4.1) sind daher unerlässlich. An dieser Stelle haben wir jedenfalls

[8] Beispielsweise im vierten Eintrag von `http://or.journal.informs.org/content/vol50/issue1/index.dtl`.


George Tagaras and Yiannis Nikolaidis
Comparing the Effectiveness of Various Bayesian $\bar{X}$ Control Charts
OPERATIONS RESEARCH 2002 50: 878-888. [Abstract] [PDF] [References]

```
        <DL>
        <DT><INPUT TYPE="checkbox" ID="hwTOCGCA_245" NAME="gca" VALUE="50/5/878">
        George Tagaras and Yiannis Nikolaidis

                <DD><STRONG><LABEL FOR="hwTOCGCA_245">Comparing the Effectiveness of Various Bayesian
<I><IMG SRC="/content/vol50/issue5/fulltext/878/f1.gif" ALT="X" BORDER="0"></I> Control Charts</LABEL>
</STRONG>
                <BR>OPERATIONS RESEARCH 2002 50: 878-888.
                <NOBR>
                        <A HREF="/cgi/content/abstract/50/5/878">[Abstract]</A>

                        <A HREF="/cgi/reprint/50/5/878">[PDF]</A>
                        <A HREF="/cgi/content/refs/50/5/878">[References]</A>

                 </NOBR>
        <P>
        </DL>
```

Abb. C.14: Ein Image-Tag innerhalb eines Titels: Oben die Ansicht im Browser, unten der HTML-Quellcode. Das ALT-Attribut beinhaltet eine alternative Darstellung, gegen welche der Wrapper das Bild ersetzen sollte.
Quelle: `http://or.journal.informs.org/content/vol50/issue5/index.dtl`

sämtliche verfügbaren bibliographischen Daten eines Artikels erfasst. Am Ende der Programmschleife wird nun ein neues Objekt des Typs `DblpRecord` erzeugt, welchem daraufhin sämtliche extrahierten Werte hinzugefügt werden. Um deren Normalisierung brauchen wir uns nicht zu kümmern, dies erledigt der Wrapper automatisch. Nachdem wir am Ende der `get`-Methode des Wrappers angelangt sind, ist unsere Arbeit an diesem beendet. Nun müssen wir ihn lediglich in das gesamte Softwarepaket einbinden, und schon kann er in Betrieb genommen werden.

C.3 Einbindung des Wrappers

Nun, da der Wrapper fertig ist, muss er nur noch in die ihn aufrufende `get`-Klasse eingebunden werden. Hierzu sind lediglich zwei simple Erweiterungen innerhalb der Klasse nötig.

C.3.1 Einbindung in die `checkPublisher`-Methode

Zunächst muss anhand des vom Benutzer angegebenen URLs entschieden werden können, von welchem Server die Daten gelesen werden sollen. Hierzu wird in der Methode `checkPublisher`, die als Parameter den vom Benutzer angegebenen URL erhält, ein Attribut `publisher` gesetzt. Dies geschieht mittels einer einfachen `if`-Anweisung, der wir die folgenden Zeilen hinzufügen und damit sämtliche URLs der Domain `informs.org` identifizieren können:

```
...
} else if (domain.endsWith(".informs.org")) {
    publisher = "informs";
...
```

C.3.2 Einbindung in die `getWrapper`-Methode

Nun müssen wir lediglich der `getWrapper`-Methode sagen, welchen Wrapper sie zurück liefern soll, wenn ein URL eingegeben wurde, dem wir den publisher *informs* zuordnen können. Auch dies geschieht in einer simplen `if`-Anweisung, in der wir entsprechende Zeilen ergänzen:

```
...
} else if (publisher.equals("informs")) {
    wr = new InformsWrapper();
...
```

C.4 Abschließende Aufgaben

Nach den zuvor erledigten Aufgaben ist der Wrapper, sofern wir keine Fehler begangen haben, vollkommen lauffähig. Doch der Teufel steckt leider wie so oft im Detail. Es ist daher anzuraten, mit dem fertig gestellten Wrapper eine Reihe von manuellen Tests durchzuführen, um eventuelle Probleme und Unregelmäßigkeiten der Seitenstrukturen aufzuspüren. Zudem gilt es noch, die Dokumentation der `get`-Klasse um den neuen Wrapper zu ergänzen.

C.4.1 Eingehende manuelle Überprüfung des Wrappers

Leider kommt es in der Praxis häufig vor, dass sich der Aufbau der HTML-Seiten nicht in allen Fällen an die von uns ermittelten Regeln hält. Wie wir bereits gesehen haben, nutzt das *informs* mindestens zwei verschiedene Strategien zur Generierung der DOIs. Hätten wir nur den Spezialfall implementiert und angenommen, dass sich die DOIs stets aus den Verzeichnisangaben des URLs konstruieren ließen, hätten wir einen fatalen Fehler begangen. Es ist daher unverzichtbar, den Wrapper nach dessen Fertigstellung eingehend zu überprüfen.

Betrachten wir beispielsweise noch einmal den o.g. regulären Ausdruck zur Extraktion der Titel:

```
regex = "<label[^>]*>\\s*(.*?)\\s*</label>";
```

Dieser Ausdruck ist wie oben beschrieben im Wesentlichen durch die Betrachtung zweier unabhängiger Beispielseiten (Volume 20, Issue 2 und Volume 34, Issue 2) entstanden und extrahiert bei jenen Seiten auch korrekt und vollständig alle Titelinformationen. Es fällt zwar auf, dass das `<label>`-Tag hier im Gegensatz zu allen anderen Tags in Kleinbuchstaben geschrieben ist, doch sollte uns dies prinzipiell nicht stören. Betrachten wir jedoch den Quellcode eines anderen Bandes, beispielsweise Volume 30, Issue 1 aus dem Jahre 2005 (vgl. Abb. C.15 oben), so sehen wir, dass "`<LABEL>`" hier plötzlich ebenfalls groß geschrieben wird. Es empfiehlt sich daher, den obigen regulären Ausdruck stets mittels des Parameters `Pattern.CASE_INSENSITIVE` zu bearbeiten.

```
<DL>
<DT><INPUT TYPE="checkbox" ID="hwTOCGCA_173" NAME="gca" VALUE="30/1/109">
Alexander Shapiro

        <DD><STRONG><LABEL FOR="hwTOCGCA_173">Sensitivity Analysis of Parameterized Variational Inequalities</LABEL></STRONG>
        <BR>MATHEMATICS OF OPERATIONS RESEARCH 2005 30: 109-126.
        <NOBR>
                <A HREF="/cgi/content/abstract/30/1/109">[Abstract]</A>

                <A HREF="/cgi/reprint/30/1/109">[PDF]</A>
                <A HREF="/cgi/content/refs/30/1/109">[References]</A>

         </NOBR>
<P>
</DL>
```

```
<DL>
<DT><INPUT TYPE="checkbox" NAME="gca" VALUE="32/2/345">
Anupam Gupta, R. Ravi, and Amitabh Sinha

        <DD><STRONG>LP Rounding Approximation Algorithms for Stochastic Network Design</STRONG>
        <BR>MATHEMATICS OF OPERATIONS RESEARCH 2007 32: 345-364.
        <NOBR>
                <A HREF="/cgi/content/abstract/32/2/345">[Abstract]</A>

                <A HREF="/cgi/reprint/32/2/345">[PDF]</A>
                <A HREF="/cgi/content/refs/32/2/345">[References]</A>

         </NOBR>
<P>
</DL>
```

Abb. C.15: Beispiele von Unregelmäßigkeiten in der Wahl der HTML-Tags: In beiden Fällen versagt der zuvor ermittelte reguläre Ausdruck
Quellen: `http://mor.journal.informs.org/content/vol30/issue1/index.dtl`
`http://mor.journal.informs.org/content/vol32/issue2/index.dtl`

Doch damit noch nicht genug. Betrachten wir Volume 32, Issue 2 aus dem Jahre 2007 (vgl. Abb. C.15 unten), so fällt uns auf, dass die `<label>`-Tags hier – und nur hier, denn bei allen anderen Issues dieses Bandes sind sie vorhanden – völlig fehlen. In diesem Fall müssen wir uns daher an den `<STRONG>`-Tags orientieren und unseren regulären Ausdruck wie folgt abändern:

```
regex = "<STRONG>(\\s*<LABEL[^>]*>)?\\s*(.*?)\\s*(</LABEL>\\s*)?</STRONG>";
```

Selbstverständlich wissen wir noch immer nicht, ob diese Regel nun in jedem Fall greifen wird, doch wurden bei weiteren Tests keine weiteren Fehler gefunden. Dieses Beispiel soll jedoch verdeutlichen, dass selbst nach eingehender Prüfung stets unerwartete Fehler auftreten können, die nur der Einsatz in der Praxis aufdecken wird.

