Universität Leipzig
Fakultät für Mathematik und Informatik
Institut für Informatik

Aufbau einer Forschungsinstitut-Datenbank

Daten- und Informationsextraktion zu wissenschaftlichen Publikationen

Masterarbeit

Autor:	Enrico Tappert
Matrikelnummer:	9921458
Studiengang:	Informatik (M.Sc.)
	5.Semester
Betreuender Hochschullehrer:	Prof. Dr. Erhard Rahm
	Abteilung Datenbanken
Eingereicht am:	31.01.2008

Kurzzusammenfassung

Im Rahmen der Forschung werden stetig neue Ideen entwickelt und in Form von wissenschaftlichen Arbeiten publiziert. Diese referenzieren in der Regel weitere Publikationen. Das Forschungsgebiet der Zitierungsanalyse baut auf diesem Thema auf, um beispielsweise Autoren zu ermitteln, deren Arbeiten häufig zitiert werden.

Die Ermittlung von Ideenhochburgen stellt sich als ein nächstes Ziel dar. Erste manuelle Überprüfungen wurden durchgeführt, um die Einrichtungen zu identifizieren, an denen die genannten Autoren arbeiten. Um dem hohen zeitlichen Aufwand zu umgehen, wird in dieser Masterarbeit eine automatische Methode zur Identifikation von Forschungseinrichtungen entwickelt.

Zunächst werden einige verschiedene Datenquellen, speziell Web-Bibliografiedatenbanken, auf deren Datenbereitstellung und -qualität untersucht. Die unterschiedlichen Datenformate der Quellen (HTML, XML, PDF) setzen verschiedene Verfahren zur Datenextraktion voraus. Es wird eine Herangehensweise beschrieben, die mittels Wrappern mit ungleichen Quellen umgehen kann.

Auf der Basis der heterogenen Daten aus den Quellen werden verschiedene Ansätze zur Informationsextraktion näher betrachtet. Es soll überprüft und evaluiert werden, auf welche Art und Weise sich Verzeichnisse, Textsuchmaschinen und Dienste des Web (z.Bsp. Google-Maps-Dienst) für die Extraktion von Adress- und Einrichtungstyp-Informationen aus vorgegebenen Zeichenketten nutzen lassen.

Im Anschluss erfolgt eine Schilderung der Implementierung eines Programms zur Daten- und Informationsextraktion, mit dem eine umfassende Datenbank aufgebaut werden kann. Das Programm ist in der Lage, anhand weniger URLs Daten von vielen Internetseiten zu sammeln und daraus Informationen zu extrahieren.

Dass die Ergebnisse des Programms nicht nur ihren Zweck erfüllen, sondern über die Zielstellung hinaus gehen, zeigt eine anschließende Evaluation. Diese betrachtet die erreichte Datenqualität und-quantität, auch im Vergleich zu ersten manuellen Anstrengungen.

Inhaltsverzeichnis

1 Einleitung

Dieses einleitende Kapitel schildert den Ausgangspunkt der vorliegenden Arbeit. Hierbei wird zunächst anhand eines praktischen Beispiels aus der aktuellen Wissenschaft die Motivation erläutert. Anschließend werden die sich daraus ableitende Aufgabenstellung mit den zu erreichenden Zielen dargelegt. Ein weiteres Unterkapitel gibt einen Überblick über die Gliederung des nachfolgenden Hauptteils.

1.1 Motivation

Im wissenschaftlichen Umfeld werden kontinuierlich neue Veröffentlichungen hervorgebracht. Diese dienen der Verbreitung des neu geschaffenen Wissens. Wissenschaftler bauen darauf ihre Thesen auf, indem sie relevante Arbeiten hinterfragen und neue Ideen entwickeln. Dabei muss auf die aufgebauten, zitierten Werke in einer Literaturliste referenziert werden, damit die Kausalkette für andere (Wissenschaftler) nachvollziehbar ist.

Bedeutende Arbeiten werden häufig referenziert, wodurch die Bekanntheit der jeweiligen Autoren steigt. Diese Popularität ist wiederum Gegenstand aktueller Forschungen. Über die Anzahl der Referenzen, die hier mit Zitierungen gleichzusetzen sind, lässt sich eine gewisse Aussagekraft über die Bekanntheit von Wissenschaftlern und der Einfluss derer Publikationen feststellen. Solch eine Zitierungsanalyse wird beispielsweise in [Woh07] für Software-Engineering-Zeitschriften durchgeführt, in der der Leitgedanke ist:

> *„An understanding of which research is viewed by the research community as most valuable to build upon may provide valuable insights into what research to focus on now and in the future.“* [Woh07]

Die Wertigkeit einer wissenschaftlichen Arbeit stellt demnach eine gewisse Grundlage für die zukünftige Forschung dar.

Zitierungsanalysen konzentrieren sich meist auf ein Fachgebiet. Gründe hierfür sind die unzumutbare Anzahl an zu untersuchenden Veröffentlichungen, wenn keine Ein-

schränkungen vorgenommen werden würden, als auch das Interesse der Wissenschaftler an ihrem Fachgebiet. Diese überschneiden sich zwar zu Teilen, stellen in der Regel nur kurze aber wichtige Exkurse in den Publikationen dar.

In einem weiteren Schritt der Analyse von Publikationen stellt sich die Frage nach der institutionellen Zugehörigkeit von (bekannten) Wissenschaftlern. Somit ließen sich je nach Fachgebiet aufgrund ihrer Veröffentlichungen angesehene Forschungseinrichtungen ermitteln, sowie weitergehend in einem wissenschaftlichen Fachgebiet führende Länder bestimmen. Die Forschungseinrichtungen und deren Ortszugehörigkeit identifizieren zu können stellt das Hauptanliegen der vorliegenden Arbeit dar.

In der Abteilung Datenbanken der Universität Leipzig, im Weiteren als Abteilung Datenbanken bezeichnet, wurde eine Zitierungsanalyse durchgeführt, mit anschließender Ermittlung der Einrichtungen:

> *„[...] we study the distribution of citations over originating institutions and their countries. For simplicity, we only consider the first author's institution which we extracted manually from the [scientific] papers.“* [RT05]

Wie dem Zitat zu entnehmen ist, wurden aus einer Reihe von wissenschaftlichen Publikationen manuell die zu den Autoren zugehörigen Forschungseinrichtungen selektiert und das jeweilig zugeordnete Land ermittelt. Hier kann geschlussfolgert werden, dass sich eine manuelle Informationsextraktion als sehr zeitintensiv darstellt, weshalb in [RT05] in mehreren Hinsichten Einschränkungen vorgenommen wurden. Einerseits fanden nur Namen und dazugehörige Informationen zu den jeweils erst genannten Autoren einer Publikation Verwendung. Weiterhin bestand die Analyse aus den Veröffentlichungen von „nur“ zwei Datenbankkonferenzen und drei Datenbankzeitschriften. Auf der anderen Seite wurde der Aufwand reduziert, indem nur Publikationen mit mindestens 20 Zitierungen und davon die obere Hälfte der meist referenzierten als bedeutend erachtet wurden.
Es ist denkbar, dass dieser Aufwand nicht gerechtfertigt ist und somit eine manuelle Informationsextraktion keine dauerhafte Lösung darstellen kann.

1.2 Aufgabenstellung

Die in der Motivation aufgeführten Beweggründe für eine Erfassung von Autorzugehörigkeiten und Erkenntnisse erster Anstrengungen einer manuellen Vorgehensweise führen folglich zur in diesem Kapitel dargelegten Aufgabenstellung.

Früher wurden mehrere wissenschaftliche Publikationen zu einem Sachgebiet oder einer Konferenz in einem Buch beziehungsweise einer Zeitung zusammengefasst, um sie allen Interessierten zugänglich zu machen. Heutzutage werden wissenschaftliche Publikationen nicht mehr nur im Printformat sondern auch separat digital im Web veröffentlicht. Hierzu existiert eine Vielzahl an so genannten Bibliografiedatenbanken für eine zentrale Speicherung als auch Zugriff, die im Kapitel 2 umfassend beschrieben werden. Weiterhin besteht die Möglichkeit, über Suchmaschinen an benötigte Daten wie Titel, Autoren und Orte zu gelangen, zumeist aber nicht an den Volltext. Strukturierte Informationen bezüglich der Forschungsinstitute, von denen die jeweiligen Publikationen stammen, sind jedoch kaum in den verschiedenen Quellen verfügbar.

Ziel dieser Arbeit ist es, eine umfassende Forschungsinstitutsdatenbank aufzubauen. Der Schwerpunkt der Extraktion einer Adresse liegt hierbei auf der Zuordnung von Forschungsinstituten zum jeweiligen Land und damit auch Kontinent. Detailliertere Adressangaben sollen allerdings, insofern zugänglich, ebenso gespeichert werden. Weiterhin soll möglichst eine Zuordnung des jeweiligen Instituts zu dessen Typ (Universität, Industrie, etc.) erfolgen.

Die Adressen und Informationen zum Einrichtungstyp sind innerhalb einer Datenbank performant zu speichern. Um sie für weitergehende Aufgaben nutzen zu können, muss die Zugehörigkeit dieser Informationen zu Autoren und Publikationen ebenfalls bereitgehalten werden. Die Daten können somit beispielsweise nach einer Integration mit Data-Warehouse-Daten für *Online Analytical Processing (OLAP)*-Auswertungen genutzt werden.

Für die Aufgaben der Daten- und Informationsextraktion sind verschiedene Strategien zur Gewinnung möglichst umfassender Angaben zu untersuchen. Dabei kann eine Beschränkung auf Tagungen oder Papiere gegebener Fachgebiete erfolgen. Insbesondere ist es sinnvoll, auf den Bereich Datenbanken Rücksicht zu nehmen, da die

Aufgabenstellung von der Abteilung Datenbanken stammt.
In diesem Rahmen ist weiterhin zu beachten, inwiefern bei der Datengewinnung Geschäftsbedingungen verletzt werden könnten.

1.3 Interpretation der Aufgabenstellung

Um die Aufgabe zu erfüllen, eine Forschungsinstitutsdatenbank aufzubauen, und aufgrund der Motivation, einer nicht-manuellen Erfassung der Daten, lässt sich schlussfolgernd, dass ein Programm zu entwerfen und umzusetzen ist. Dieses muss in der Lage sein, die gewünschten Daten und Informationen für ausgewählte Publikationen automatisch zu extrahieren und anschließend zentral zu speichern. Damit kann zusätzlich gewährleistet werden, dass Analysen bezüglich wissenschaftlicher Institute ebenso für weitere Fachgebiete durchführbar sind, auch zu einem späteren Zeitpunkt. Hierzu ist es notwendig, die Umsetzung möglichst allgemein zu betrachten, so dass im Endeffekt Datenquellen unproblematisch austauschbar sind sowie die Daten- und Informationsextraktion dadurch nicht negativ beeinflusst werden.

1.4 Gliederung der Arbeit

Der Aufbau dieser Arbeit ergibt sich aus der Aufgabenstellung und deren Interpretation. So wird ein Programm entworfen, dessen Implementierungsdetails in Kapitel 5 beschrieben werden. Zur Verdeutlichung der Arbeitsgliederung wird in Abbildung 1 der grobe Programmaufbau vorweggenommen.

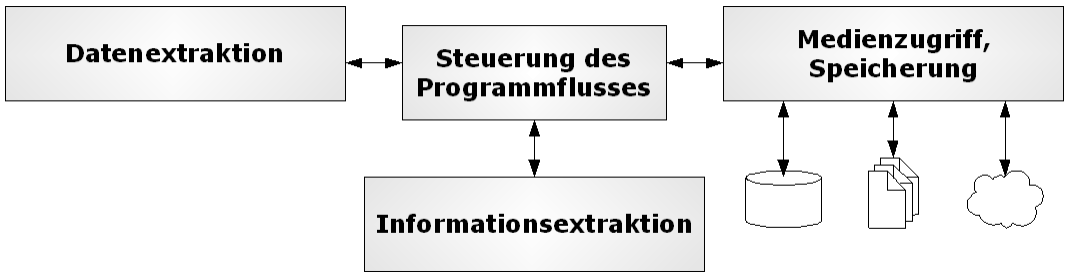


Abbildung 1: Grober, logischer Programmaufbau

Zu erkennen sind die logischen Programmteile, die die gleichnamigen Aufgaben Datenextraktion, Informationsextraktion und die Speicherung der Informationen über-

nehmen. In dieser Reihenfolge ist demzufolge diese Arbeit aufgebaut, um dem Arbeitsablauf des Programms zu folgen.