C.4.2 Eintrag geeigneter Testfälle

Um in Zukunft sicher zu stellen, dass der Wrapper zumindest auf den im Vorfeld getesteten Datensätzen lauffähig ist und bleibt, sollten ein oder mehrere Testfälle in die Datei `config/testcases.txt` eingefügt werden, um diese bei Verwendung des `test_get`-Kommandos (siehe Kapitel A.4.1) automatisch auf Veränderungen zu überprüfen. Es empfiehlt sich hier, keine brandaktuellen Daten anzugeben, die sich in naher Zukunft voraussichtlich verändern werden. Im obigen Fall wäre der Eintrag der Zeile

```
...
http://mor.journal.informs.org/ 34
...
```

beispielsweise zum momentanen Zeitpunkt nicht empfehlenswert, da dieser die Extraktion des gesamten Volume 34, welches derzeit aktuell ist und aus zwei Issues (Februar und Mai 2009) besteht, veranlassen würde. In Kürze werden jedoch weitere Issues folgen (August und November 2009), was natürlich eine Veränderung des Extraktionsergebnisses zur Folge hat. Wir möchten allerdings mittels des `test_get`-Kommandos nur solche Veränderungen erfahren, die auf Fehler der Software oder derzeit unvorhersehbare Änderungen von Seiten des Servers zurück zu führen sind. Dies können generelle Änderung der Struktur sein, durch die der Wrapper nutzlos wird, aber auch Überarbeitungen der bibliographischen Daten von Seiten des Servers. Letzterer Fall ist beim Erstellen der Software beispielsweise bei IEEE mehrmals zu beobachten gewesen, indem die Artikel einiger Journals zunächst mit abgekürzten Autorennamen eingetragen waren, nach einige Wochen jedoch überarbeitet wurden und plötzlich eine höhere Qualität aufwiesen.[9]

Aus diesem Grund fügen wir die folgenden Zeilen in die Datei `config/textcases.txt` ein und überprüfen damit ab Volume 30 bis zum derzeit aktuellsten Heft (Volume 34, Issue 2), sowie die fünf ersten Bände des Journals "MOR". Zudem nehmen wir zwei Testfälle anderer Journale auf, zum einen einige Bände des "JOC", sowie das oben beschriebene Heft des "OR", welches über eine Doppelnummer verfügt:

```
...
# INFORMS [58-61]
http://mor.journal.informs.org/ 30 34.2
http://mor.journal.informs.org/ 1 5
http://joc.journal.informs.org/ 20.4 21.1
http://or.journal.informs.org/archive/1980.dtl 28.2 28.4
...
```

[9] Beispielsweise wiesen die Autorennamen der Artikel des Journals "Systems, Man, and Cybernetics, Part B, IEEE Transactions on"
(`http://ieeexplore.ieee.org/xpl/tocresult.jsp?isYear=2009&isnumber=4802395`)
am 03.07.2009 noch zahlreiche Initialen auf, die am 28.07.2009 gegen vollständige Vornamen ersetzt waren.

Die Nummern in eckigen Klammern helfen lediglich, einzelne Testfälle anzusprechen. Sie müssen von Hand errechnet und in den folgenden Einträgen korrigiert werden. Nun sollte die `test_get`-Software einmal für die vier neuen Einträge gestartet werden, um die entsprechenden Ergebnisse für spätere Vergleiche zu generieren:

```
> java test_get 58 61
```

Die Ergebnisse dieses Testlaufs sollten zudem manuell auf Korrektheit überprüft werden, um den Wrapper bei Bedarf zu korrigieren.

C.4.3 Ergänzung der Dokumentation

Abschließend sollten wir noch die Wrapper-Dokumentation auf den neusten Stand bringen. Hierzu ergänzen wir in der Datei `doku-get.txt` die folgende Zeile im Abschnitt der unterstützten Verlage:

```
...
* informs [key].journal.informs.org
...
```

Dies zeigt einem Benutzer, auf welche Information es bei der Wahl eines URLs ankommt – in unserem Fall auf die jeweilige Subdomain.

C.4.4 Inbetriebnahme des Wrappers

Damit sind alle Aufgaben erfüllt und der neue Wrapper kann in Betrieb genommen werden. Um das oben mehrfach als Beispiel zitierte Volume 20, Issue 2 des Journals "MOR" zu erfassen, starten wir den Wrapper mittels des Kommandos

```
> java get http://mor.journal.informs.org/ 20.2
```

Der Wrapper liefert daraufhin eine Datei '`informs20-2.bht`', deren Inhalt Abbildung C.16 darstellt.

```
<h2>Volume 20, Number 2, May 1995</h2>
<ul>
<li>Serge A. Plotkin, David B. Shmoys, &Eacute;va Tardos:
Fast Approximation Algorithms for Fractional Packing and Covering Problems.
257-301
<ee>http://dx.doi.org/10.1287/moor.20.2.257</ee>
<li>Eugene A. Feinberg, Adam Shwartz:
Constrained Markov Decision Models with Weighted Discounted Rewards.
302-320
<ee>http://dx.doi.org/10.1287/moor.20.2.302</ee>
<li>Vi&ecirc;n Nguyen:
Fluid and Diffusion Approximations of a Two-Station Mixed Queueing Network.
321-354
<ee>http://dx.doi.org/10.1287/moor.20.2.321</ee>
<li>Partha P. Bhattacharya, Leonidas Georgiadis, Pantelis Tsoucas:
Problems of Adaptive Optimization In Multiclass <i>M</i>/<i>GI</i>/1 Queues with
Bernoulli Feedback.
355-380
<ee>http://dx.doi.org/10.1287/moor.20.2.355</ee>
<li>Charles Harvey:
Proportional Discounting of Future Costs and Benefits.
381-399
<ee>http://dx.doi.org/10.1287/moor.20.2.381</ee>
<li>Roland Durier:
The General One Center Location Problem.
400-414
<ee>http://dx.doi.org/10.1287/moor.20.2.400</ee>
<li>Robert M. Freund, Michael J. Todd:
Barrier Functions and Interior-Point Algorithms for Linear Programming with Zero-,
One-, or Two-Sided Bounds on the Variables.
415-440
<ee>http://dx.doi.org/10.1287/moor.20.2.415</ee>
<li>Osman G&uuml;ler:
Generalized Linear Complementarity Problems.
441-448
<ee>http://dx.doi.org/10.1287/moor.20.2.441</ee>
<li>Ciyou Zhu:
Asymptotic Convergence Analysis of the Forward-Backward Splitting Algorithm.
449-464
<ee>http://dx.doi.org/10.1287/moor.20.2.449</ee>
<li>Jen-Chih Yao:
Generalized-Quasi-Variational Inequality Problems with Discontinuous Mappings.
465-478
<ee>http://dx.doi.org/10.1287/moor.20.2.465</ee>
<li>Ren&eacute; Poliquin, Liqun Qi:
Iteration Functions in Some Nonsmooth Optimization Algorithms.
479-496
<ee>http://dx.doi.org/10.1287/moor.20.2.479</ee>
<li>Sjur Didrik Fl&aring;m:
Successive Averages of Firmly Nonexpansive Mappings.
497-512
<ee>http://dx.doi.org/10.1287/moor.20.2.497</ee>
</ul>
<footer>
```

Abb. C.16: Extraktionsergebnis: informs20-2.bht. Alle in Abbildung C.7 gezeigten Datensätze wurden korrekt erfasst und ins BHT_j-Format gebracht.
Quelle: eigene Erstellung

Anhang D

Informationsextraktionsquellen

In diesem Abschnitt sind alle Tabellen abgedruckt, die zur Studie der Informationsextraktionsquellen (siehe Kapitel 3) erstellt wurden.

Tabelle D.1 listet zunächst die URLs der entsprechenden Organisation auf. Trägt deren DL einen abweichenden Namen, so wird dieser in Spalte "Name der DL" genannt. Abschließend wird der URL aufgeführt, unter welchem die entsprechenden bibliographischen Daten zu finden sind.

Tabelle D.2 listet die verfügbaren bibliographischen Daten und deren Qualität auf. Im ersten Bereich wird zunächst angegeben, welche Daten wir mittels eines Wrappers erfassen wollen, wobei sich das Format ('conference', 'journal') auf die in Kapitel 2.2 gegebene Definition bezieht und lediglich auf das Ausgabeformat (BHT_j oder BHT_c) ausgerichtet ist. Im zweiten Bereich der Tabelle sind die jeweils verfügbaren Daten aufgeführt. Hierbei bedeutet ein "+", dass diese Daten stets vorhanden sind, bei einem in Klammern gesetzten "(+)" findet man die Daten nur in einigen Fällen, während ein "-" signalisiert, dass die entsprechenden Daten nie gefunden werden können. Der dritte Bereich schließlich befasst sich mit problematischen Eigenheiten (zusammengefasste Bandnummern, Daten in reiner Großschrift etc.) und verwendet die gleichen Symbole wie der zweite Bereich.

In Tabelle D.3 schließlich sind die technischen Details noch einmal zusammen gefasst. Im ersten Bereich bedeutet das Zeichen "**!**", dass die entsprechende Technik unterstützt werden *muss*, d.h. man ohne die Nutzung dieser Technik keine zufrieden stellenden Ergebnisse erhält. Ein "+" signalisiert, dass die Technik zwar vorhanden ist, jedoch getrost ignoriert werden kann. Wird die entsprechende Technik nicht eingesetzt, so ist das Feld mit einem "-" versehen. Der zweite Bereich der Tabelle listet technische Details wie Server, Übertragungsmodus oder HTML-Version auf. Hierbei wurde jeweils das HTTP-Protokoll untersucht. "k.A." deutet an, dass hier keine Information erhalten werden konnte. In der letzten Spalte schließlich ist das Ergebnis einer Validierung unter `http://validator.w3.org` zu finden, welches bei allen untersuchten Seiten stets negativ ausgefallen ist (-).

	URL der Hauptseite	**Name der DL**	**URL der DL**
ACM	http://www.acm.org	ACM Portal	http://portal.acm.org/dl.cfm
ACTA Press	http://www.actapress.com	–	http://www.actapress.com/journals.aspx
BMC	http://www.biomedcentral.com	–	(variabel)
Cambridge U.P.	http://www.cambridge.org	–	http://journals.cambridge.org
Elsevier	http://www.elsevier.com	ScienceDirect	http://www.sciencedirect.com
ICST	http://www.icst.org	EUDL	http://eudl.eu
IEEE	http://www.ieee.org	Xplore	http://ieeexplore.ieee.org
IEICE	http://www.ieice.org	–	http://search.ieice.org/
IGI-Global	http://www.igi-global.com	–	http://www.igi-global.com/journals
Inderscience	http://www.inderscience.com	–	http://www.inderscience.com
IOS Press	http://www.iospress.nl	–	http://iospress.metapress.com
MetaPress	http://www.metapress.com	–	http://www.metapress.com
MIT Press	http://mitpress.mit.edu	–	http://www.mitpressjournals.org
Oxford U.P.	http://www.oup.com	–	http://www.oxfordjournals.org
Revuesonline	http://www.revuesonline.com	–	http://www.revuesonline.com
SAGE Publications	http://www.sagepub.com	–	http://www.sagepub.com
SIAM	http://www.siam.org	–	http://epubs.siam.org
Springer	http://www.springer.com	Springerlink	http://springerlink.metapress.com
Taylor & Francis	http://www.taylorandfrancis.com	**informa**world	http://www.informaworld.com
Wiley	http://eu.wiley.com	InterScience	http://www.interscience.wiley.com
WorldScientific	http://www.worldscientific.com	–	http://www.worldscientific.com

Tab. D.1: IE-Quellen – allgemeine Informationen

	Conferences (BHT_C-Datei)	Journals (BHT_J-Datei)	Titel, Autor, Jahr	Seitenzahlen	DOIS	Publikationsmonat	Zwischenüberschriften	gruppierte Nummern	aufgespaltene Nummern	mehrere Indexseiten	Sonderbände (Supplements)	Titel in GROSSBST.	Autoren in GROSSBST.	Zwischenüb. in GROSSBST.
ACM	+	+	+	+	+	+	(+)	(+)	-	-	(+)	-	-	-
ACTA Press	-	+	+	-	(+)	-	-	-	-	-	-	(+)	(+)	-
BMC	-	+	+	-	+	+	-	-	-	(+)	(+)	-	-	-
Cambridge U.P.	-	+	+	+	+	+	(+)	(+)	-	-	-	-	(+)	(+)
Elsevier	-	+	+	+	+	+	(+)	(+)	(+)	-	-	-	-	-
ICST	+	-	+	-	(+)	+	-	-	-	(+)	-	(+)	-	-
IEEE	+	+	+	+	+	(+)	(+)	-	(+)	(+)	-	-	-	-
IEICE	-	+	+	+	-	+	+	-	-	-	-	(+)	(+)	-
IGI-Global	-	+	+	(+)	-	-	-	-	-	-	-	-	-	-
Inderscience	-	+	+	+	(+)	-	-	(+)	-	-	-	-	-	-
IOS Press	+	+	+	+	-	(+)	-	(+)	-	(+)	-	(+)	-	-
MetaPress	+	+	+	+	(+)	(+)	-	(+)	-	(+)	-	-	-	-
MIT Press	-	+	+	+	+	+	(+)	-	-	-	(+)	-	-	-
Oxford U.P.	-	+	+	+	+	+	(+)	-	-	-	-	-	(+)	(+)
Revuesonline	-	+	+	+	(+)	-	-	(+)	-	-	(+)	(+)	(+)	-
SAGE Publications	-	+	+	+	+	+	(+)	-	-	-	-	-	-	(+)
SIAM	-	+	+	+	(+)	(+)	-	-	-	-	-	-	-	-
Springer	+	+	+	+	+	(+)	(+)	(+)	-	(+)	(+)	-	-	-
Taylor & Francis	-	+	+	+	+	-	+	(+)	-	-	-	(+)	-	(+)
Wiley	+	+	+	+	+	+	+	(+)	-	-	-	-	-	-
World Scientific	-	+	+	+	(+)	(+)	-	(+)	(+)	-	(+)	(+)	(+)	-

Tab. D.2: IE-Quellen – Datenverfügbarkeit, -Qualität und Besonderheiten

	Skriptsprache	JavaScript bzw. AJAX	Cookies	Frames	Serveridentifikation	Übertragungsmodus	Zeichensatz	HTML-Version (DOCTYPE)	Valider HTML-Code?
ACM	CFML	-	+	-	MS IIS/6.0	k.A.	UTF-8	xhtml 1.0 Transitional	-
ACTA Press	ASP	+	**!**	-	MS IIS/6.0	content-length	UTF-8	xhtml 1.0 Strict	-
BMC	ASP	**!**	+	-	MS IIS/5.0	k.A.	Latin-1	MathML	-
Cambridge U.P.	PHP	**!**	+	-	Apache	chunked	UTF-8	xhtml 1.0 Transitional	-
Elsevier	k.A.	+	+	-	k.A.	content-length	UTF-8	k. A.	-
ICST	PHP	**!**	**!**	-	Apache	chunked	UTF-8	xhtml 1.1	-
IEEE	JSP	+	+	-	"IEEE Webserver"	chunked	Latin-1	k. A.	-
IEICE	PHP	-	-	-	Apache	chunked	Latin-1	xhtml 1.0 Strict	-
IGI-Global	ASP	+	+	-	MS IIS/6.0	content-length	Latin-1	k. A.	-
Inderscience	PHP	+	+	-	Apache	chunked	Latin-1	HTML 4.01 Transitional	-
IOS Press	ASP	+	+	-	MS IIS/6.0	content-length	UTF-8	HTML 4.01 Transitional	-
MetaPress	ASP	+	+	-	MS IIS/6.0	content-length	UTF-8	HTML 4.0 Transitional	-
MIT Press	ASP	+	**!**	-	AtyponWS/7.1	chunked	UTF-8	xhtml 1.0 Transitional	-
Oxford U.P.	Highwire	+	+	-	Apache	chunked	Latin-1	k.A.	-
Revuesonline	JSP	+	+	+	Apache	chunked	Latin-1	HTML 4.01 Frameset	-
SAGE Publications	Highwire	+	**!**	-	Apache	chunked	UTF-8	k.A.	-
SIAM	JSP	**!**	+	-	Sun-Java-System	chunked	Latin-1	xhtml 1.0 Transitional	-
Springer	ASP	+	+	-	MS IIS/6.0	content-length	UTF-8	HTML 4.0 Transitional	-
Taylor & Francis	k.A.	+	+	-	Apache	chunked	Latin-1	xhtml 1.0 Transitional	-
Wiley	k. A.	+	**!**	-	Apache	content-length	Latin-1	xhtml 1.0 Transitional	-
World Scientific (alt)	JSP & SSI	**!**	**!**	+	Apache & MS IIS	chunked	k. A.	k. A.	-
World Scientific (neu)	SSI	+	-	-	Apache	chunked	Latin-1	k. A.	-

Tab. D.3: IE-Quellen – Technische Details

Anhang E

HTML-Konferenzprogramme

Dieser Abschnitt beinhaltet die einzelnen Ergebnisse der in Kapitel 9) durchgeführten Studie der HTML-Konferenzprogramme. Die hier gezeigten Tabellen beinhalten die Ergebnisse diverser Untersuchungen, welche vor und nach der Programmierung der dort beschriebenen Software erzielt wurden. Die einzelnen Testdatensätze wurden hierbei von 1 bis 100 durchnummeriert, entsprechend der Erfassung mittels der in Kapitel 9.2.1 beschriebenen Methode.

Die Tabellen E.1 und E.2 zeigen die jeweiligen URLs der Testdatensätze. Jene sind ebenfalls auf der beiliegenden CD-ROM in der Datei `config/testcases_merge.xml` zu finden.

Die Betrachtungen der Struktur der untersuchten HTML-Seiten sind in den Tabellen E.3, E.4 und E.5 zu finden. Die einzelnen Werte haben folgende Bedeutung:

Anzahl HTML-Seiten gibt an, auf wie vielen einzelnen Seiten das komplette Konferenzprogramm verteilt ist.

Seitenaufbau **L**iste oder **T**abelle. Eine Liste zeichnet sich dadurch auf, dass die Daten untereinander stehen. Tabellen, in welchen die Daten in nur einer einzigen Spalte stehen, wurden daher auch als Listen gewertet.

Irrelevante header/Irrelevante footer Ein '+' gibt an, dass vor/hinter den eigentlichen Daten irrelevante Codeblöcke vorhanden sind, die mittels eines HLRT-Wrappers (vgl. Kapitel 1.6) eliminiert werden könnten. Ein '–' bedeutet demnach, dass keine derartigen Blöcke existieren.

HTML-Version Wert des `<DOCTYPE>`-Tags der Seite. Fehlt dieser, so wurde 'k.A.' (keine Angaben) eingetragen.

HTML-Generator Wert des Metatags `<meta type="generator">`. Fehlt diese Angabe, so wurde 'k.A.' eingetragen.

valides HTML? Hier gibt ein '+' an, dass der HTML-Quellcode einer Validierung mittels des Validators auf `http://validator.w3.org` standgehalten hat, ein '–' zeigt an, dass hierbei Fehler auftraten.

vollständige Namen Enthält das Konferenzprogramm (zumindest einige) ausgeschriebene Autorennamen ('+'), oder wurden nur Initiale verwandt ('–')?

Zwischenüberschriften Lassen sich Zwischenüberschriften erkennen ('+') oder nicht ('–')?

Artikel in Reihenfolge Ein '+' gibt an, dass die Artikel innerhalb des Konferenzprogramms in gleicher Reihenfolge auftreten wie bei IEEE Xplore. Liegen sie in völlig anderer Reihenfolge vor, oder sind sie – was oftmals der Fall ist – nur innerhalb der Sessions in gleicher Reihenfolge, so wurde ein '–' eingetragen.

Anordnung Titel / Autoren Werden erst die Titel und dann die Autoren genannt ('TA'), oder verhält es sich anders herum ('AT'). In wenigen Fällen variiert die Anordnung innerhalb der Seite ('var.').

Anordnung Namensteile Gibt die Reihenfolge der Namensteile an: 'Vorname(n) Nachname' (VN) oder 'Nachname, Vorname(n)' (NV).

Tags innerhalb Titel/Namen Treten hin und wieder HTML-Tags innerhalb der Titel/Autorennamen auf, so wurde '+' eingetragen. Fielen keine derartigen Stellen auf, so wird dies durch ein '–' angezeigt.