Das Kapitel 2 erläutert beginnend die verfügbaren (digitalen) Datenquellen. In diesem Zusammenhang werden die verschiedenen Arten der Datenbereitstellung erörtert und beispielsweise auf mögliche Verstöße gegen Geschäftsbedingungen eingegangen. Im Kapitel 3 werden die Möglichkeiten zur Datenextraktion aus den Quellen betrachtet. Weiterhin werden praktische Ergebnisse dieser Extraktion aufgeführt, da sie die Grundlage für den Aufbau der Informationsextraktion bilden. Hierfür wird wiederum in Kapitel 4 auf verschiedene Möglichkeiten ausführlich eingegangen. Mit einer detaillierten Betrachtung des entwickelten Programms, die unter anderem die Speicherung der gewonnenen Daten, Informationen und deren Schemata umfasst, beschäftigt sich Kapitel 5. Eine Untersuchung der erfassten Daten und Informationen hinsichtlich der Brauchbarkeit erfolgt in Kapitel 6. Letztendlich werden in der Schlussbetrachtung in Kapitel 7 Einsatz- und Erweiterungsmöglichkeiten besprochen.

2 Datenquellen

Als Datenquellen kann man in erster Linie zwei verschiedene Arten ausmachen. Zum einen sind dies gedruckte Werke. In dem hier betrachteten wissenschaftlichen Zusammenhang sind dies zumeist Artikel in Büchern und Zeitschriften, die den Stand der Forschung in regelmäßigen Abständen wiedergeben. Um eine computergestützte Informationsextraktion aufzubauen, müssten diese Publikationen vorerst digitalisiert werden. Dabei müssen verschiedene Hürden wie die Erkennung der Überschrift und Weiteres überwunden werden. Hierzu existiert ein eigener Forschungszweig, der sich mit Zeichenerkennung (*Optical Character Recognition, OCR*), künstlicher Intelligenz und Anderem befasst.

Da eine Digitalisierung im Rahmen dieser Arbeit aufgrund der angesprochenen Komplexität nicht durchgeführt werden kann, wird im Weiteren nur auf Veröffentlichungen eingegangen, die schon in digitalen Formaten vorliegen. Hierzu soll im ersten Unterkapitel eine Einführung und ein kurzer Überblick über digitale Quellen gegeben werden. Weitere Unterkapitel stellen Ausprägungen verschiedener, signifikanter Quellen im Sinne des Umfangs der Daten, deren Arten der Bereitstellung als auch eventuelle Zugangsbeschränkungen vor.
Wie im weiteren Verlauf der Arbeit festgestellt wird, existieren diverse Formate, mit denen Daten dem Nutzer zur Verfügung gestellt werden. In den folgenden Unterkapiteln erfolgt eine kurze Beschreibung solcher Formate und der Erörterung der Nützlichkeit für eine Datenextraktion. Detaillierter werden dagegen in dieser Arbeit genutzte Formate im Kapitel 3 erläutert.

2.1 Überblick digitaler Quellen

Wissenschaftliche Arbeiten werden nicht mehr nur in gedruckten Formaten veröffentlicht, sondern zunehmend auch in so genannten Bibliografiedatenbanken. Diese Verzeichnisse halten in der Regel alle Informationen zu der abhängigen Literatur als auch diese selbst vor. Abhängig bedeutet, dass die Werke im Rahmen einer Konferenz eines bestimmten Fachgebiet oder Ähnlichem entstanden und diesbezüglich in einer gemeinsamen Publikation (Buch, Zeitschrift, etc.) veröffentlicht worden sind.

Vorteile von Bibliografiedatenbanken sind beispielsweise die relativ günstige Vorhaltung aller Daten und Informationen. Diese müssen nicht unzählige Male gedruckt werden, im Hinblick auf die Abnehmerzahl, zum Beispiel durch Bibliotheken. Statt dessen kann auf die zentral gespeicherten Werke über das Internet mittels einer benutzerzentrierten Web-Schnittstelle zugegriffen werden. Über die Webseite kann der Zugriff differenzierend gestaltet werden. Nicht registrierte Nutzer erhalten zum Beispiel nur Einsicht in allgemeine Beschreibungen, andere können je nach Ertragsmodell auf einen Teil oder eventuell alle Publikationen zugreifen.
In den folgenden Kapiteln werden die Bibliografiedatenbanken *ACM Portal*, *DBLP* und *Scopus* aufgrund der Relevanz für die Abteilung Datenbanken vorgestellt. Weitere Beispiele sind:

- Elsevier mit *ScienceDirect*[1]
- MetaPress mit *MetaPress*[2]
- Springer mit *SpringerLink*[3]

Aufgrund der weitaus größeren Anzahl an Bibliografiedatenbanken kann, um die Verhältnismäßigkeit zu wahren, nicht jede hier aufgezählt und ausgeführt werden.

Weitere Datenquellen können Suchmaschinen oder spezielle Verzeichnisse im Web sein, die keinen beziehungsweise selten ausführlichen Inhalt (Volltext) anbieten, sondern hauptsächlich Metadaten darüber. Über vorgehaltene Verweise zu den Inhalten gelangt man somit oftmals zu Bibliografiedatenbanken oder andere Quellen wie beispielsweise Dokumentenserver von Universitäten.
Ein Beispiel solcher Suchmaschinen stellt Elsevier mit *Scirus*[4] bereit. Über ein Web-Formular lässt sich nach bestimmten Werken recherchieren. Nähere Vorstellungen finden in folgenden Kapiteln zu *CiteSeer* als auch *Google Scholar* statt. Für die Variante der Web-Verzeichnisse kann *Directory of Open Access Journals (DOAJ)*[5] beispielsweise genannt werden. Leitgedanke ist hier die Bereitstellung von Verweisen zu offenen, für jeden frei zugänglichen Artikeln. Solche Verweise werden von Hand in

[1] www.sciencedirect.com
[2] www.metapress.com
[3] www.springerlink.com
[4] www.scirus.com
[5] www.doaj.org

das Verzeichnis eingefügt und über Redakteure überprüft. Die Publikationen können über das jeweilige Fachgebiet oder den Titel nachgeschlagen werden. Ebenso bietet DOAJ eine Suchmöglichkeit durch ein Formular über die erfassten Dokumentdaten.

Eine exakte Abgrenzung für eine Quelle (Bibliografiedatenbank, Suchmaschine) lässt sich mitunter nicht durchführen, da die entsprechenden Funktionalitäten häufig gemischt bereitgestellt werden.

2.2 ACM Portal

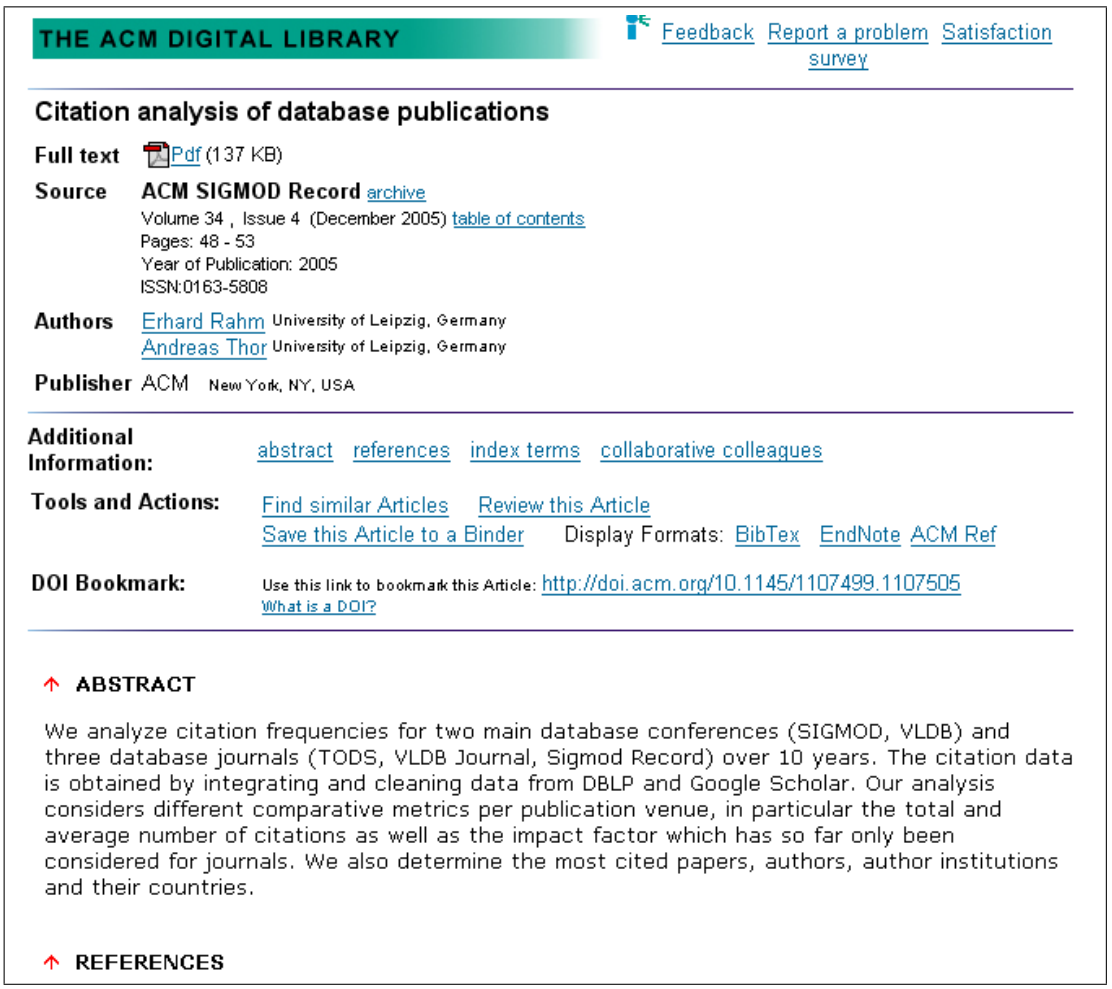
Die *Association for Computing Machinery* (ACM) bietet mit ihrem *ACM Portal*[6] im Web Zugang zur eigenen Bibliografiedatenbank. Das Angebot wird hier in zwei Teile gegliedert. So stellt der so genannte *Guide* eine Sammlung von Metadaten und Kurzbeschreibungen zu Publikationen bereit. Die *Digital Library* hingegen bietet auch Publikationen im Volltext von ACM selbst und kooperierenden Herausgebern an.
Beide Teile können zum Beispiel über das jeweilige Fachgebiet, die Art der Veröffentlichung (Buch, Zeitschrift, Konferenz, etc.) und Titel erforscht oder spezielle Inhalte über ein Suchformular recherchiert werden.

In den folgenden Unterkapiteln werden verschiedene Arten der Datenbereitstellung durch ACM diskutiert.

2.2.1 Abstract-Seiten

Über die Suchmöglichkeiten gelangt man zu so genannten Abstract-Seiten, wie sie in Abbildung 2 für die Publikation [RT05] aufgezeigt wird. Zu erkennen sind typische Metadaten, die die Publikation näher beschreiben. Dies sind zum Beispiel der Titel des Artikels, Titel als auch numerische Angaben zur Ausgabe, Namen der Autoren, ein Abstract und auch, im unteren Teil des Bildes nur angedeutet, die Referenzen, auf die sich die Publikation bezieht.

[6] portal.acm.org


THE ACM DIGITAL LIBRARY

Feedback Report a problem Satisfaction survey

Citation analysis of database publications

Full text Pdf (137 KB)

Source **ACM SIGMOD Record** archive
Volume 34 , Issue 4 (December 2005) table of contents
Pages: 48 - 53
Year of Publication: 2005
ISSN:0163-5808

Authors Erhard Rahm University of Leipzig, Germany
Andreas Thor University of Leipzig, Germany

Publisher ACM New York, NY, USA

Additional Information: abstract references index terms collaborative colleagues

Tools and Actions: Find similar Articles Review this Article
Save this Article to a Binder Display Formats: BibTex EndNote ACM Ref

DOI Bookmark: Use this link to bookmark this Article: http://doi.acm.org/10.1145/1107499.1107505
What is a DOI?

ABSTRACT

We analyze citation frequencies for two main database conferences (SIGMOD, VLDB) and three database journals (TODS, VLDB Journal, Sigmod Record) over 10 years. The citation data is obtained by integrating and cleaning data from DBLP and Google Scholar. Our analysis considers different comparative metrics per publication venue, in particular the total and average number of citations as well as the impact factor which has so far only been considered for journals. We also determine the most cited papers, authors, author institutions and their countries.

REFERENCES

Abbildung 2: Ausschnitt einer ACM-Abstract-Seite

Im Sinne der Aufgabenstellung sind vor allem die Angaben hinter den Autornamen interessant. Diese geben die Institutszugehörigkeit des jeweiligen Verfassers an, in diesen beiden Fällen jeweils `University of Leipzig, Germany`. Allerdings ist dies nicht der Regelfall, vor allem bei älteren Publikationen fehlen diese Angaben.
Für eine Datenextraktion sind weiterhin verschiedene in den Abstract-Seiten untergebrachte Verweise zu beachten. Einerseits führen sie zur eigentlichen Publikation im Volltext (`Full text Pdf`), andererseits werden in der Kategorie `Tools and Actions` die Verweise `BibTex`, `EndNote` als auch `ACM Ref` aufgeführt. Deren Einsatzzweck, sowie Vor- und Nachteile werden nachfolgend erörtert.

2.2.2 Volltext (PDF)

Das *Portable Document Format (PDF)* der Firma *Adobe Systems Inc.* ist heutzutage weit verbreitet, um Dokumente plattformübergreifend auszutauschen und dabei eine einheitliche Darstellung mittels frei verfügbarer Software zu gewährleisten.
Viele Abstract-Seiten von ACM beinhalten einen Verweis zu einem PDF-Dokument. Mittels solcher PDF-Dateien wird der Volltext der wissenschaftlichen Arbeiten zur Verfügung gestellt. Insofern ein solches Dokument existiert, ist allerdings der Zugang hierzu beschränkt. So benötigt man je nach Artikel eine Mitgliedschaft in einem speziellen Interessenverband oder beispielsweise ein Abonnement der jeweiligen Zeitschrift.
Im Hinblick auf die Aufgabenstellung lässt diese Tatsache PDF-Dateien als Datenquelle im Grunde nicht zu, da der Zugang nicht für die gesamte digitale Bibliothek von ACM vorausgesetzt werden kann. Faktisch besitzt die Universität Leipzig einen Vertrag mit ACM, der den Nutzern des Universitätsnetzes Einsicht in alle Volltexte bietet. Da die Aufstellung einer Forschungsdatenbank an dieser Hochschule durchgeführt wird, können damit die PDF-Dokumente weiter in Betracht gezogen werden.

Für wissenschaftliche Arbeiten existieren in der Regel Formatvorlagen und damit auch gewisse Vorgaben für den Inhalt (Metadaten) der entstehenden PDF-Dateien. So werden heutzutage üblicherweise die institutionellen Zugehörigkeiten zu den jeweiligen Autoren mit angegeben.

Darüber hinaus besteht ein Nachteil in der Verwendung von PDF-Dateien darin, dass sämtliche Dokumente zunächst erfasst werden müssen und somit ein hoher Datenverkehr erzeugt wird. Wie in Abbildung 2 zu sehen, beträgt die Größe der Volltextdatei rund 137KB, die im Bild aufgezeigte Webseite benötigt hingegen etwa ein Viertel (34KB). Dieser Unterschied kann noch viel ausgeprägter sein, da PDF-Dokumente oftmals Bilder enthalten und diese somit schnell mehr als 1MB an Speicherplatz benötigen. Abstract-Seiten hingegen stellen in Bezug auf die Veröffentlichung ausschließlich Text dar.

2.2.3 BibTeX

ACM stellt auf den Abstract-Seiten einen weiteren Verweis zu einer so genannten *BibTeX*-Datei bereit. BibTeX wird im Rahmen von *TeX* angewandt, einem Textsatzsystem, bei dem Dokumente durch spezielle Software gestaltet werden können. Dieses System wird oftmals für Publikationen von Wissenschaftlern eingesetzt wird. Die Arbeiten referenzieren dabei in der Regel andere wissenschaftliche Arbeiten, wofür wiederum der Einsatz von BibTeX gedacht ist.
Nachdem Literaturlisten nach einer bestimmten Syntax aufgestellt worden sind, untersucht BibTeX diese während der Erstellung von Dokumenten nach relevanten Einträgen und generiert anhand dessen ein Literaturverzeichnis [Gün02]. BibTeX-Dateien stellen demnach die zu einer Literatur relevanten Metadaten in strukturierter Form zur Verfügung.

Die von ACM bereitgestellten BibTeX-Einträge dienen den Wissenschaftlern als Hilfe, indem die Inhalte in die eigene BibTeX-Datei direkt kopiert und damit in die Literaturliste übernommen werden können. Somit entfällt ein ständiges Zusammensuchen der gewünschten Metadaten, wie sie beispielhaft für die Publikation [RT05] nachfolgend aufgeführt sind.

```
@article{1107505,
author = {Erhard Rahm and Andreas Thor},
title = {Citation analysis of database publications},
journal = {SIGMOD Rec.},
volume = {34},
number = {4},
year = {2005},
issn = {0163-5808},
pages = {48–53},
doi = {http://doi.acm.org/10.1145/1107499.1107505},
publisher = {ACM},
address = {New York, NY, USA}
```

Hinsichtlich der Problemstellung dieser Arbeit ist hierbei vor allem das Feld *address* interessant, da es Ortsangaben enthält. Allerdings wird hierbei die Adresse des Verlages beschrieben [MGB+05], nicht die der Autoren. Dies lässt sich ferner anhand

der Angaben in der zugehörigen Abstract-Seite vergleichen.
Die Autoren werden in das Feld *author* geschrieben. Dabei ist es unwesentlich, ob es mehrere sind. Deren Namen werden durch das Schlüsselwort *and* voneinander getrennt.
Somit wird ersichtlich, dass eine Zugehörigkeitsangabe der Autoren in BibTeX nicht vorgesehen ist. Für diese Arbeit stellen die BibTeX-Dateien demnach keine relevante Datenquelle dar.

2.2.4 Weitere Zitierungsformate

Außer BibTeX stellt ACM noch die Zitierungsformate *EndNote* und *ACM Ref* zur Verfügung. Dabei unterscheidet sich das Erste nur unwesentlich von BibTeX, wie im folgenden Beispiel ersichtlich wird.

```
%0 Journal Article
%1 1107505
%A Erhard Rahm
%A Andreas Thor
%T Citation analysis of database publications
%J SIGMOD Rec.
%@ 0163-5808
%V 34
%N 4
%P 48-53
%D 2005
%R http://doi.acm.org/10.1145/1107499.1107505
%I ACM
```

Einzelne Buchstaben und Ziffern beschreiben dabei den dahinter angegebenen Inhalt. So steht *A* beispielsweise für Autor. Da kaum semantischer Unterschied zu BibTeX auszumachen ist, bestehen die selben Vor- und Nachteile. Nachteilig kommt hinzu, dass sämtliche Ortsinformationen (zum Veröffentlicher) fehlen.

Mit dem *ACM Reference Format* (ACM Ref) wird formatierter Text angeboten, der unverändert in ein Literaturverzeichnis untergebracht werden kann, wie zum

Beispiel:

> Rahm, E. and Thor, A. 2005. Citation analysis of database publications. SIGMOD Rec. 34, 4 (Dec. 2005), 48-53. DOI= http://doi.acm.org/10.1145/1107499.1107505

Dieses Format ist gänzlich ungeeignet für eine Datenextraktion im Rahmen der Aufgabenstellung, da die für diese Arbeit notwendigen Daten der Autorzugehörigkeiten fehlen.

2.2.5 Überblick

Dieses Kapitel führt die vorgestellten Arten der Datenbereitstellung nochmals übersichtlich tabellarisch auf, zum Teil mit subjektiver Wertung, um anschließend Schlussfolgerungen ziehen zu können.

Art	**1**	**2**	**3**	**4**
Abstract-Seite	+	+	o	+
Volltext (PDF)	−	−	o	−
BibTeX	+	+	−	+
EndNote	+	+	−	+
ACM Ref	+	+	−	+

1 ... Existenz der Quellen (+ → ausnahmslos; − → mit Einschränkungen)
2 ... Zugang (+ → jederzeit möglich; − → beschränkt)
3 ... Existenz benötigter Daten (+ → immer; o → begrenzt; − → nie)
4 ... Dateigröße (+ → niedrig; − → hoch)

Tabelle 1: Überblick der Datenbereitstellung durch ACM

Da die Zitierungsformate nicht die gesuchten Daten vorhalten, bleiben als potentielle Quellen der Volltext und die Abstract-Seiten. Aufgrund der besprochenen Nachteile der PDF-Dateien bieten sich bei ACM in erster Linie die Abstract-Seiten als Datenquellen an.

2.3 CiteSeer

Die Pennsylvania State University stellt mit *CiteSeer.IST Scientific Literature Digital Library*[7], im Weiteren als *CiteSeer* bezeichnet, eine Suchmaschine und digitale Bibliothek wissenschaftlicher Literatur für Informatik und Informationswissenschaften zur Verfügung.
Das System hinter der Suchmaschine CiteSeer erfasst uneingeschränkt zugängliche Publikationen vollkommen automatisch, indem das so genannte Crawling (siehe hierzu Kapitel 5.3) im Web als auch andere Suchmaschinen genutzt werden [Kan05]. Außerdem ist das persönliche, manuelle Einreichen von wissenschaftlichen Arbeiten möglich.
Die aufgenommenen Publikationen werden maschinell verarbeitet, wobei der Volltext und dazugehörige Metadaten, wie Titel oder Autoren, extrahiert und indexiert werden. Hierdurch wird eine Suche auf diesen Daten möglich. Weiterhin wurde für CiteSeer ein System für die Behandlung von Zitierungen entwickelt, welches das erste seiner Art war. Es indexiert und verlinkt Zitierungen automatisch, so dass letztendlich CiteSeer „weiß“, welche Publikationen über Referenzen miteinander verbunden sind [GBL98].

Für Benutzer von CiteSeer stehen verschiedene Zugangsmöglichkeiten zu den Daten beziehungsweise Formate der Daten bereit, die im Weiteren näher beleuchtet werden sollen. Die gesamten Möglichkeiten sind dabei ohne Beschränkung zugänglich.
Insofern Verweise zu Seiten mit Metadaten einer Publikation (Abstract-Seite) von DBLP (siehe Kapitel 2.4) und der ACM Digital Library (Kapitel 2.2) vorhanden sind, werden diese außerdem angeboten.

2.3.1 Datenbereitstellung in Analogie zu ACM

Ähnlich ACM bietet auch CiteSeer Abstract-Seiten mit wichtigen Informationen zu Veröffentlichungen an. Diese können über ein Formular über Stichworte gesucht oder über eine Kategoriensuche, dem so genannten *Computer Science Directory*, erreicht werden.
Die Abstract-Seiten von CiteSeer enthalten die wichtigsten Daten zu den Publika-

[7] citeseer.ist.psu.edu

tionen wie Titel, Autorennamen und ein Abstract, nicht aber die hier benötigten Zugehörigkeitsinformationen der Autoren. Die angesprochenen Zitierungsverlinkungen zu anderen Veröffentlichungen machen einen essentiellen Teil der Abstract-Seiten aus. Außerdem sind die jeweiligen BibTeX-Einträge im Klartext auf den Seiten untergebracht.

Da CiteSeer die Publikationen automatisch aus dem Web bezieht, werden weiterhin Verweise zu den Quellen angegeben. Diese Quelldokumente werden bei CiteSeer zwischengespeichert und gegebenenfalls in andere Formate umgewandelt. Dadurch können verschiedene Formate für die Publikation angeboten werden. Auf den Abstract-Seiten finden sich somit zum Teil Verweise zu PS(PostScript)- und PDF-Dateien, die den Volltext beinhalten. Zusätzlich bietet CiteSeer die Publikationen als Bilddateien an, jeweils eine Datei für eine Volltextseite.