T / A mittels Tags getrennt Ein '+' gibt an, dass zwischen Titel und Autorennamen mindestens ein HTML-Tag zu finden ist. Ein '–' zeigt dementsprechend, dass Titel und Autorennamen direkt aneinander anknüpfen; i.d.R. werden sie in einem solchen Fall durch ein Komma oder eine öffnende Klammer getrennt. Die Reihenfolge 'T / A' wurde hier o.B.d.A. angegeben; der Wert gilt auch bei anderer Anordnung.

Dynamischer Code zwischen T / A Sind die Zeichen zwischen Titeln und Autorennamen stets identisch, so wurde hier ein '–' angegeben. Handelt es sich um (zumindest teilweise) verschiedene Inhalte, so erscheint ein '+'.

Schlüssel in Zw.-Überschriften Falls der überwiegende Teil der Zwischenüberschriften mittels eines Schlüsselwortes identifiziert werden kann, so wurde dieses hier eingetragen.

Die Tabellen E.6 und E.7 zeigen abschließend die Ergebnisse der Anwendung der drei Enhance-Strategien auf die Testdatensätze. Im ersten Block sind die in Xplore verfügbaren Datensätze (absolute Anzahl der Records sowie aller darin enthaltener Autorennamen) eingetragen. Der zweite Block zeigt die Anzahl der Records, in welchen Autorennamen gefunden wurden, sowie die konkrete Anzahl der gefundenen Autorennamen mit jeder der drei Methoden. Der dritte Block schließlich zeigt die gleichen Werte in Prozentangaben. Zur Erstellung von Abbildung 10.2 auf Seite 170 wurden die Prozentangaben der Autorenfunde verwandt.

			URL bei IEEE Xplore (http://ieeexplore.ieee.org/...)	URL des Konferenzprogramms
1	COMSWARE	2008	xpl/tocresult.jsp?isnumber=4554360&isYear=2008	http://edas.info/p5573
2	COMSWARE	2007	xpl/RecentCon.jsp?punumber=4267954&conhome=1001172	http://www.comsware.org/2007/techprog.html
3	CCNC	2008	xpl/tocresult.jsp?isnumber=4446298&isYear=2008	http://cms.comsoc.org/CCNC_2008/Content/Home/Conf_Program/Technical_Sessions.html
4	RE	2006	xpl/tocresult.jsp?isnumber=35962&isYear=2006	http://www.ifi.uzh.ch/req/events/RE06/ConferenceProgram/TechnicalProgram.html
5	RE	2003	xpl/RecentCon.jsp?punumber=87258&conhome=10000630	http://conferences.computer.org/RE/main/program.html
6	EEE	2005	xpl/tocresult.jsp?isnumber=30444&isYear=2005	http://www.comp.hkbu.edu.hk/~eee05/program/
7	CEC	2004	xpl/RecentCon.jsp?punumber=92118&conhome=1000221	http://tab.computer.org/tcec/cec04/program.html
8	CEC/EEE	2007	xpl/tocresult.jsp?isnumber=4285176&isYear=2007	http://ieejc.ise.eng.osaka-u.ac.jp/CEC2007/program.html
9	CEC/EEE	2008	xpl/RecentCon.jsp?punumber=4785029&conhome=1000221	http://cec2008.cs.georgetown.edu/schedule.html
10	PerCom	2006	xpl/tocresult.jsp?isnumber=337166&isYear=2006	http://cmd.iit.cnr.it/percom2006/program.html
11	PerCom	2003	xpl/tocresult.jsp?isnumber=26747&isYear=2003	http://www.percom.org/2003/advancedprogram.htm
12	PerCom	2004	xpl/tocresult.jsp?isnumber=28556&isYear=2004	http://www.percom.org/2004/advancedprogram.htm
13	PerCom	2007	xpl/tocresult.jsp?isnumber=4144735&isYear=2007	http://www.cse.psu.edu/~hurson/percom2007/advancedprogram.html
14	HST	2009	xpl/tocresult.jsp?isnumber=5168000&isYear=2009	http://www.ieeehomelandsecurityconference.org/program.htm
15	CCC	2006	xpl/tocresult.jsp?isnumber=34834&isYear=2006	http://www.math.cas.cz/~ccc06/program.html
16	CCC	2008	xpl/tocresult.jsp?isnumber=4558794&isYear=2008	http://www.register123.com/profile/web/index.cfm?PkWebID=0x1106386294&varPage=info
17	ICIP	2008	xpl/tocresult.jsp?isnumber=4711669&isYear=2008	http://www.icip08.org/PapersAbstractSearch.asp?show=search
18	BROADNETS	2007	xpl/tocresult.jsp?isnumber=4550387&isYear=2007	http://www.broadnets.org/2007/wirelessprogram.html
19	RFID	2009	xpl/tocresult.jsp?isnumber=4911159&isYear=2009	http://cms.comsoc.org/eprise/main/SiteGen/RFID/Content/Home/PROGRAM_OUTLINE.html
20	IITC	2009	xpl/tocresult.jsp?isnumber=5080321&isYear=2009	http://www.his.com/~iitc/monday.htm#2
21	INFOCOM	2003	xpl/RecentCon.jsp?punumber=85858&conhome=10000359	http://www.ieee-infocom.org/2003/technical_programs.htm
22	INFOCOM	2004	xpl/RecentCon.jsp?punumber=93698&conhome=10000359	http://www.ieee-infocom.org/2004/technicalprogram.htm
23	INFOCOM	2006	xpl/tocresult.jsp?isnumber=4146653&isYear=2006	http://www.ieee-infocom.org/2006/technical_program.htm
24	INFOCOM	2007	xpl/tocresult.jsp?isnumber=4215592&isYear=2007	http://www.ieee-infocom.org/2007/technical.html
25	INFOCOM	2008	xpl/tocresult.jsp?isnumber=4509595&isYear=2008	http://www.ieee-infocom.org/2008/tech_prog.html
26	INFOCOM	2009	xpl/tocresult.jsp?isnumber=5061888&isYear=2009	http://www.ieee-infocom.org/2009/technicalProgram.htm
27	DEST	2008	xpl/tocresult.jsp?isnumber=4635078&isYear=2008	http://www.ieee-dest.curtin.edu.au/2008/program_new.php
28	DEST	2007	xpl/tocresult.jsp?isnumber=4233617&isYear=2007	http://www.ieee-dest.curtin.edu.au/2007/program.php
29	ICEBE	2005	xpl/tocresult.jsp?isnumber=330628&isYear=2005	http://www.cs.hku.hk/icebe2005/AdvanceProgram.htm
30	RE	2001	xpl/tocresult.jsp?isnumber=20518&isYear=2001	http://www.cs.toronto.edu/~sme/RE01/program.html
31	ICRE	2000	xpl/tocresult.jsp?isnumber=35810&isYear=2000	http://www.cse.msu.edu/ICRE2000/AdvProgram9-web.htm
32	ETFA	2003	xpl/tocresult.jsp?isnumber=27939&isYear=2003	http://www.uninova.pt/etfa2003/programme.htm
33	SEFM	2006	xpl/tocresult.jsp?isnumber=35815&isYear=2006	http://www.iist.unu.edu/SEFM06/programme.html
34	RE	1998	xpl/tocresult.jsp?isnumber=14673&isYear=1998	http://www.cs.technion.ac.il/~icre98/sessions.html
35	ICUMB	2008	xpl/tocresult.jsp?isnumber=4653270&isYear=2008	http://www.icuwb2008.org/p/General_Information/Program/September-10/
36	P2P	2002	xpl/tocresult.jsp?isnumber=22420&isYear=2002	http://www.ida.liu.se/conferences/p2p/p2p2002/program.htm
37	ESCIENCE	2005	xpl/tocresult.jsp?isnumber=33262&isYear=2005	http://www.gridbus.org/escience/escience2005schedule.html
38	IPSN	2008	xpl/tocresult.jsp?isnumber=4505449&isYear=2008	http://ipsn.acm.org/2008/#Program
39	IPSN	2007	xpl/tocresult.jsp?isnumber=4379652&isYear=2007	http://www.cse.wustl.edu/~lu/ipsn07.html#Program
40	IPSN	2006	xpl/tocresult.jsp?isnumber=347958&isYear=2006	http://ipsn.acm.org/2006/ipsn06Program.htm
41	CVPR	2005	xpl/RecentCon.jsp?punumber=9901&conhome=10000147	http://www.cs.duke.edu/cvpr2005/program.htm
42	CSEET	2008	xpl/tocresult.jsp?isnumber=4556933&isYear=2008	http://www.csc2.ncsu.edu/conferences/cseet/schedule.php
43	CSEET	2006	xpl/tocresult.jsp?isnumber=339018&isYear=2006	http://db-itm.shidler.hawaii.edu/cseet2006/program.php
44	CSEE&T	2004	xpl/tocresult.jsp?isnumber=28540&isYear=2004	http://www.cs.virginia.edu/~horton/cseet04//program-sched.html
45	CSEE&T	2002	xpl/tocresult.jsp?isnumber=21464&isYear=2002	http://www.site.uottawa.ca/cseet2002/program.html
46	Cluster	2008	xpl/tocresult.jsp?isnumber=4663724&isYear=2008	http://www.clustercomp.org/cluster2008/program.html
47	Cluster	2007	xpl/tocresult.jsp?isnumber=4629185&isYear=2007	http://www.cluster2007.org/program.php
48	Cluster	2006	xpl/tocresult.jsp?isnumber=4100338&isYear=2006	http://www.clustercomp.org/cluster2006/SessionProgram.htm
49	Cluster	2005	xpl/tocresult.jsp?isnumber=4139312&isYear=2005	http://cluster2005.org/program.html
50	Cluster	2004	xpl/tocresult.jsp?isnumber=30802&isYear=2004	http://www.clustercomp.org/cluster2004/program.html

Tab. E.1: Studie der HTML-Konferenzprogramme: URLs (I)