Die in diesem Kapitel aufgezeigten Datenquellen von CiteSeer kommen denen von ACM gleich. Ein gravierender Unterschied allerdings ist, dass die Autorzugehörigkeiten nicht in den Abstract-Seiten vorliegen, allerdings in den PS- sowie PDF-Dokumenten. Da PS-Dateien thematisch eng mit PDF-Dateien verbunden sind, beide stellen Sprachen zur Seitenbeschreibung dar, sind sie im Rahmen dieser Arbeit in ihren Vor- und Nachteilen als gleichwertig zu betrachten, die in Kapitel 2.2.2 für PDF-Dateien beschrieben wurden.
Die angesprochenen Bilddateien stellen ebenfalls ein einheitliches, für den Menschen lesbares Format des Volltextes dar. Eine maschinelle Verarbeitung allerdings erfordert beispielsweise durch den Einsatz von OCR-Techniken einen großen Aufwand, ähnlich dem für gedruckte Werke, wie zu Beginn des Kapitels 2 geschildert.
Weiterhin ist zu beachten, dass der Zugang zu jedem einzelnen Bild erfolgen muss. Für die zehn Seiten der Beispielpublikation [GBL98] beträgt die Größe der PDF-Datei etwa 133KB, die entsprechenden Bilder benötigen zusammen rund 320KB Speicherplatz. Durch den Zugriff auf die Bilddateien entsteht demnach ein wesentlich höheres Datenaufkommen, ein weiterer Nachteil für die Nutzung der Bilder als Datenquelle, vor allem wenn auf zahlreiche Publikationsdaten zugegriffen werden soll.

2.3.2 Weitere Bereitstellungsmethoden

Für die große Zahl (767.558) an in CiteSeer gespeicherten Dokumente werden sämtliche BibTeX-Einträge aufgeteilt auf 12 einzelne, gepackte Dateien zum Herunterladen angeboten[8]. Zum Teil enthalten diese Einträge eine URL (Uniform Resource Locator) zum originalen Volltext, im Allgemeinen aber sind Verweise zu den Abstract-Seiten von CiteSeer vorhanden. Wie schon behandelt, finden sich in diesen Dateien keine Zugehörigkeitsinformationen zu den Autoren. Weiterhin sind keine Informationen zur Aktualität der Dateien ersichtlich.

CiteSeer betont, dass es konform zum *Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH)* ist. Die *Open Archives Initiative (OAI)*[9] möchte den Zugang zu digitalen Ressourcen, speziell deren Metadaten verbessern, um damit schulische und wissenschaftliche Bildung zu fördern. Hierfür entwickelt sie Interoperabilitäts-Standards wie zum Beispiel das OAI-PMH, für das die Dokumentation auf der OAI-Internetseite zu finden ist. Diesbezüglich lassen sich die CiteSeer-Datensätze über ein Skript[10] programmatisch durchsuchen und herunterladen.
Die Datensätze können ebenso direkt komplett heruntergeladen werden. Diese bietet CiteSeer in zwei Formaten an. Einerseits gehört dazu der *Dublin Core Metadata Standard* der *Dublin Core Metadata Initiative (DCMI)*[11]. Diese Organisation entwickelt Metadaten-Standards für das Internet, um ein breites Spektrum an Verwendungszwecken zu unterstützen. Das OAI-PMH beruht auf diesen Standards. Die beschreibenden Elemente des Dublin Core Metadata Standards können auf der Internetseite der DCMI eingesehen werden. Dort wird ersichtlich, dass diese Elemente gleichermaßen nicht die Zugehörigkeit von Autoren berücksichtigen.
Andererseits bietet CiteSeer die Daten in einem zweiten Format an, die ebenso nach dem Dublin Core Metadata Standard aufgebaut, zusätzlich aber mit Informationen zu Zitierungsbeziehungen und den Autorzugehörigkeiten angereichert sind. Allerdings sind hierfür auf der Internetseite keine Informationen über die Aktualität ersichtlich. Die am 16.11.2007 testweise heruntergeladenen Dateien wurden, nach den Dateieigenschaften zu urteilen, zuletzt am 13.12.2005 geändert. Des Weiteren sind

[8] Stand: 16.11.2007
[9] www.openarchives.org
[10] cs1.ist.psu.edu/cgi-bin/oai.cgi
[11] www.dublincore.org

nicht zu jedem Autor die hier benötigten Zugehörigkeitsinformationen vorhanden.

2.3.3 Überblick

Art	**1**	**2**	**3**	**4**
Abstract-Seite	+	+	−	+
Volltext (PS, PDF, Bild)	+	+	o	−
BibTeX	+	+	−	+
nach DCMI Standard	+	+	−	+
nach DCMI Standard und Zusatzinformationen	+	+	o	+

1 ... Existenz der Quellen (+ → ausnahmslos; − → mit Einschränkungen)
2 ... Zugang (+ → jederzeit möglich; − → beschränkt)
3 ... Existenz benötigter Daten (+ → immer; o → begrenzt; − → nie)
4 ... Dateigröße (+ → niedrig; − → hoch)

Tabelle 2: Überblick der Datenbereitstellung durch CiteSeer

Im Gegensatz zu ACM bieten sich bei CiteSeer die Abstract-Seiten nicht als Quellen an, da die nötigen Daten fehlen. Statt dessen sind hier die herunterladbaren Dateien mit den Datensätzen nach DCMI Standard plus der eventuell vorhandenen Zusatzinformationen (Adresse und Institutszugehörigkeit der Autoren) in erster Linie von Interesse. Diese CiteSeer-Daten bieten sich allerdings in Ermangelung neuester wissenschaftlicher Publikationen ebenso nicht als Quelle an. Ein Grund hierfür kann sein, dass neue Veröffentlichungen im Internet meist in ihrem Zugang beschränkt sind und damit nicht automatisch erfasst werden können.

2.4 DBLP

Zunächst als Bibliografiedatenbank für „DataBase systems and Logic Programming“ entwickelt, bietet die *Digital Bibliographie & Library Project (DBLP)*[12] der Universität Trier mittlerweile Informationen zu Veröffentlichungen im gesamten Bereich der Informatik.

[12] dblp.uni-trier.de

2.4.1 Datenbereitstellung

Das Projekt DBLP versteht sich als zentrale Anlaufstelle, um Publikationen zu suchen, wobei aufgrund von eventuellen Urheberrechtsverletzungen selbst keine Volltexte vorgehalten werden.
DBLP bietet Verzeichnisse zum Beispiel über wissenschaftliche Zeitschriften oder Konferenzen mit den wichtigsten Metadaten als auch Daten zu den darin erschienenen Artikeln (Titel, Autoren, Seitenzahlen). Letztere werden auch als BibTeX-Eintrag zur Verfügung gestellt, wobei in beiden Fällen die benötigten Angaben für die Problemlösung dieser Arbeit fehlen.
Ein wichtiger Bestandteil ist der jeweilige Verweis `Electronic Edition` zu jeder Publikation in den Verzeichnissen. Der Inhalt im referenzierten Dokument ist allerdings sehr variabel. Insofern der Volltext als PDF-Datei frei zugänglich ist, wird dieser referenziert. Bei einigen Publikationen wird auf die Abstract-Seiten der Anbieter verwiesen, bei anderen Artikeln bietet DBLP eigene solcher Seiten mit verschiedener Qualität der Inhalte. Gelegentlich fehlen solche Verweise auch.
Die Inkonsistenzen in der Beschaffenheit auf Artikelebene entspringen augenscheinlich daher, dass Redakteure wissenschaftlicher Medien angehalten sind, bei dem Aufbau des Datenbestands zu helfen. Diese liefern die Informationen, deren Detailgrad und Ausrichtung nicht strikt vorgegeben sind und daher schwanken.
Weiterhin wurde DBLP, wie es der Dokumentation zu entnehmen ist, nicht von Grund auf so geplant, wie es sich dem Nutzer heute darstellt. Vielmehr stellt es ein stetig gewachsenes System dar, was schon anhand der Benennung deutlich wird.

Zusätzlich zu den Übersichtsseiten stellt die Universität Trier die in DBLP vorhandenen Daten zusammengeführt in einer aktuell gehaltenen XML-Datei (siehe Kapitel 3.4) zum Herunterladen bereit. Ein Teil an darin aufgeführten Publikationen beinhaltet Zugehörigkeitsinformationen beziehungsweise Adressangaben zu jeweiligen Autoren.

2.4.2 Überblick

Art	1	2	3	4
Abstract-Seite	−	+	−	+
BibTeX	+	+	−	+
Volltext (PDF)	−	+	o	−
XML	+	+	o	+

1 ... Existenz der Quellen (+ → ausnahmslos; − → mit Einschränkungen)
2 ... Zugang (+ → jederzeit möglich; − → beschränkt)
3 ... Existenz benötigter Daten (+ → immer; o → begrenzt; − → nie)
4 ... Dateigröße (+ → niedrig; − → hoch)

Tabelle 3: Überblick der Datenbereitstellung durch DBLP

Die DBLP bietet sich für die Lösung der Problemstellung bedingt an. Die einzige einheitliche Quelle, der hier benötigten Daten, stellen die XML-Dateien dar. Das restliche Datenangebot ist hohen Unstetigkeiten unterworfen.

2.5 Google Scholar

Die Firma *Google Inc.* bietet mit *Google Scholar*[13] eine Suchmöglichkeit für wissenschaftliche Literatur im gesamten Web.

> *„Dazu gehören von Kommilitonen bewertete Seminararbeiten, Magister-, Diplom- sowie Doktorarbeiten, Bücher, Zusammenfassungen und Artikel, die aus Quellen wie akademischen Verlagen, Berufsverbänden, Magazinen für Vorabdrucke, Universitäten und anderen Bildungseinrichtungen stammen.“* [Goo08]

2.5.1 Datenbereitstellung

Über ein Formular lässt sich nach Schlagwörtern suchen, wobei über eine erweiterte Suche die Recherche nach speziellen Arbeiten weiter eingeschränkt werden kann. Ein Beispiel einer solchen Suche und deren Ergebnis ist in Abbildung 3 dargestellt.

[13] scholar.google.de

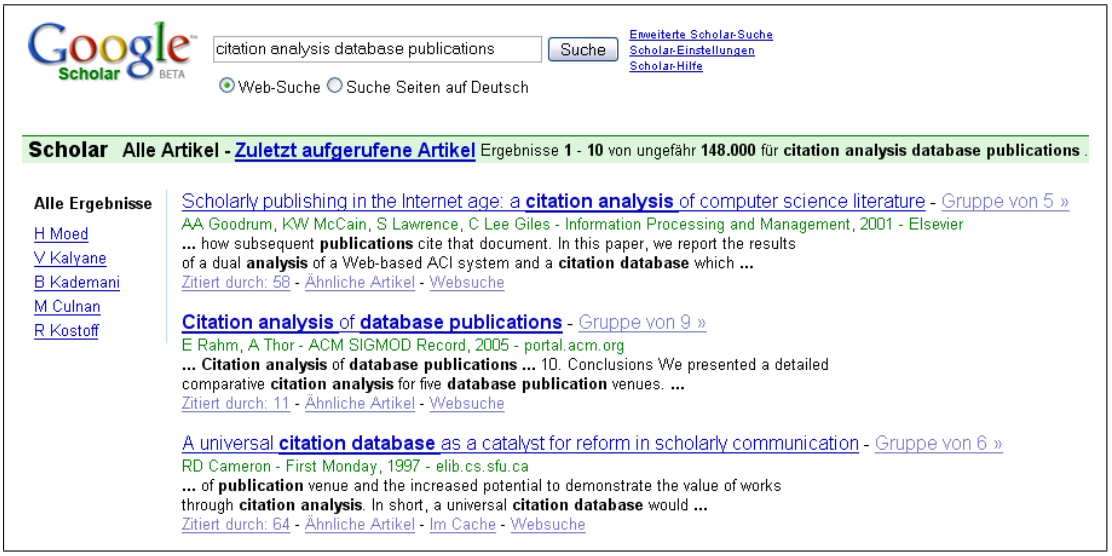


Abbildung 3: Ausschnitt einer Google-Scholar-Ergebnissseite

Es findet eine Auflistung der gefundenen Literatur statt, wobei unter dem Titel der jeweiligen Publikation verschiedene Metadaten aufgeführt werden. Diese erstrecken sich über die Autorennamen, Daten zur Quelle als auch des Veröffentlichers. Weiterhin ist ein kurzer Inhaltsauszug angegeben, in dem die gesuchten Schlagwörter aufgeführt sind.
Die auf diesen Ergebnisseiten aufgeführten Daten eignen sich in der Regel nicht für eine Extraktion im Sinne dieser Arbeit. Da bei Google Scholar eine einheitliche Gestaltung der Seiten eingehalten wird, werden reichhaltige Daten, zum Beispiel viele Autorennamen, abgeschnitten und dies mit drei Folgepunkten dem Benutzer zu verstehen gegeben.

Weiterhin fehlen für eine Datenextraktion in der Ergebnisauflistung Angaben zu den Zugehörigkeiten der Autoren. Wenn den Schlagwörtern zur Suche, die in Abbildung 3 aufgezeigt wurde, beispielsweise der Begriff `University` hinzugefügt wird, ändert sich der dargestellte Inhalt für die hier fiktiv gesuchte Publikation [RT05]. Im Ergebnis sind in diesem Fall der Ort und E-Mail-Adressen erkennbar, wie in Abbildung 4 veranschaulicht.

Abbildung 4: Ausschnitt eines erweiterten Google-Scholar-Suchergebnisses

Somit ist eine Datenextraktion aus den Ergebnisseiten prinzipiell denkbar. Allerdings wird dabei vorausgesetzt, dass zusätzliche Informationen, wie beispielsweise des Einrichtungstyps, zu Publikationen bereits bekannt sind, um sie den Schlagwörtern der Suche hinzuzufügen und diese damit detaillierter auszuführen.
Weiterhin kann nicht ausgeschlossen werden, dass die dargestellten Ergebnisse, vor allem die des Inhaltsauszugs, nicht aus dem Volltext der Veröffentlichung selbst stammen und damit möglicherweise nicht mit den Autoren in Verbindung zu bringen sind. So könnte die Ergebnis-Zeichenkette `University of Leipzig`, die in Abbildung 4 ersichtlich wird, aus dem Publikationstext oder einer anschließenden Referenzliste stammen. Der Inhalt der Ergebnisse unterliegt demnach der Willkür von Google Scholar.

Die Suchmaschine Google Scholar bietet weiterhin auf den Ergebnisseiten Verweise zu den verschiedensten Inhalteanbietern. So führt die Verknüpfung mit dem Titel `Citation analysis of database publications` beispielsweise zur entsprechenden Abstract-Seite von ACM, siehe Abbildung 2. Eventuell folgende Verweise in der Form `Gruppe von ...` zeigen eine weitere Ergebnisseite auf, die wiederum zu einem Publikationstitel alle gefundenen Quellen auflistet.

Gleichgültig auf welcher Seite ein Ergebnis aufgeführt ist, die damit verbundenen Inhalte stellen sich dem Nutzer in mindestens einer der folgenden Arten dar:

- **Abstract-Seite** der Veröffentlicher (Verlag, Universität, etc.). Die Anzahl der Veröffentlicher ist dabei gewissermaßen unbegrenzt, wodurch auch Inhalt und Aufbau dieser Internetseiten zueinander differieren.
- **Volltext** (PDF, PS) des Veröffentlichers. Für wissenschaftliche Publikationen bestehen meist Formatvorlagen, wodurch sich dem Benutzer diese Dokumente relativ einheitlich darstellen. Dennoch unterscheiden sie sich mehr oder weniger in Details der Formatierung und dem Umfang bezüglich der Metadaten. Da auf die Volltexte im Original verwiesen wird, gelten dort die Restriktionen der Veröffentlicher. Ein Volltext (des Veröffentlichers) kann somit auch im Zugang beschränkt sein.
- Von Google Scholar zwischengespeicherte **Internetseiten als auch Volltext**, als Internetseite vorgehalten. Diese Seiten werden direkt von Google Scholar

bereitgestellt und stellen einen Schnappschuss des Inhalts zur Zeit der Erfassung des Originals dar. Hier erfolgt eine Umwandlung in die *Hypertext Markup Language (HTML)*, einer Beschreibungssprache für Web-Dokumente [BRS03]. Die Seiten bieten im Endeffekt die gleichen Vor- und Nachteile wie bei den Originalen (Abstracts und Volltext) bezüglich der Existenz der hier benötigten Daten.

Bei der automatischen Erfassung von Volltextdokumenten wird die für den Nutzer sichtbare Gestaltung möglichst übernommen. Zum Teil bestehen allerdings Unterschiede zum Original. Bilder beispielsweise werden eventuell nicht übernommen, die allerdings auch nicht für diese Arbeit notwendig sind.

Da Abstract-Seiten bereits als HTML-Dokumente vorliegen, besteht bei dieser Zwischenspeicherung inhaltlich keine Änderung.

Weitere Arten der Datenbereitstellung, wie zum Beispiel BibTeX-Dokumente, existieren bei Google Scholar nicht.

Insofern der jeweilige Inhalt nicht zwischengespeichert wurde, kann es vorkommen, dass in der Ergebnisliste von Google Scholar veraltete Verweise vorliegen, die Quelle demnach nicht mehr vorhanden ist oder sich der Inhalt geändert hat.

2.5.2 Überblick

Art	**1**	**2**	**3**	**4**
Abstract-Seite, auch zwischengespeichert	−	+	o	+
Volltext (PDF, PS), auch zwischengespeichert	−	−	o	−
Volltext (HTML), zwischengespeichert	−	+	o	+

1 ... Existenz der Quellen (+ → ausnahmslos; − → mit Einschränkungen)
2 ... Zugang (+ → jederzeit möglich; − → beschränkt)
3 ... Existenz benötigter Daten (+ → immer; o → begrenzt; − → nie)
4 ... Dateigröße (+ → niedrig; − → hoch)

Tabelle 4: Überblick der Datenbereitstellung durch Google Scholar

Jede der angegebenen Datenbereitstellungsmethoden kann in einer Ergebnisliste vorkommen, keine davon wird allerdings immer angezeigt. Weiterhin halten diese Methoden die hier benötigten Daten nur stellenweise vor, zudem nicht durchweg einheitlich.

Zusätzlich besteht die Problematik, dass ein relevantes Ergebnis aus der Auflistung herausgesucht werden muss. Einen Algorithmus für eine programmatische Auswahl zu erstellen, erfordert einen Mehraufwand.
Aus diesen Gründen kann Google Scholar nicht als primäre Datenquelle herangezogen werden.

2.6 Scopus

Elsevier ist nach eigenen Angaben[14] der weltgrößte Herausgeber in den Bereichen der Wissenschaft und Technik als auch der Gesundheitswissenschaften. So bietet es neben dem schon erwähnten *ScienceDirect* (siehe Kapitel 2.1) weiterhin das Portal *Scopus*[15]. Der Internetseite zu diesem Angebot kann entnommen werden, dass Scopus die größte Abstract- und Zitierungsdatenbank für wissenschaftliche Literatur sei, wobei ebenso frei zugängliche Journale, Patente als auch Internetseiten von Wissenschaftlern erfasst wurden.

2.6.1 Zugangsbeschränkungen

Um via Web-Browser in der Datenbank von Scopus recherchieren zu können, sieht sich der Nutzer zuvor allerdings einigen Restriktionen gegenüber. Zum einen muss im Browser die Annahme von so genannten *Cookies* aktiviert sein. Mit diesen werden verschiedene Parameter an den Browser übertragen, der die jeweiligen Inhalte als Textdateien, den Cookies abspeichert. Bei jedem Aufruf einer Internetseite des Angebots, hier Scopus, werden diese Parameter an den Server übertragen. Somit lässt sich für den Dienstanbieter ein Nutzer wieder identifizieren und dessen Sitzung nachvollziehen [BRS03].
Mit der Akzeptanz von Cookies hat der Benutzer jedoch noch nicht automatisch Zugang zu den Daten. Scopus bietet diesen Zugang lediglich für gesamte Forschungseinrichtungen an. Diese müssen sich für eine Zugangsberechtigung unter Angabe verschiedenster Daten anmelden, darunter einen Bereich an IP-Adressen, der vom

[14] „ELSEVIER AT A GLANCE“, www.elsevier.com/wps/find/intro.cws_home/ataglance, 21.11.2007
[15] www.scopus.com

jeweiligen Institut genutzt wird. Eine IP-Adresse weist den Rechner des Benutzers aus. Fällt diese in den spezifizierten Bereich, erhält der Nutzer Zugang zu Scopus.

Mit den oben genannten beiden Sicherheitsvorkehrungen möchte Elsevier sicher gehen, dass nur autorisierte Nutzer Zugang zum gesamten Angebot erhalten und deren Recherche nachvollziehbar ist. Somit wird dem Missbrauch des Angebots vorgebeugt. Dieser kann darin bestehen, dass programmatisch versucht wird, die Inhalte auszulesen, wie beispielsweise mittels Suchmaschinen. Die auf der Internetseite aufgeführten Geschäftsbedingungen besagen zudem:

> *„You agree that you will not use any robots, spiders, crawlers or other automated downloading programs or devices to: (i) continuously and automatically search or index and Content, unless authorized by us; (ii) harvest personal information from the Site for purposes of sending unsolicited or unauthorized material; or (iii) cause disruption to the working of the Site.“* [Els04]

Bei einer Verletzung der Nutzungsbedingungen kann dies bis hin zu einem Ausschluss des Instituts führen.

Scopus bietet mit *Scopus Custom Data*[16] interessierten Forschungseinrichtungen an, speziell auf die Bedürfnisse angepasste Daten aus der eigenen Datenbank zu liefern. Anhand der auszuhändigenden Daten werden die Kosten festgelegt, die hierfür gezahlt werden müssen.
Dieses Geschäftsmodell erfordert die voran beschriebenen Zugangsbeschränkungen, damit kein anderer Weg existiert, an die Daten zu gelangen.

2.6.2 Überblick

Scopus bietet dem Benutzer eine einfache als auch erweiterte Suche nach Publikationen über Schlagwörter mittels Formular an. Weiterhin lässt sich die Datenbank über Fachbereiche, Publikationsart und dem Titel nach bestimmten Veröffentlichungen recherchieren. Letztendlich gelangt man zu Abstract-Seiten, die denen von ACM

[16] info.scopus.com/customdata

ähnlich sind. Neben den regulären Metadaten finden sich auch häufig zu den Autorennamen die Zugehörigkeitsinformationen.
Darüber hinaus bietet Scopus die Möglichkeit, die Daten zu exportieren. Als mögliche Dateiformate stehen zum Beispiel normales Textformat und so genannte *Comma-Separated-Value-Dateien (CSV)* zur Auswahl. Zudem kann der Umfang des Inhalts gewählt werden, beispielsweise nur im Zitierungsformat mit wenigen Daten wie Autornamen, Titel und Jahr, oder komplett mit dem Text des Abstracts, Veröffentlicher- und den Autorzugehörigkeitsinformationen. Ein Abstractformat als Text stellt sich für die Publikation [RT05] gekürzt wie folgt dar.

> Scopus
> EXPORT DATE: 21 November 2007
>
> Rahm, E., Thor, A.
> Citation analysis of database publications
> (2005) SIGMOD Record, 34 (4), pp. 48-53. Cited 3 times.
> http://www.scopus.com/scopus/inward/record.url?eid=[...]
> AFFILIATIONS: University of Leipzig, Germany
> ABSTRACT: We analyze citation frequencies for two [...]
> DOCUMENT TYPE: Article
> SOURCE: Scopus

Eine Ausgabe etwa im BibTeX-Format ist ebenso möglich. Wie Scopus sich selbst beschreibt, ist es eine Abstract- und Zitierungsdatenbank. Das bedeutet, dass keine Volltexte vorgehalten werden. Statt dessen existieren jeweils Verweise zu den entsprechenden Veröffentlichern.

Art	**1**	**2**	**3**	**4**
Abstract-Seite	+	−	o	+
verschiedene Ausgabeformate	+	−	o	+

1 ... Existenz der Quellen (+ → ausnahmslos; − → mit Einschränkungen)
2 ... Zugang (+ → jederzeit möglich; − → beschränkt)
3 ... Existenz benötigter Daten (+ → immer; o → begrenzt; − → nie)
4 ... Dateigröße (+ → niedrig; − → hoch)

Tabelle 5: Überblick der Datenbereitstellung durch Scopus

Mit Scopus besteht ein großes Angebot, oft angereichert mit den hier benötigten Daten. Zugangsbeschränkungen lassen ein programmatisches Auslesen allerdings nicht zu. Statt dessen lassen sich über den Dienst namens Scopus Custom Data benutzerdefinierte Daten erhalten. Der hierfür nötige Kostenaufwand ist für diese Aufgabe nicht gerechtfertigt.
In Folge dessen ist Scopus als Quelle nicht relevant. Vielmehr muss ein Programmentwurf die Einbindung verschiedener Quellen zulassen, damit Daten von beispielsweise Scopus nachträglich noch genutzt werden können.