			URL bei IEEE Xplore (http://ieeexplore.ieee.org/...)	URL des Konferenzprogramms
51	Cluster	2003	xpl/tocresult.jsp?isnumber=28041&isYear=2003	http://www.csis.hku.hk/cluster2003/program.html
52	Cluster	2001	xpl/tocresult.jsp?isnumber=20737&isYear=2001	http://www.cacr.caltech.edu/cluster2001/program/
53	Cluster	2000	xpl/tocresult.jsp?isnumber=19246&isYear=2000	http://www.tu-chemnitz.de/informatik/RA/cluster2000/scheddetail1.htm
54	MFI	2003	xpl/tocresult.jsp?isnumber=27619&isYear=2003	http://www.cvl.iis.u-tokyo.ac.jp/mfi2003/Program.html
55	ICASSP	2008	xpl/tocresult.jsp?isnumber=4517521&isYear=2008	http://www.icassp2008.com/RegularProgram.asp
56	CSEET	2007	xpl/tocresult.jsp?isnumber=4271572&isYear=2007	http://www.computing.dcu.ie/cseet2007/program.php
57	SC	2005	xpl/tocresult.jsp?isnumber=33129&isYear=2005	http://sc05.supercomp.org/programs/technical_papers.php
58	GRC	2007	xpl/tocresult.jsp?isnumber=4403045&isYear=2007	http://xanadu.cs.sjsu.edu/~grc/grc2007/?index=program
59	ICGSE	2008	xpl/tocresult.jsp?isnumber=4638633&isYear=2008	http://icgse2008.di.uniba.it/techprogram.html
60	ICGSE	2007	xpl/tocresult.jsp?isnumber=4299826&isYear=2007	http://www.inf.pucrs.br/icgse2006/icgse2007/tech.htm
61	ICDM	2007	xpl/tocresult.jsp?isnumber=4470210&isYear=2007	http://www.ist.unomaha.edu/icdm2007/conference/program.php
62	ICDM	2005	xpl/tocresult.jsp?isnumber=33217&isYear=2005	http://www.cacs.louisiana.edu/~icdm05/finalprograms.html
63	ICDM	2004	xpl/tocresult.jsp?isnumber=30565&isYear=2004	http://icdm04.cs.uni-dortmund.de/Programm1b.html
64	ICDM	2002	xpl/tocresult.jsp?isnumber=28564&isYear=2002	http://www.vv-lab.com/icdm02/program.html
65	ICDM	2001	xpl/tocresult.jsp?isnumber=21329&isYear=2001	http://www.cs.uvm.edu/~icdm/program-01.shtml
66	P2P	2008	xpl/tocresult.jsp?isnumber=4627243&isYear=2008	http://www.p2p08.org/program
67	SASO	2007	xpl/tocresult.jsp?isnumber=4274872&isYear=2007	http://projects.csail.mit.edu/saso2007/program.html
68	SASO	2008	xpl/tocresult.jsp?isnumber=4663391&isYear=2008	http://polaris.ing.unimo.it/saso2008/paper_program.html
69	NANO	2002	xpl/tocresult.jsp?isnumber=22166&isYear=2002	http://ewh.ieee.org/tc/nanotech/nano2002/techprog.html
70	ICPC	2006	xpl/tocresult.jsp?isnumber=3420&isYear=2006	http://www.icpc2006.uwaterloo.ca/icpc06-program-detailed.htm
71	SMC	2006	xpl/RecentCon.jsp?punumber=4273787&conhome=1000738	http://irs.cn.nctu.edu.tw/smc2006/session_list.htm
72	NGMAST	2008	xpl/tocresult.jsp?isnumber=4756389&isYear=2008	http://www.comp.glam.ac.uk/NGMAST08/NGMAST2008_presentations/NGMAST_programme_Wlinks.htm
73	NGMAST	2007	xpl/tocresult.jsp?isnumber=4343385&isYear=2007	http://www.comp.glam.ac.uk/ngmast07/NGMAST2007-Detailed%20Programme.htm
74	RCIS	2009	xpl/tocresult.jsp?isnumber=5089258&isYear=2009	http://www.farcampus.com/rcis2009/program.php
75	RCIS	2008	xpl/tocresult.jsp?isnumber=4632084&isYear=2008	http://www.farcampus.com/rcis2008/program.php
76	Polytronic	2007	xpl/tocresult.jsp?isnumber=4389119&isYear=2007	http://www.su.t.u-tokyo.ac.jp/~polytronic/programme.html
77	AVSS	2007	xpl/tocresult.jsp?isnumber=4425266&isYear=2007	http://www.elec.qmul.ac.uk/staffinfo/andrea/avss2007_p.html
78	EDOC	2008	xpl/tocresult.jsp?isnumber=4634720&isYear=2008	http://www.lrz-muenchen.de/~edoc2008/researchPaperProgram.html
79	EDOC	2005	xpl/tocresult.jsp?isnumber=32900&isYear=2005	http://edoc2005.cttt.utwente.nl/sessions.html
80	EDOC	2004	xpl/tocresult.jsp?isnumber=29569&isYear=2004	http://www.cis.uab.edu/info/edoc2004/program.htm
81	SECON	2006	xpl/RecentCon.jsp?punumber=4068086&conhome=1001051	http://www.ieee-secon.org/2006/techprog.html
82	SECON	2005	xpl/tocresult.jsp?isnumber=33064&isYear=2005	http://www.ieee-secon.org/2005/program.html
83	SECON	2004	xpl/tocresult.jsp?isnumber=30129&isYear=2004	http://www.ieee-secon.org/2004/sessions.htm
84	SEFM	2004	xpl/tocresult.jsp?isnumber=29639&isYear=2004	http://www.iist.unu.edu/SEFM2004/programme.html
85	HST	2007	xpl/tocresult.jsp?isnumber=4227668&isYear=2007	http://www.ieeehomelandsecurity2007.org/homeland2007_program_agenda.htm
86	ICPS	2005	xpl/tocresult.jsp?isnumber=32279&isYear=2005	http://icps2005.cs.ucr.edu/program.html
87	EuroSimE	2009	xpl/tocresult.jsp?isnumber=4838404&isYear=2009	http://www.eurosime.org/b08.htm
88	ICDE	2008	xpl/tocresult.jsp?isnumber=4497384&isYear=2008	http://www.icde2008.org/index.php?option=com_content&task=view&id=57&Itemid=93
89	AINA	2003	xpl/tocresult.jsp?isnumber=26745&isYear=2003	http://www.aina-conference.org/2003/finalprogram.html
90	ETFA	2006	xpl/tocresult.jsp?isnumber=4178171&isYear=2006	http://www.action-m.com/etfa2006/program/index.html
91	ICET	2008	xpl/tocresult.jsp?isnumber=4777454&isYear=2008	http://www.ceme.edu.pk/IEEE/icet2008/New%20IEEE/TechnicalProgram.html
92	INDIN	2005	xpl/RecentCon.jsp?punumber=4384708&conhome=1001443	http://www.indin2007.org/article_list.php
93	ICFPT	2007	xpl/tocresult.jsp?isnumber=4439214&isYear=2007	http://www.kameyama.ece.tohoku.ac.jp/icfpt07/program.html
94	ICFPT	2005	xpl/tocresult.jsp?isnumber=33244&isYear=2005	http://www.comp.nus.edu.sg/~icfpt05/prog.htm
95	ICFPT	2004	xpl/tocresult.jsp?isnumber=30303&isYear=2004	http://icfpt04.itee.uq.edu.au/program.html
96	ICFPT	2002	xpl/tocresult.jsp?isnumber=26638&isYear=2002	http://www.icfpt.org/fpt2002/preliminary.html
97	AICCSA	2005	xpl/tocresult.jsp?isnumber=30191&isYear=2005	http://lyle.smu.edu/cse/AICCSA-05/sessions.htm
98	MASCOTS	2008	xpl/tocresult.jsp?isnumber=4770548&isYear=2008	http://www.mascots-conference.org/sessions-2008.html
99	ICEBE	2007	xpl/tocresult.jsp?isnumber=4402049&isYear=2007	http://www.cs.hku.hk/icebe2007/program/ICEBE07AdvanceProgram.html
100	BIBM	2007	xpl/tocresult.jsp?isnumber=4413018&isYear=2007	http://www.ischool.drexel.edu/ieeebibm/bibm07/IEEE%20BIBM%2007%20Program.htm

Tab. E.2: Studie der HTML-Konferenzprogramme: URLs (II)