2.7 Zusammenfassung

Wie dem Inhalt dieses Kapitels 2 zu entnehmen ist, existiert eine Vielzahl an möglichen Datenquellen. Diese Arbeit legt den Schwerpunkt auf eine prototypische Umsetzung von Daten- und Informationsextraktion zu wissenschaftlichen Publikationen. Hierfür sollen verschiedene Wege aufgezeigt und evaluiert werden, die zum Teil Inhalt der Umsetzung sein sollen. Aufgrund der Fokussierung auf Umsetzungsmöglichkeiten muss eine Auswahl bezüglich der Priorität an zu nutzenden Datenquellen getroffen werden. Die hohe Anzahl schließt eine vollständige Einbindung aller Quellen aus.

Die Abteilung Datenbanken, von der die hier bearbeitete Aufgabe stammt, beschäftigt sich hauptsächlich mit ACM, DBLP und Google Scholar als Datenquellen für ihre Arbeiten. Deshalb sind diese zu bevorzugen.
Des Weiteren fällt CiteSeer als Quelle weg, da offensichtlich neueste Publikationen fehlen. Scopus ist aufgrund seines reichhaltigen Datenangebots für die Abteilung Datenbanken von großem Interesse. Daten können beziehungsweise dürfen dort allerdings nicht automatisch gewonnen werden. Somit entfällt auch der Inhalt von Scopus als Datengrundlage für diese Arbeit.
Die Suchmaschine Google Scholar bietet eine allgemeine, zentrale Anlaufstelle für die Suche nach wissenschaftlicher Literatur im Web. In Folge dessen stellen sich die Ergebnisse als stark heterogen heraus, was eine Datenextraktion erschwert. Zudem wird zum Teil auf Bibliografiedatenbanken wie ACM verwiesen, welche ebenso als Datenquelle vorgesehen ist. Aus diesen Gründen und den im Kapitel 2.2 behandelten Eigenschaften soll ACM hier als primäre Datenquelle dienen, um daran den weiteren Verlauf dieser Arbeit zu gestalten.

Mit DBLP können die Vorgänge von der Datenextraktion bis zum Informationserhalt anhand eines anderen Formates aufgezeigt werden. Folglich ist DBLP mit den herunterladbaren XML-Dateien als sekundäre Datenquelle zu behandeln.

3 Datenextraktion

Aufgrund der hohen Anzahl an möglichen Datenquellen soll in diesem Kapitel erläutert werden, wie der damit bestehenden Heterogenität in einem Prozess allgemein begegnet werden kann. Anhand der ausgewählten Beispielquellen ACM und DBLP soll die Einbeziehung verschiedener Datenformate betrachtet werden.

3.1 Vergleich zur Informationsextraktion

Um den Unterschied zwischen Daten- und Informationsextraktion zu verdeutlichen, müssen die grundlegenden Definitionen von Datum und Information betrachtet werden. Hierfür existieren jedoch keine allgemein gültigen Begriffsklärungen.
Im Bereich des Information Retrieval Daten von [Kor97] dadurch bestimmt, dass sie von einem Informationssystem erhalten, gespeichert und wieder aufgefunden werden können. Daten sind weiterhin sachlich, also ohne personellem Zusammenhang, und für jeden Nutzer des Systems gleichermaßen verfügbar. Eine Information hingegen ist eine Zusammenstellung von Daten, die entsprechend eines bestimmten Informationsbedürfnisses zusammen gehören. Demnach kann mit Informationen Wissen für den Nutzer generiert werden. Informationen besitzen nach [Kor97] also, im Gegensatz zu Daten, zeitliche und personengebundene Aspekte.

[Koe06] grenzt die Ausdrücke Datum und Information technischer voneinander ab. Datum stammt aus dem Lateinischen (*etwas Gegebenes*), kann demnach als etwas Existierendes, gewissermaßen als ein Ausgangsstoff interpretiert werden. Im Rahmen der Informatik allerdings besteht kein Ausgangsstoff. Vielmehr ist ein Datum nach [Koe06] eine Syntax, die auf eine Reihe an Zeichen angewandt wird. Dieses Datum besitzt maßgebliche Eigenschaften. Es kann generiert, gespeichert, gelöscht, kopiert, besessen, weitergegeben und gestohlen werden. Informationen werden als interpretierte Daten beschrieben, also mit einer Semantik (Bedeutung). Für eine Weiterleitung von Informationen werden diese in Daten transformiert, vom Empfänger neu interpretiert und damit in (eventuell andersartige) Informationen transformiert.

Basierend auf den vorgehenden Begriffsbestimmungen und der zugrunde liegenden Aufgabenstellung soll ein Datum hier als eine Zeichenkette ohne Bedeutung angese-

hen werden. Sobald einem Datum über Metadaten eine Semantik zugeordnet wird, handelt es sich hierbei um Informationen. Diese Semantik stellt eine Aussagekraft für den Nutzer der Daten dar, ohne die sie nicht angemessen verarbeitet werden können.

Demzufolge ist die Überschrift des Kapitels 3 nicht hundertprozentig korrekt. Denn mit den folgend erläuterten Extraktionsmethoden können neben Daten ebenso deren Semantik und damit Informationen erhalten werden. Dabei kann es sich beispielsweise um einen Publikationstitel oder Autorenbezeichnungen handeln. Doch in einem besonderen Fokus dieser Arbeit steht die Extraktion von Informationen zu den Zugehörigkeiten der Autoren, die spezielle Verfahren voraussetzt. Die extrahierbaren Zugehörigkeitsdaten sind vielfältig in ihrem Inhalt, so dass zunächst strukturierte Informationen gewonnen werden müssen. Aus diesem Blickwinkel werden im aktuellen Kapitel Daten und einfache Informationen, im anschließenden Kapitel 4 hingegen die Informationen mit besonderem Interesse und in Folge dessen mit eigenen Extraktionsmethoden gewonnen.

3.2 Extraktionsarchitektur

Im Rahmen von Integrationssystemen besteht die Notwendigkeit, Daten aus mehreren, voneinander unabhängigen Quellen so zu extrahieren, dass anschließend ein einheitliches Datenschema vorliegt. [RV03] schildert, auf welchen Ebenen sich Probleme aufgrund der Heterogenität der Datenquellen, insbesondere des Web, äußern können:

- *„auf Systemebene in Form unterschiedlicher Anfrageschnittstellen,*
- *auf Datenmodellebene durch die Verwendung von HTML, XML, relationaler oder objektorientierter Schemata,*
- *auf Schemaebene durch unterschiedliche Modellierung des Weltausschnittes und schließlich*
- *auf Datenebene durch verschiedene Repräsentationen ein und desselben Sachverhaltes.“* [RV03]

Hinsichtlich der Umsetzung einer Integration werden nach [RV03] zwei Varianten unterschieden. Bei einer *Materialisierung* werden benötigte Daten aus den jeweiligen Quellen extrahiert und zentral gespeichert. Eine relative Aktualität der Daten erfolgt durch periodisches Abfragen. Beim *virtuellen Ansatz* werden hingegen Sichten auf die Quelldaten eingesetzt. Eine Abfrage der Quellen und damit der Daten findet zur Laufzeit der Datenermittlung statt, wodurch die Daten zum jeweiligen Zeitpunkt aktuell erhalten werden können.

> *„Beiden Ansätzen gemeinsam ist jedoch die Lösung von Problemen der Datenextraktion aus den Quellen sowie die Überwindung von Integrationskonflikten.“* [RV03]

Für die zweite, virtuelle Variante entwickelte Wiederhold [Wie92] die Idee einer so genannten *Mediator-Wrapper-Architektur* auf, wie sie in Abbildung 5 gezeigt wird.

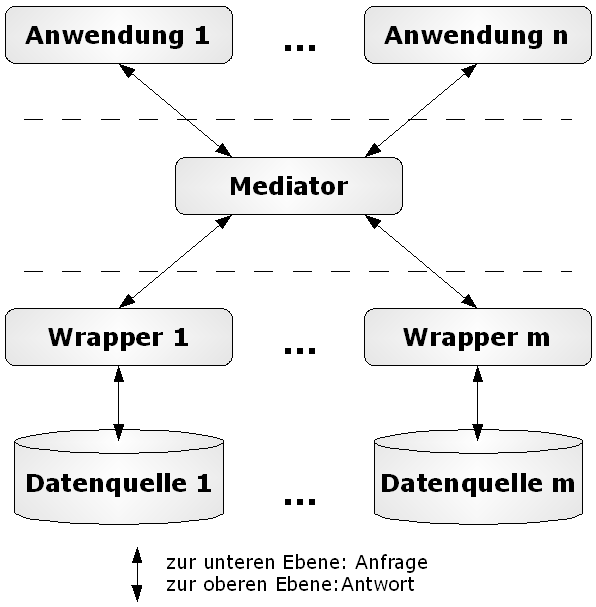


Abbildung 5: Mediator-Wrapper-Architektur

Wiederholds Architekturkomponenten, deren Zusammenarbeit als auch deren Ziele werden in [LN07] kurz folgendermaßen beschrieben:

> *„Wrapper sind zuständig für den Zugriff auf einzelne Datenquellen. Dabei überwinden sie Schnittstellen-, technische, Datenmodell- und schematische Heterogenität. Mediatoren greifen auf einen oder mehrere Wrapper*

> *zu und leisten einen bestimmten Mehrwert, in der Regel die strukturelle und semantische Integration von Daten.“* [LN07]

Der Nutzen von Wrappern liegt demnach darin begründet, dass man mit ihnen dem größten Teil der oben von [RV03] zitierten Probleme begegnen kann. Um verschiedene Repräsentationen eines Sachverhaltes zusammenzuführen, werden Mediatoren eingesetzt. Dabei kann allerdings auch ein Wrapper durchaus Datentransformationen durchführen, um den Mediator in dieser Hinsicht zu unterstützen. Diesbezüglich wird in [LN07] zwischen zwei Arten an Wrappern unterschieden. *Leichtgewichtige* dienen demnach der reinen Übersetzung, wobei *schwergewichtige* Wrapper zusätzliche Funktionen besitzen.

In [LN07] wird die Definition eines Mediators nach [Wie92] folgendermaßen ins Deutsche übersetzt:

> *„Ein Mediator ist eine Softwarekomponente, die Wissen über bestimmte Daten benutzt, um Informationen für höherwertige Anwendungen zu erzeugen.“* [LN07]

Demnach werden die Daten der Wrapper genutzt, um im Rahmen einer Anwendung gewünschte Informationen zu konstruieren.

Der Definition eines Mediators ist zu entnehmen, dass es sich dabei um eine Softwarekomponente, eine eigenständig ausführbare Einheit handelt. Darüber hinaus wird ein Wrapper im gleichen Sinne als ein Programm definiert, das zwischen der Integrationskomponente (Mediator) und den Datenquellen eingesetzt wird.
Im Rahmen dieser Arbeit soll ein Prototyp entwickelt werden, der in der Lage ist, mit heterogenen Quellen umzugehen. Um den Implementierungsaufwand nicht unnötig zu steigern, werden allerdings nicht für jede Quelle und den Mediator jeweils ein Programm umgesetzt. Statt dessen liegt hier eine Definition zugrunde, die die Begriffe einerseits als unabhängig bezeichnet, wobei diese nicht selbstständig lauffähig sind. Mediator und die verschiedenen Wrapper sind demnach in einem Programm vereint, in ihrem Implementierungsentwurf allerdings losgelöst voneinander.
Vielmehr wird hier der Fokus darauf gelegt, dass Wrapper eine Art Hülle um eine Datenquelle bilden, um nach außen hin eine einheitliche Sicht mit gleichen Funktionen

auf diese Quelle zu gewährleisten. Weiterhin legt [RV03] fest, dass Mediatorsysteme in den Kontext einer virtuellen Integration fallen. Allerdings wird auch gesagt, dass eine exakte Abgrenzung zwischen virtueller und materialisierter Integration schwierig ist. So stellt es sich in dieser Arbeit dar, in der heterogene Web-Datenquellen abgefragt, mit erhaltenen Daten neue Informationen erschaffen und in einer zentralen Datenbank gespeichert werden.

In den folgenden Kapiteln werden die Möglichkeiten der Umsetzung solcher Wrapper erörtert, die in dieser Arbeit für die genannten Datenquellen erforderlich sind. Speziell wird darauf eingegangen, wie die Daten aus den verschiedenen Formaten extrahiert werden können. Es wird somit davon ausgegangen, dass die Dokumente bereits für den jeweiligen Wrapper vorliegen.

3.3 Wrapper für HTML-Dokumente

Die *Hypertext Markup Language (HTML)* ist eine in den Jahren 1989/1990 entwickelte Seitenbeschreibungssprache für die Beschreibung von Web-Seiten [BRS03], wie es die hier betrachteten Abstract-Seiten sind. Dabei ist sie in der Lage, Texte zum Beispiel mit Bildern zu vereinen. Ein Web-Browser übernimmt auf dieser Basis die Darstellung des Inhalts für den menschlichen Nutzer. Im Allgemeinen existieren heute weitere Bestandteile für die Darstellung von Web-Seiten wie die Formatierungssprache *Cascading Style Sheets (CSS)*, die im Rahmen dieser Arbeit allerdings irrelevant sind.

> *„Der Zusatz Hypertext bezieht sich auf die Fähigkeit, multimediale Inhalte in einem Dokument zu integrieren und vor allem auf die Möglichkeit, einzelne Dokumente über Hyperlinks [Verweise] zu verknüpfen.“* [BRS03]

Diese Querverweise spielen auch in dieser Arbeit eine Rolle, um von einer Web-Seite auf weitere wichtige schließen zu können. Ein Hyperlink beschreibt grundsätzlich eine *Uniform Resource Locator (URL)*, in diesem Rahmen einen eindeutigen Verweis auf ein bestehendes Web-Dokument. Wenn solch eine URL aus einer HTML-Seite extrahiert wird, kann damit die nächste Seite aufgerufen werden. Damit gehören diese Querverweise mit zu den hier zu extrahierenden Daten.

HMTL-Seiten bestehen aus bestimmten Elementen, den so genannten *Tags*, die den Text (Inhalt) zwischen ihnen beschreiben und sich von diesem durch spitze Klammern abgrenzen (`<Tag>Inhalt`). Die einsetzbaren Tags mit möglichen Attributen und deren Reihenfolge gibt dabei eine *Document Type Definition (DTD)* vor, auf die zu Beginn in der HMTL-Seite verwiesen werden muss. Auf eine detailliertere Betrachtung wird hier verzichtet, statt dessen sei beispielsweise auf [BRS03] verwiesen.

Vielmehr soll hier die Fragestellung behandelt werden, wie sich Wrapper für solche HTML-Seiten erstellen lassen, als auch welche Möglichkeiten der Unterstützung hierfür existieren. [Eik99] bietet eine Übersicht über verschiedene Verfahren der Erstellung:

- Wrapper können zum einen *manuell* erstellt werden, wobei zu Beginn die Dokumentenstruktur verstanden werden muss, um anschließend den Programmcode umsetzen zu können. Bei semistrukturierten Dokumenten (HTML-Seiten) lässt sich dies einfacher realisieren als bei unstrukturierten wie zum Beispiel freiem Text. Aufgrund des hohen Zeitaufwands und der Notwendigkeit an Expertenwissen ist diese Art eher geeignet, wenn viele Dokumente mit der gleichen, relativ einfachen Struktur vorliegen.

- Bei einer *halbautomatischen* Wrapper-Erstellung wird diese durch Werkzeuge unterstützt. Benutzer geben beispielsweise an, aus welchem Teil eines Dokumentes Daten extrahiert werden sollen, die Werkzeuge liefern dann Beispiele zur Umsetzung. Damit lässt sich die Erforderlichkeit von Expertenwissen in einem gewissen Maß umgehen.

- Die *(voll-)automatische* Wrapper-Erstellung ist lange Gegenstand der Forschung, Techniken des Maschinenlernens wurden entwickelt. Dabei benötigen diese Systeme meist dennoch ein Minimum an Eingriffen der Nutzer, eine beginnende Trainingsphase anhand von repräsentativen Beispielen oder manuell erstellte Regeln werden eingesetzt.

Da in dieser Arbeit auf Quellen zurückgegriffen wird, die über eine einheitliche Gestaltung viele Dokumente vorhalten, die relativ geringen Aufwand in der Wrapper-Erstellung benötigt, wird hier der manuelle Ansatz gewählt. Dies ermöglicht unmittelbare Erfolge, um Daten für die Informationsextraktion bezüglich der Autorzu-

gehörigkeiten zu erhalten. Es gilt im Weiteren, Verfahren aufzuzeigen, wie sich aus den Abstract-Seiten im HTML-Format die notwendigen Daten herausfiltern lassen.

Semistrukturierte Dokumente liefern nicht nur Inhalte, sondern gleichzeitig Angaben zu deren Aufbau (Tags). Die Tags lassen sich hier nutzen, um gewünschte Daten zu erhalten, wobei auch deren Reihenfolge zu beachten ist. So lässt sich allein anhand von `<td class="small-text">...</td>` nicht erkennen, welchen Inhalt die Auslassungspunkte darstellen. Betrachtet man hingegen `<td class="small-text"> <strong>...</strong></td>`, handelt es sich wahrscheinlich um eine Hervorhebung, eine Art Titel wie `Full Text`.

Um solche Teile allerdings aufzufinden, muss das Dokument durchsucht werden. Hierzu bietet sich in erster Form das sequentielle Lesen aller Zeichen, der Aufbau von Zeichenketten und damit der Vergleich mit Identifikatoren an. Diese Methode ist allerdings sehr aufwändig in der Umsetzung. In der Informatik werden für solche Aufgaben seit langem so genannte *reguläre Ausdrücke* genutzt, die ein effizientes Mittel für die Textbearbeitung darstellen [Fri98]. Mittels Metazeichen lassen sich Textmuster, wie oben angeführt, beschreiben. Die jeweilige Umgebung, in der reguläre Ausdrücke eingesetzt werden, stellt dabei die Umsetzung des Auffindens des gesuchten Textes sicher. Mit den Mitteln der regulären Ausdrücke muss die Funktionalität einer speziellen Suche nicht immer wieder neu umgesetzt werden, der Implementierungsaufwand verringert sich dadurch.
Eine Stärke in der Verwendung von regulären Ausdrücken liegt zum Beispiel in der Flexibilität. So lässt sich mit so genannten *Quantifiern* die Anzahl an auftretbaren Zeichen angeben. Damit kann man Variationen in Dokumenten, beispielsweise in der Anzahl an Leerzeichen oder Zahlen begegnen.
Reguläre Ausdrücke haben sich mittlerweile etabliert, werden von vielen Programmiersprachen wie Perl, Python [Fri98] und Java unterstützt, finden aber auch in den Suchfunktionen von Büroanwendungen wie *OpenOffice*[17] Verwendung.

[17] www.openoffice.org

3.4 Wrapper für XML-Dokumente

Während HTML als Beschreibungssprache für Web-Seiten entwickelt worden ist, wurden mit dem Fortschritt des Web andere Konzepte notwendig, um einen Austausch von Dokumenten zu gewährleisten. Hierfür hat sich die *Extensible Markup Language (XML)* durchgesetzt, die die Daten in einer selbstdefinierbaren hierarchischen Struktur vorhält. XML-Dokumente halten selbstbeschreibende Daten bereit, die damit automatisch und unkompliziert weiterverarbeitbar sind [BRS03].
Doch mit dieser Definition eines semistrukturierten Formats werden Parallelen zu HTML ersichtlich. Weiterhin gemeinsam ist beiden Formaten der Einsatz von Tags, als auch, dass ebenso das Schema eines XML-Dokuments mittels einer DTD beschrieben werden kann. Dies ist dadurch bedingt, dass beide mit SGML die selbe Wurzel besitzen:

> *„XML entstand aus den Erfahrungen mit der Structured Generalized Markup Language (SGML) und der ebenfalls zur Familie der Auszeichnungssprachen (markup languages) gehörenden HyperText Markup Language (HTML). Während sich mit SGML ein sehr geradliniger und wohlstrukturierter, jedoch komplexer und damit schwierig zu handhabender Standard entwickelt hat, weist HTML deutliche Schwächen auf, was die Trennung unterschiedlicher Dokumenteninhalte angeht [...]. Außerdem ist HTML darstellungsorientiert und unterstützt damit die Trennung zwischen Dokumenteninhalten und deren Präsentation ungenügend bis gar nicht.“* [RV03]

XML legt für den Inhalt restriktivere Methoden vor, so muss zum Beispiel jedes Element neben einem Start- auch ein Ende-Tag aufweisen (`<Start-Tag>Inhalt </Ende-Tag>`). Ein detaillierterer Überblick über Aufbau von XML-Dokumenten, Restriktionen und Ziele von und Einsatzmöglichkeiten mit XML gibt beispielsweise [RV03].

Die Ähnlichkeiten der beiden Datenmodelle lassen den Schluss zu, dass für XML ebenso die Extraktionsmethoden wie in Kapitel 3.3 für HTML in Frage kommen. Doch existieren für diese Aufgabe spezialisierte Verfahren, wie DOM und SAX.

Das *Document Object Model (DOM)* wie auch *Simple API for XML (SAX)* spezifizieren, wie auf rechnerinterne Darstellungen von XML-Dokumenten zugegriffen werden kann. Eine jeweilige Implementierung dieser Definitionen stellt in Form einer *API (Application Programming Interface; Programmierschnittstelle)* Funktionen bereit, durch die Anwendungen den jeweiligen Zugriff durchführen können [RV03].
DOM wurde vom internationalen World Wide Web Consortium (W3C)[18] definiert, welches sich mit der Erstellung von Standardisierungen und Leitfäden für Techniken des Web beschäftigt. Mittels DOM wird die Struktur eines XML-Dokumentes rechnerintern als Baum repräsentiert. Ein Wurzelknoten enthält Unterknoten entsprechend den Unterelementen des XML-Dokumentes, die wiederum auch Unterknoten aufweisen können. Die Bezeichnung *Document Object Model* lässt darauf schließen, dass das XML-Dokument aus einer objektorientierten Sicht repräsentiert wird, Knoten sind demnach die Objekte. Diese umfassen weiterhin Attribute, die sich über die jeweilige API mit einer bestimmten Funktion oder Methode abfragen lassen.
Im Kontrast dazu wurde SAX entwickelt. Es handelt sich hierbei jedoch nicht um eine Spezifikation eines Konsortium oder Ähnlichen und stellt damit höchstens einen de-facto-Standard dar. Die Intention von SAX ist, ein XML-Dokument nicht komplett rechnerintern als Baum zu repräsentieren, sondern dieses sequentiell lesen zu können, um beim Auftreten eines Knotenobjekts die Anwendung über dieses Ereignis direkt zu informieren. Damit sollen eventuelle Speicherprobleme umgangen werden, denn DOM hält den gesamten Dokumenten-Baum speicherintern vor, was bei großen XML-Dokumenten zur Speicherauslastung führen kann. Mittels SAX werden nur wenige Teile des Dokumentes temporär im Speicher gehalten und nach der Bearbeitung wird dieser wieder frei gegeben.
Das bedeutet, dass mittels DOM ein Zugriff auf jedes Objekt jederzeit möglich ist, mit SAX nur zuletzt aufgetretene Ereignisse und deren Daten behandelt werden können.

3.5 Wrapper für PDF-Dokumente

Die erste Spezifikation für das *Portable Document Format (PDF)* wurde mit den Adobe Acrobat Produkten 1993 veröffentlicht. Das Dateiformat basiert auf der Seiten-

[18] www.w3.org

beschreibungssprache PostScript und stellt heute einen de-facto-Standard im Rahmen des Dokumentenaustauschs dar. Zudem wurde die aktuelle Version *PDF 1.7* von der *International Organization for Standardization (ISO)* in einer ersten Abstimmung zertifiziert und ist damit zum allgemein gültigen Standard *ISO 32000*[19] in einem Entwurfsstadium geworden.[20].

> *„PDF builds on the PostScript page description language by layering a document structure and interactive navigation features on PostScript's underlying imaging model, providing a convenient, efficient mechanism enabling documents to be reliably viewed and printed anywhere.“* [pdf06]

PDF stellt demnach ein plattformunabhängiges Dateiformat dar, welches beispielsweise jeweils eingesetzte Schriftarten mitführt, um sie jederzeit darstellen zu können. Ein ausführlicher Überblick über die aktuelle Spezifikationsversion 1.7 wird von Adobe mit [pdf06] bereit gestellt.