			Anzahl HTML-Seiten	Seitenaufbau	Irrelevanter header	Irrelevanter footer	HTML-Version	HTML-Generator	valides HTML?	vollständige Namen	Zwischenüberschriften	Artikel in Reihenfolge	Anordnung Titel / Autoren	Anordnung Namensteile	Tags innerhalb Titel	Tags innerhalb Namen	T / A mittels Tags getrennt	Dynamischer Code zw. T / A	Schlüssel in Zw.-Überschrift
1	COMSWARE	2008	1	L	–	–	xhtml 1.1	k.A.	–	+	+	–	TA	VN	–	–	+	–	–
2	COMSWARE	2007	1	L	+	–	HTML 4.01 Transitional	k.A.	–	+	–	–	TA	VN	–	–	+	–	–
3	CCNC	2008	1	L	+	–	HTML 4.01 Transitional	MSHTML 6.00.2800.1565	–	+	+	+	TA	VN	–	–	+	–	–
4	RE	2006	1	L	+	–	k. A.	Claris Home Page 2.0	–	+	+	–	TA	VN	–	–	+	–	–
5	RE	2003	1	L	+	+	k. A.	k.A.	–	+	+	+	TA	VN	–	+	+	–	Session
6	EEE	2005	1	L	+	–	HTML 4.01 Transitional	k.A.	–	+	+	+	TA	VN	–	–	+	–	–
7	CEC	2004	1	L	+	–	xhtml 1.0 Transitional	Microsoft Word 10	–	+	+	+	TA	VN	+	+	+	+	–
8	CEC/EEE	2007	1	L	+	–	HTML 4.0 Transitional	k.A.	–	+	+	+	TA	VN	–	–	+	–	Session
9	CEC/EEE	2008	1	T	–	–	xhtml 1.0 Transitional	k.A.	–	+	+	–	AT	VN	–	–	+	–	Session
10	PerCom	2006	1	L	+	–	xhtml 1.0 Transitional	k.A.	–	+	+	+	TA	VN	–	–	+	–	Session
11	PerCom	2003	1	L	+	–	k. A.	Microsoft Frontpage 5.0	–	+	+	+	TA	VN	–	–	+	–	Session
12	PerCom	2004	1	L	+	–	k. A.	Microsoft Frontpage 5.0	–	+	+	+	TA	VN	+	+	+	+	Session
13	PerCom	2007	1	L	+	+	HTML 4.01 Transitional	k.A.	–	+	+	+	TA	VN	–	+	+	–	–
14	HST	2009	1	L	+	–	xhtml 1.0 Transitional	Microsoft Frontpage 5.0	–	+	+	+	TA	VN	–	–	+	–	Session
15	CCC	2006	1	L	+	–	HTML 4.0 Final	k.A.	–	+	–	–	TA	VN	–	–	+	–	–
16	CCC	2008	1	L	+	+	HTML 4.01 Transitional	k.A.	–	+	+	+	TA	VN	–	–	+	–	–
17	ICIP	2008	1	L	+	+	xhtml 1.0 Transitional	k.A.	–	+	+	–	TA	VN	–	–	+	–	Session
18	BROADNETS	2007	3	L	–	+	HTML 4.01 Transitional	Adobe GoLive	–	+	+	–	TA	VN	–	–	+	–	–
19	RFID	2009	1	L	+	+	HTML 4.01 Transitional	MSHTML 6.00.2800.1565	–	+	+	–	TA	VN	–	–	+	–	Session
20	ITC	2009	1	L	+	–	HTML 4.01 Transitional	k.A.	–	–	+	+	TA	VN	–	–	+	–	Session
21	INFOCOM	2003	1	L	+	–	HTML 4.01 Transitional	Microsoft Frontpage 4.0	–	+	+	+	TA	VN	–	–	+	–	Track
22	INFOCOM	2004	1	L	–	+	HTML 4.01 Transitional	Microsoft Frontpage 5.0	–	+	+	+	TA	VN	–	–	+	–	–
23	INFOCOM	2006	1	T	+	+	HTML 4.01 Transitional	Microsoft Word 10	–	+	+	–	TA	VN	+	+	+	+	Session
24	INFOCOM	2007	1	T	+	–	xhtml 1.0 Transitional	Microsoft Word 11	–	+	+	+	TA	VN	+	+	+	+	Session
25	INFOCOM	2008	1	L	+	–	xhtml 1.0 Transitional	k.A.	–	+	+	–	TA	VN	–	+	+	–	Session
26	INFOCOM	2009	1	L	+	–	xhtml 1.0 Transitional	Microsoft Excel 12	–	+	+	+	TA	VN	–	–	+	–	–
27	DEST	2008	1	T	+	+	HTML 4.01 Transitional	k.A.	–	+	+	–	TA	VN	–	–	+	–	Track
28	DEST	2007	1	T	+	+	HTML 4.01 Transitional	k.A.	–	+	+	–	TA	VN	–	–	+	–	–
29	ICEBE	2005	1	T	–	–	xhtml 1.0 Transitional	Microsoft Excel 11	–	–	+	+	TA	VN	–	–	+	–	Session
30	RE	2001	1	L	+	+	k. A.	k.A.	–	+	+	+	TA	VN	–	–	+	–	–
31	ICRE	2000	1	L	+	+	HTML 4.0 Transitional	Microsoft Frontpage 3.0	–	+	+	+	TA	VN	–	–	+	–	–
32	ETFA	2003	1	L	+	–	k. A.	Microsoft Frontpage 5.0	–	+	+	+	TA	VN	–	–	+	+	–
33	SEFM	2006	1	L	+	+	HTML 4.01 Strict	k.A.	–	+	+	–	TA	VN	–	–	+	–	Session
34	RE	1998	1	L	+	–	k. A.	Microsoft Frontpage Exp. 2.0	–	+	+	+	AT	VN	–	–	+	–	–

Tab. E.3: Studie der HTML-Konferenzprogramme: Struktur (I)

			Anzahl HTML-Seiten	Seitenaufbau	Irrelevanter header	Irrelevanter footer	HTML-Version	HTML-Generator	valides HTML?	vollständige Namen	Zwischenüberschriften	Artikel in Reihenfolge	Anordnung Titel / Autoren	Anordnung Namensteile	Tags innerhalb Titel	Tags innerhalb Namen	T / A mittels Tags getrennt	Dynamischer Code zw. T / A	Schlüssel in Zw.-Überschrift
35	ICUWB	2008	3	T	–	–	xhtml 1.0 Strict	k.A.	–	+	+	–	TA	VN	–	–	+	–	–
36	P2P	2002	1	L	–	–	k. A.	k.A.	–	+	–	–	TA	VN	–	–	+	–	–
37	ESCIENCE	2005	1	T	+	–	xhtml 1.0 Transitional	Microsoft Frontpage 5.0	–	+	+	–	TA	VN	–	–	+	–	–
38	IPSN	2008	1	L	+	+	xhtml 1.0 Transitional	Microsoft Word 12	–	+	+	+	TA	VN	–	+	+	–	Session
39	IPSN	2007	1	L	+	+	HTML 4.0 Transitional	Microsoft Frontpage 6.0	–	+	+	+	TA	VN	–	–	+	–	Session
40	IPSN	2006	1	L	+	–	HTML 4.01 Transitional	k.A.	–	+	+	+	TA	VN	–	–	+	–	–
41	CVPR	2005	1	L	+	+	HTML 4.01 Transitional	Microsoft Frontpage 5.0	–	–	+	+	TA	VN	–	–	+	–	Session
42	CSEET	2008	1	L	+	–	HTML 4.01 Transitional	k.A.	–	+	+	+	TA	VN	–	–	+	+	–
43	CSEET	2006	1	L	–	–	HTML 4.01 Transitional	k.A.	–	+	+	–	TA	VN	–	+	+	+	Session
44	CSEE&T	2004	1	L	–	–	HTML 3.2	Visual Page 2.0 for Windows	–	+	+	+	TA	VN	–	–	+	–	–
45	CSEE&T	2002	1	L	+	–	HTML 4.01 Transitional	k.A.	–	+	+	–	AT	VN	–	–	+	–	Session
46	Cluster	2008	1	T	+	+	HTML 4.01 Transitional	IBM WebSphere Studio	–	+	+	+	TA	VN	–	–	+	+	Session
47	Cluster	2007	1	L	+	+	HTML 4.01 Transitional	k.A.	–	+	+	+	TA	VN	–	–	+	–	–
48	Cluster	2006	1	L	+	+	xhtml 1.0 Transitional	k.A.	–	+	+	+	TA	VN	–	–	+	–	Session
49	Cluster	2005	1	T	–	–	xhtml 1.0 Transitional	Microsoft Word 10	–	+	+	+	TA	VN	–	+	+	–	Session
50	Cluster	2004	1	T	–	–	HTML 4.0 Transitional	OpenOffice.org 1.1.0 (Linux)	–	–	+	+	TA	VN	–	–	+	–	Session
51	Cluster	2003	1	T	+	–	HTML 4.0 Transitional	k.A.	–	+	+	+	TA	VN	–	–	+	–	Session
52	Cluster	2001	1	T	+	–	HTML 4.0 Transitional	Microsoft Frontpage 4.0	–	+	+	–	AT	VN	–	+	+	+	–
53	Cluster	2000	1	T	–	–	HTML 4.01 Transitional	k.A.	–	+	+	+	TA	VN	–	–	+	–	Session
54	MFI	2003	1	L	+	–	HTML 4.01 Transitional	IBM WebSphere Studio	–	+	+	+	TA	VN	–	–	+	–	–
55	ICASSP	2008	> 200	L	+	–	HTML 4.01 Strict	k.A.	–	+	+	–	TA	VN	–	–	+	–	–
56	CSEET	2007	3	L	+	–	k. A.	k.A.	–	+	+	+	TA	VN	–	–	+	–	–
57	SC	2005	1	L	–	–	HTML 4.01 Transitional	k.A.	–	+	+	–	TA	VN	–	–	+	–	(Spalte)
58	GRC	2007	1	L	+	–	xhtml 1.0 Transitional	k.A.	–	+	+	–	var.	VN	–	–	–	–	–
59	ICGSE	2008	3	L	–	–	xhtml 1.0 Transitional	k.A.	–	+	+	–	TA	VN	–	–	+	–	Session
60	ICGSE	2007	3	L	–	–	HTML 4.01 Transitional	k.A.	–	+	+	–	TA	VN	–	–	+	+	–
61	ICDM	2007	1	L	+	+	HTML 4.01 Transitional	k.A.	–	+	+	–	AT	VN	–	–	+	–	–
62	ICDM	2005	1	L	+	–	HTML 4.0 Transitional	Microsoft Word 9	–	+	+	–	TA	VN	–	–	+	–	Session
63	ICDM	2004	1	L	+	–	HTML 4.01 Transitional	k.A.	–	+	+	–	AT	VN	–	–	+	–	–
64	ICDM	2002	1	L	+	–	HTML 4.01 Transitional	k.A.	–	+	+	–	TA	VN	–	–	+	–	Session
65	ICDM	2001	1	T	+	+	HTML 4.01 Transitional	k.A.	–	+	+	–	AT	VN	–	–	+	–	–
66	P2P	2008	1	L	+	+	xhtml 1.0 Transitional	Plone – http://plone.org	–	+	+	+	TA	VN	–	–	+	–	Session
67	SASO	2007	1	L	+	–	HTML 4.01 Transitional	k.A.	–	+	+	+	AT	VN	–	–	+	+	Session
68	SASO	2008	1	L	+	–	HTML 4.01 Transitional	MSHTML 6.00.5730.11	+	+	+	–	TA	VN	–	–	+	–	Session

Tab. E.4: Studie der HTML-Konferenzprogramme: Struktur (II)

			Anzahl HTML-Seiten	Seitenaufbau	Irrelevanter header	Irrelevanter footer	HTML-Version	HTML-Generator	valides HTML?	vollständige Namen	Zwischenüberschriften	Artikel in Reihenfolge	Anordnung Titel / Autoren	Anordnung Namensteile	Tags innerhalb Titel	Tags innerhalb Namen	T / A mittels Tags getrennt	Dynamischer Code zw. T / A	Schlüssel in Zw.-Überschrift
69	NANO	2002	1	L	–	+	xhtml 1.0 Transitional	Microsoft Word 9	–	+	+	–	TA	VN	–	+	+	+	–
70	ICPC	2006	1	L	–	+	HTML 4.01 Transitional	Microsoft Word 10	–	+	+	+	TA	VN	–	+	+	+	Session
71	SMC	2006	1	L	–	+	HTML 4.0 Transitional	Microsoft Excel 10	–	+	+	+	TA	VN	–	+	+	–	–
72	NGMAST	2008	1	L	+	+	xhtml 1.0 Transitional	Microsoft Word 12	–	+	+	–	TA	VN	+	+	+	+	Session
73	NGMAST	2007	1	L	+	–	xhtml 1.0 Transitional	Microsoft Word 11	–	+	+	–	TA	VN	+	+	+	+	Session
74	RCIS	2009	1	L	+	+	HTML 4.01 Transitional	Microsoft Word 12 (filtered)	–	+	+	+	TA	VN	+	+	+	+	Session
75	RCIS	2008	1	L	+	+	HTML 4.01 Transitional	k.A.	–	+	+	+	TA	VN	–	–	+	–	Session
76	Polytronic	2007	1	L	+	+	HTML 4.0 Transitional	k.A.	–	+	–	+	AT	VN	–	–	+	–	–
77	AVSS	2007	1	L	+	+	HTML 4.0 Transitional	Microsoft Frontpage 5.0	–	+	+	+	TA	VN	–	+	+	+	–
78	EDOC	2008	1	L	+	–	HTML 4.01 Transitional	Eclipse 3.3	–	+	+	–	TA	VN	–	–	+	–	–
79	EDOC	2005	1	L	+	–	HTML 4.01 Transitional	Microsoft Frontpage 6.0	–	+	+	+	TA	VN	–	–	+	–	Session
80	EDOC	2004	1	L	+	–	HTML 4.01 Transitional	k.A.	–	+	+	+	TA	VN	–	–	+	–	Session
81	SECON	2006	1	L	+	+	HTML 4.0 Transitional	k.A.	–	+	+	+	TA	VN	–	–	+	–	Session
82	SECON	2005	1	L	+	–	HTML 4.0 Transitional	k.A.	–	+	+	+	TA	VN	–	–	+	–	–
83	SECON	2004	1	L	+	–	HTML 4.01 Transitional	k.A.	–	+	+	+	TA	VN	–	+	+	–	Session
84	SEFM	2004	1	L	+	+	HTML 4.01 Strict	k.A.	–	+	+	+	TA	VN	–	–	+	–	Section
85	HST	2007	1	L	+	+	HTML 4.01 Transitional	Microsoft Frontpage 5.0	–	+	+	+	TA	NV	–	–	+	–	Session
86	ICPS	2005	1	L	+	–	HTML 4.01 Transitional	k.A.	–	+	+	+	TA	VN	–	+	+	–	Session
87	EuroSimE	2009	1	L	+	–	xhtml 1.0 Transitional	k.A.	–	+	+	–	TA	VN	–	+	+	–	Session
88	ICDE	2008	1	L	+	–	HTML 4.01 Transitional	k.A.	–	+	+	+	TA	VN	–	–	+	–	Session
89	AINA	2003	1	L	+	–	HTML 4.01 Transitional	k.A.	–	–	+	+	TA	VN	–	–	–	+	Session
90	ETFA	2006	1	L	–	–	HTML 4.0 Transitional	k.A.	–	+	–	–	TA	VN	+	+	+	–	–
91	ICET	2008	1	L	–	–	HTML 4.01 Transitional	k.A.	–	+	+	–	TA	VN	–	–	+	–	Session
92	INDIN	2005	1	L	+	–	HTML 4.01 Transitional	k.A.	–	+	–	–	TA	VN	–	–	+	–	–
93	ICFPT	2007	1	L	+	–	HTML 4.01 Transitional	Dicre Sitehina	–	+	+	+	TA	VN	+	+	+	–	Session
94	ICFPT	2005	1	L	+	–	xhtml 1.0 Transitional	Microsoft Frontpage 5.0	–	+	+	+	TA	VN	–	–	+	–	Session
95	ICFPT	2004	1	T	+	–	xhtml 1.0 Transitional	k.A.	–	+	+	–	TA	VN	–	–	+	–	Session
96	ICFPT	2002	1	L	–	–	xhtml 1.0 Transitional	Microsoft Word 9	–	+	+	+	TA	VN	–	–	+	–	Session
97	AICCSA	2005	1	L	–	–	xhtml 1.0 Transitional	Microsoft Frontpage 5.0	–	+	+	–	TA	VN	–	+	+	–	–
98	MASCOTS	2008	1	L	+	–	HTML 4.01 Transitional	k.A.	–	+	+	–	TA	VN	–	–	+	–	–
99	ICEBE	2007	1	T	–	–	xhtml 1.0 Transitional	Microsoft Excel 11	–	+	+	–	TA	VN	+	+	+	+	Session
100	BIBM	2007	1	L	–	–	xhtml 1.0 Transitional	Microsoft Word 12	–	+	+	+	var.	VN	+	+	–	–	Session

Tab. E.5: Studie der HTML-Konferenzprogramme: Struktur (III)

			Xplore		Strategie 1		Strategie 2		Strategie 3		Strategie 1		Strategie 2		Strategie 3	
			vorhandene Records	vorhandene Autorennamen	Records	Autorennamen	Records	Autorennamen	Records	Autorennamen	Records %	Namen %	Records %	Namen %	Records %	Namen %
1	COMSWARE	2008	114	352	82	203	71	187	76	232	71,93	57,67	62,28	53,13	66,67	65,91
2	COMSWARE	2007	140	437	98	247	85	225	73	232	70,00	56,52	60,71	51,49	52,14	53,09
3	CCNC	2008	297	984	140	290	115	246	0	0	47,14	29,47	38,72	25,00	0,00	0,00
4	RE	2006	65	176	54	125	55	128	29	82	83,08	71,02	84,62	72,73	44,62	46,59
5	RE	2003	62	146	61	128	58	123	0	0	98,39	87,67	93,55	84,25	0,00	0,00
6	EEE	2005	141	426	100	240	97	227	115	310	70,92	56,34	68,79	53,29	81,56	72,77
7	CEC	2004	54	153	21	35	1	3	0	0	38,89	22,88	1,85	1,96	0,00	0,00
8	CEC/EEE	2007	98	296	53	123	51	119	34	85	54,08	41,55	52,04	40,20	34,69	28,72
9	CEC/EEE	2008	66	190	47	97	45	95	0	0	71,21	51,05	68,18	50,00	0,00	0,00
10	PerCom	2006	38	121	35	95	35	94	19	54	92,11	78,51	92,11	77,69	50,00	44,63
11	PerCom	2003	65	213	57	150	36	89	0	0	87,69	70,42	55,38	41,78	0,00	0,00
12	PerCom	2004	40	114	22	32	12	18	0	0	55,00	28,07	30,00	15,79	0,00	0,00
13	PerCom	2007	28	103	6	7	4	5	0	0	21,43	6,80	14,29	4,85	0,00	0,00
14	HST	2009	100	535	0	0	0	0	0	0	0,00	0,00	0,00	0,00	0,00	0,00
15	CCC	2006	30	56	27	47	25	45	22	45	90,00	83,93	83,33	80,36	73,33	80,36
16	CCC	2008	32	74	0	0	0	0	0	0	0,00	0,00	0,00	0,00	0,00	0,00
17	ICIP	2008	808	2412	1	5	0	0	0	0	0,12	0,21	0,00	0,00	0,00	0,00
18	BROADNETS	2007	541	1581	0	0	0	0	0	0	0,00	0,00	0,00	0,00	0,00	0,00
19	RFID	2009	47	160	40	95	41	100	0	0	85,11	59,38	87,23	62,50	0,00	0,00
20	IITC	2009	80	679	16	91	3	32	0	0	20,00	13,40	3,75	4,71	0,00	0,00
21	INFOCOM	2003	224	645	212	520	103	207	197	555	94,64	80,62	45,98	32,09	87,95	86,05
22	INFOCOM	2004	268	784	224	503	193	430	0	0	83,58	64,16	72,01	54,85	0,00	0,00
23	INFOCOM	2006	344	1064	279	724	10	21	0	0	81,10	68,05	2,91	1,97	0,00	0,00
24	INFOCOM	2007	320	1015	262	606	22	50	0	0	81,88	59,70	6,88	4,93	0,00	0,00
25	INFOCOM	2008	311	1002	248	558	245	550	0	0	79,74	55,69	78,78	54,89	0,00	0,00
26	INFOCOM	2009	378	1236	299	732	295	700	116	312	79,10	59,22	78,04	56,63	30,69	25,24
27	DEST	2008	122	326	87	214	77	180	0	0	71,31	65,64	63,11	55,21	0,00	0,00
28	DEST	2007	113	329	83	226	77	209	0	0	73,45	68,69	68,14	63,53	0,00	0,00
29	ICEBE	2005	120	369	10	13	2	2	0	0	8,33	3,52	1,67	0,54	0,00	0,00
30	RE	2001	152	428	0	0	0	0	0	0	0,00	0,00	0,00	0,00	0,00	0,00
31	ICRE	2000	131	369	1	3	0	0	0	0	0,76	0,81	0,00	0,00	0,00	0,00
32	ETFA	2003	197	557	6	10	1	1	0	0	3,05	1,80	0,51	0,18	0,00	0,00
33	SEFM	2006	173	502	0	0	0	0	0	0	0,00	0,00	0,00	0,00	0,00	0,00
34	RE	1998	159	461	2	2	0	0	0	0	1,26	0,43	0,00	0,00	0,00	0,00
35	ICUWB	2008	141	375	1	1	0	0	0	0	0,71	0,27	0,00	0,00	0,00	0,00
36	P2P	2002	146	373	1	1	0	0	0	0	0,68	0,27	0,00	0,00	0,00	0,00
37	ESCIENCE	2005	75	291	54	165	47	152	0	0	72,00	56,70	62,67	52,23	0,00	0,00
38	IPSN	2008	63	218	54	143	2	2	0	0	85,71	65,60	3,17	0,92	0,00	0,00
39	IPSN	2007	76	275	62	168	54	155	12	46	81,58	61,09	71,05	56,36	15,79	16,73
40	IPSN	2006	63	252	63	226	61	214	31	136	100,00	89,68	96,83	84,92	49,21	53,97
41	CVPR	2005	353	1011	113	158	9	13	0	0	32,01	15,63	2,55	1,29	0,00	0,00
42	CSEET	2008	36	82	35	71	32	65	0	0	97,22	86,59	88,89	79,27	0,00	0,00
43	CSEET	2006	38	89	36	75	36	75	0	0	94,74	84,27	94,74	84,27	0,00	0,00
44	CSEE&T	2004	34	69	32	55	33	55	0	0	94,12	79,71	97,06	79,71	0,00	0,00
45	CSEE&T	2002	35	83	32	69	29	61	0	0	91,43	83,13	82,86	73,49	0,00	0,00
46	Cluster	2008	63	193	41	110	39	108	0	0	65,08	56,99	61,90	55,96	0,00	0,00
47	Cluster	2007	82	280	62	184	58	176	38	152	75,61	65,71	70,73	62,86	46,34	54,29
48	Cluster	2006	83	279	71	195	61	167	0	0	85,54	69,89	73,49	59,86	0,00	0,00
49	Cluster	2005	85	277	28	45	20	34	0	0	32,94	16,25	23,53	12,27	0,00	0,00
50	Cluster	2004	68	299	15	26	13	23	25	53	22,06	8,70	19,12	7,69	36,76	17,73

Tab. E.6: Studie der HTML-Konferenzprogramme: Auswertung (I)

			Xplore		Strategie 1		Strategie 2		Strategie 3		Strategie 1		Strategie 2		Strategie 3	
			vorhandene Records	vorhandene Autorennamen	Records	Autorennamen	Records	Autorennamen	Records	Autorennamen	Records %	Namen %	Records %	Namen %	Records %	Namen %
51	Cluster	2003	67	216	45	101	26	63	38	111	67,16	46,76	38,81	29,17	56,72	51,39
52	Cluster	2001	67	149	53	93	52	90	0	0	79,10	62,42	77,61	60,40	0,00	0,00
53	Cluster	2000	34	102	31	73	31	71	0	0	91,18	71,57	91,18	69,61	0,00	0,00
54	MFI	2003	59	184	56	169	39	106	24	76	94,92	91,85	66,10	57,61	40,68	41,30
55	ICASSP	2008	1352	3976	7	11	0	0	0	0	0,52	0,28	0,00	0,00	0,00	0,00
56	CSEET	2007	48	109	20	34	17	31	0	0	41,67	31,19	35,42	28,44	0,00	0,00
57	SC	2005	74	324	61	231	13	55	62	310	82,43	71,30	17,57	16,98	83,78	95,68
58	GRC	2007	155	430	67	146	59	129	110	328	43,23	33,95	38,06	30,00	70,97	76,28
59	ICGSE	2008	36	92	15	28	11	24	0	0	41,67	30,43	30,56	26,09	0,00	0,00
60	ICGSE	2007	41	102	13	29	10	26	0	0	31,71	28,43	24,39	25,49	0,00	0,00
61	ICDM	2007	102	316	75	166	75	165	0	0	73,53	52,53	73,53	52,22	0,00	0,00
62	ICDM	2005	141	430	119	285	117	280	2	6	84,40	66,28	82,98	65,12	1,42	1,40
63	ICDM	2004	105	304	83	171	81	167	46	119	79,05	56,25	77,14	54,93	43,81	39,14
64	ICDM	2002	121	313	90	176	88	170	0	0	74,38	56,23	72,73	54,31	0,00	0,00
65	ICDM	2001	109	287	75	158	57	116	41	178	68,81	55,05	52,29	40,42	37,61	62,02
66	P2P	2008	45	154	40	121	39	119	0	0	88,89	78,57	86,67	77,27	0,00	0,00
67	SASO	2007	53	153	29	71	24	59	0	0	54,72	46,41	45,28	38,56	0,00	0,00
68	SASO	2008	63	195	61	155	59	148	0	0	96,83	79,49	93,65	75,90	0,00	0,00
69	NANO	2002	118	392	35	52	19	35	0	0	29,66	13,27	16,10	8,93	0,00	0,00
70	ICPC	2006	43	117	24	44	8	16	0	0	55,81	37,61	18,60	13,68	0,00	0,00
71	SMC	2006	916	2791	451	1004	259	262	479	1255	49,24	35,97	28,28	9,39	52,29	44,97
72	NGMAST	2008	95	322	68	186	23	41	0	0	71,58	57,76	24,21	12,73	0,00	0,00
73	NGMAST	2007	53	166	37	57	1	2	0	0	69,81	34,34	1,89	1,20	0,00	0,00
74	RCIS	2009	50	150	40	105	8	15	0	0	80,00	70,00	16,00	10,00	0,00	0,00
75	RCIS	2008	53	152	44	96	44	95	0	0	83,02	63,16	83,02	62,50	0,00	0,00
76	Polytronic	2007	63	234	58	166	53	155	0	0	92,06	70,94	84,13	66,24	0,00	0,00
77	AVSS	2007	105	350	98	270	89	227	0	0	93,33	77,14	84,76	64,86	0,00	0,00
78	EDOC	2008	60	164	42	128	41	124	0	0	70,00	78,05	68,33	75,61	0,00	0,00
79	EDOC	2005	25	79	24	66	24	66	0	0	96,00	83,54	96,00	83,54	0,00	0,00
80	EDOC	2004	24	81	23	70	23	70	0	0	95,83	86,42	95,83	86,42	0,00	0,00
81	SECON	2006	124	341	62	117	10	14	60	181	50,00	34,31	8,06	4,11	48,39	53,08
82	SECON	2005	55	167	49	108	49	108	24	55	89,09	64,67	89,09	64,67	43,64	32,93
83	SECON	2004	69	188	65	147	56	104	0	0	94,20	78,19	81,16	55,32	0,00	0,00
84	SEFM	2004	43	108	36	74	33	70	18	42	83,72	68,52	76,74	64,81	41,86	38,89
85	HST	2007	51	161	37	119	18	56	24	129	72,55	73,91	35,29	34,78	47,06	80,12
86	ICPS	2005	59	199	59	169	45	125	0	0	100,00	84,92	76,27	62,81	0,00	0,00
87	EuroSimE	2009	114	514	97	275	78	236	30	145	85,09	53,50	68,42	45,91	26,32	28,21
88	ICDE	2008	238	856	195	571	192	560	0	0	81,93	66,71	80,67	65,42	0,00	0,00
89	AINA	2003	155	477	0	0	0	0	0	0	0,00	0,00	0,00	0,00	0,00	0,00
90	ETFA	2006	199	640	181	511	140	366	0	0	90,95	79,84	70,35	57,19	0,00	0,00
91	ICET	2008	62	197	0	0	0	0	0	0	0,00	0,00	0,00	0,00	0,00	0,00
92	INDIN	2005	209	645	168	506	91	226	0	0	80,38	78,45	43,54	35,04	0,00	0,00
93	ICFPT	2007	69	245	55	177	55	175	29	166	79,71	72,24	79,71	71,43	42,03	67,76
94	ICFPT	2005	68	209	22	28	12	18	0	0	32,35	13,40	17,65	8,61	0,00	0,00
95	ICFPT	2004	81	269	0	0	26	49	34	85	0,00	0,00	32,10	18,22	41,98	31,60
96	ICFPT	2002	79	246	13	18	9	14	0	0	16,46	7,32	11,39	5,69	0,00	0,00
97	AICCSA	2005	158	417	117	250	12	19	0	0	74,05	59,95	7,59	4,56	0,00	0,00
98	MASCOTS	2008	45	151	33	90	32	88	22	60	73,33	59,60	71,11	58,28	48,89	39,74
99	ICEBE	2007	110	349	52	91	16	22	0	0	47,27	26,07	14,55	6,30	0,00	0,00
100	BIBM	2007	62	198	17	19	16	18	0	0	27,42	9,60	25,81	9,09	0,00	0,00
Summe			13626	42152	6415	15545	4533	10581	1830	5540	60,06	47,82	46,74	37,26	13,97	14,31

Tab. E.7: Studie der HTML-Konferenzprogramme: Auswertung (II)

Erklärung zur Diplomarbeit

Hiermit erkläre ich, dass ich die Diplomarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und aus fremden Quellen direkt oder indirekt übernommenen Gedanken als solche kenntlich gemacht habe. Die Diplomarbeit habe ich bisher keinem anderen Prüfungsamt in gleicher oder vergleichbarer Form vorgelegt. Sie wurde bisher auch nicht veröffentlicht.

_____

Datum

_____

Unterschrift