Seit der PDF-Version 1.4 werden in der Spezifikation standardisierte Strukturtypen und Attribute definiert, auf deren Basis sich Seiteninhalt (Text, Grafiken) extrahieren lässt. Dies ist zum Beispiel dafür gedacht, um das PDF-Dokument in ein anderes Format wie HTML oder XML zu konvertieren,

> *„with document structure and basic styling information preserved“* [pdf06].

Diese Art der Konvertierung wird beispielsweise im Rahmen von Google Scholar durchgeführt (siehe Kapitel 2.5). Anwendungen oder APIs, die diese Aufgaben übernehmen, lassen sich zum Beispiel auf der Softwareentwicklungsseite von SourceForge, Inc.[21] finden. Mit diesen lassen sich somit HTML- oder XML-Dokumente erstellen, für die folglich die in den Kapiteln 3.3 als auch 3.4 dargelegten Extraktionsmöglichkeiten in Frage kommen.
Mit dieser Methode wird beispielsweise in [BPZ01] gearbeitet, um die Referenzen

[19] www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=45873, „ISO/DIS 32000“, 7.12.2007

[20] www.heise.de/newsticker/meldung/100085, „Adobes PDF besteht wichtige ISO-Abstimmung“, 07.12.2007

[21] sourceforge.net

aus wissenschaftlichen Arbeiten zu beziehen. Dort ist es theoretisch egal, ob eine Konvertierung nach HTML stattfindet. Das PDF-Dokument enthält keine Angaben darüber, an welcher Stelle sich eine Referenz(liste) befindet, somit kann ein entstehendes HTML-Dokument ebenso keine solchen Informationen beinhalten. Nur durch eine genaue Untersuchung des Inhalts und dessen Formatierung lassen sich Rückschlüsse diesbezüglich ziehen. In [BPZ01] wurde eine ebenso aufwändige Konvertierung von PDF zu HTML durchgeführt, da schon ein Programm zur Auswertung von HTML-Dokumenten bestand.
Dem Zitat oben zufolge wird versucht, bei einer Konvertierung hauptsächlich die Struktur zu erhalten und dem entstehenden Dokument in zweiter Linie elementare Formatierungen zu übertragen.

Die unzureichende Unterstützung seitens des PDF bezüglich der Inhaltsstrukturierung stellt einen weiteren Nachteil in der Verwendung dieses Formates als Datenquelle dar. Hierbei müsste zuerst ein aufwändiges Auffinden der in dieser Arbeit benötigten Daten stattfinden.
Bei der Arbeit [HGM+03] handelt es sich um ein Beispiel, bei dem aus wissenschaftlichen Arbeiten Metadaten wie Autoren und Zugehörigkeiten extrahiert wurden. Dafür musste ein anspruchsvolles Verfahren mittels Methoden des maschinellen Lernens entwickelt werden.

3.6 Umsetzungsergebnisse

Details zur Umsetzung der Implementierung werden im Kapitel 5 veranschaulicht. In diesem Kapitel sollen im Vorfeld Resultate dieser Umsetzung bezüglich der Datenextraktion mittels der voran besprochenen Wrapper-Technologien aufgelistet werden. Die nachfolgende Tabelle 3.6 enthält dabei Beispieldaten (Zeichenketten), die für eine Informationsextraktion bezüglich der Autorzugehörigkeiten von Interesse sind und in Folge dessen als Musterbeispiele dienen.

1	Aachen
2	Applications Divisions
3	C&C Systems Research Laboratories, NEC Corporation, Miyamae, Kawasaki, Kanagawa 213, Japan and Computer Science Research Department, Lawrence Berkeley Laboratory, University of California

4	Concordia U.
5	Concordia Univ.
6	Concordia University
7	Depto. de Informática, Pontlfícia Universidade, Catollca, Rio de Janeiro, RJ, BRASIL
8	INRIA Rocquencourt
9	INRIA Rocquencourt, projet RODIN, BP 105, F-78153 Le Chesnay Cedex, France
10	Institut füur Angewandte Informatik und Formale Beschreibungsverfahren University of Karlsruhe, Fed. Rep. of Germany
11	M.I.T.
12	School of Environmental Science
13	Univ. Karlsruhe, Karlsruhe, West Germany
14	University of Leipzig, Leipzig, Germany
15	York University
16	York University, Toronto, ON, Canada
17	York University, 4700 Keele Street, Toronto, Ontario M3J 1P3, Canada

Tabelle 6: Repräsentative Ergebnisbeispiele der umgesetzten Datenextraktion

Die aufgeführten, repräsentativen Beispiele lassen eine starke Heterogenität der Daten erkennen, die sich in folgenden Punkten zeigen kann:

- *Eigennamen* (2), auch in abgekürzter Form (11), lassen oftmals kaum automatische Rückschlüsse auf Ort oder Einrichtungstyp zu.
- Auch wenn keine Eigennamen angegeben sind, so werden gelegentlich *keine Orts-/Institutsangaben* (1, 12) vorgehalten.
- Beispielsweise aufgrund von OCR-Fehlern (siehe Kapitel 2) oder Unachtsamkeit treten *Rechtschreibfehler* (10) auf.
- Oft unterscheiden sich Angaben in ihrem *Detailgrad*, obwohl zur selben Einrichtung gehörend (8-9, 15-16-17).
- Stellenweise beschreiben *verschiedene Darstellungen* den selben Sachverhalt (4-5-6).
- Ein Teil der Daten enthält *mehrere Angaben* (3) zu einem Autoren.
- Unterschiede bestehen weiterhin in der jeweils genutzten *Sprache* (7-14).
- Ältere Arbeiten enthalten teilweise *veraltete Angaben* (13).

Auf der Basis mehrerer Tausend solcher Zugehörigkeitskennzeichnungen aus den durchgeführten Extraktionsprozessen wurden die im Weiteren erläuterten Informationsextraktionsmöglichkeiten entwickelt.

4 Informationsextraktion

In diesem Kapitel werden verschiedene Verfahren zur Extraktion der Orts- und Institutsinformationen erörtert. Diese Informationen sollen aus den Zeichenketten entnommen werden, die unter anderem Gegenstand der Datenextraktion des vorigen Kapitels 3 waren und die Autorzugehörigkeiten angeben. In diesem Zusammenhang wird auf Möglichkeiten der Umsetzung eingegangen und diese bewertet.

4.1 Verzeichnisse

Eine Liste stellt eine Aufzählung von Ausdrücken dar, die thematisch miteinander in Beziehung stehen. Im Rahmen der Informatik besteht weiterhin die Definition einer Liste als Datenstruktur, auf die sich hier nicht bezogen wird. Vielmehr ist von Listen die Rede, die manuell erstellt werden und bestimmte Begriffe enthalten.
Solche Begriffslisten können in unterschiedlichen Hinsichten eingesetzt werden. So finden heutzutage so genannte *Black-* und *Whitelists* in verschiedenen Bereichen Anwendung.

4.1.1 Black-/Whitelists

Der englische Begriff *Blacklist* lässt sich ins Deutsche mit *Negativliste* oder *schwarze Liste* übersetzen. Diese negativen Ausdrücke lassen darauf schließen, dass der Inhalt der Listen im Rahmen einer Anwendungsdomäne nicht wünschenswert ist.
So setzt man solche Listen zum Beispiel im Rahmen der SPAM-Erkennung bezüglich E-Mails ein. SPAM bezeichnet unerwünschte E-Mails, deren Inhalte und/oder Absender in schwarzen Listen zum Vergleich gespeichert werden, um eingehende Nachrichten damit zu vergleichen und gegebenenfalls abzuweisen.

Im Gegensatz dazu kann man *Whitelist* mit *Positivliste* oder analog als *weiße Liste* bezeichnen. Deren Inhalt deklariert etwas Erwünschtes. Positivlisten können ebenso in der SPAM-Erkennung eingesetzt werden, indem man beispielsweise nur erwünschte Absenderadressen darin vorhält.

In dieser Arbeit dienen Negativlisten zum Beispiel dazu, um darin häufig auftretende Begriffe aufzuzählen, die keine Ortsinformationen beinhalten. Für das Beispiel `University of Leipzig, Leipzig, Germany` (Nr. 14, Tabelle 3.6, Seite 43) wären dies `University` und `of`, übrig bliebe `Leipzig, Leipzig, Germany`. Die doppelte Nennung des Ortes lässt sich mittels geeignetem Algorithmus entfernen, wodurch man `Leipzig, Germany` als Ergebnis erhält.
Auf den ersten Blick lassen sich so gute Ergebnisse erzielen, doch ist dies längst nicht auf alle Daten anwendbar. Bei dem Beispiel `School of Environmental Science` (12) würden eventuell alle Wörter entfernt, bei `M.I.T.` (11) dagegen nichts. Beide Male blieben keine verwertbaren Ortsinformationen übrig.

Mit Positivlisten lassen sich etwa Ausdrücke auflisten, die Institutstypen darstellen. Mit `University` ließe sich dieser Typ finden. Allerdings besteht hier einerseits die Problematik der verschiedenen Darstellungsformen (`U.`, `Univ.`, `University`), die alle aufgelistet werden müssen. Des Weiteren wird so noch keine Aussage über den Typ getroffen, also ein Metadatum darüber extrahiert. Um die Darstellungen einem Einrichtungstyp zuordnen zu können, bieten sich die im anschließenden Kapitel 4.1.2 besprochenen Tabellarischen Verzeichnisse an.

Nachteile der hier diskutierten Listentypen bestehen darin, dass die Listen für eine hohe Anzahl an Daten aufgestellt werden müssen. Aufgrund der in Kapitel 3.6 aufgezeigten Heterogenität stellt sich eine manuelle Listenerstellung als aufwändig dar. Aber auch der Aufbau einer automatischen Erstellung durch maschinelles Lernen solcher Listen erfordert hohen Umsetzungsaufwand.
Um beispielsweise Problemen mit Rechtschreibfehlern, Groß- oder Kleinschreibung einzelner Begriffe beziehungsweise Ein- oder Mehrzahl derer zu begegnen, bietet sich auch bei einer automatischen oder manuellen Listenerstellung eine Verwendung regulärer Ausdrücke an. Zum Beispiel mittels `(d|D)atabase(s)?` würden die Begriffe `database`, `Database`, `databases` als auch `Databases` gefunden werden können.

4.1.2 Tabellarische Verzeichnisse

Tabellarische Verzeichnisse stellen eine erweiterte Liste in Form einer Tabelle dar. Hier werden nicht nur bestimmte Begriffe aufgezählt, sondern darüber hinaus zu

jedem Ausdruck Metadaten vorgehalten, die einen Ausdruck genauer spezifizieren beziehungsweise Zusatzinformationen liefern.

Im Rahmen der Extraktion von Einrichtungstypen lässt sich demnach ein Verzeichnis aufstellen, wie es in Tabelle 7 exemplarisch aufgezeigt ist. Hiermit wird eine

Typ	**Ausdruck**
academy	U.
academy	Univ.
academy	University
academy	Universität
academy	Universidade
academy	Politecnico
academy	Berufsakademie
association	Alliance
association	Partnership

Tabelle 7: Beispiele für Institutstypzuordnungen

Zuweisung verschiedener Ausdrücke zu einem bekannten Typ und damit die Aussage möglich, dass zum Beispiel `Univ.` vom selben Typ ist wie `Universität`, aber sich von `Alliance` unterscheidet.
Verschiedene Erweiterungen sind vorstellbar. So kann auch hier mit regulären Ausdrücken gearbeitet werden, eine Zuordnung eines Ausdrucks zu mehreren Typen ist möglich. Ein solches Verzeichnis kann insofern weiter ausgebaut werden, dass über (nummerische) Werte eine Hierarchie gebildet werden kann. Aufgrund derer werden Aussagen möglich, dass zum Beispiel eine `association` hierarchisch über einer `academy` steht.
Dies zeigt gleichermaßen, dass zunächst eine Konvention für den Aufbau gefunden werden muss und dies anschließend eine aufwändigere Umsetzung bezüglich Positivlisten erfordert.

Eine Erstellung solcher Verzeichnisse, um Orte zu identifizieren, stellt eine noch weit anspruchsvollere Aufgabe dar, wenn möglichst alle Orte der Welt berücksichtigt werden müssen. Doch dieser Aufgabe wurde sich schon umfassend gewidmet, da solche Daten heutzutage in vielen Bereichen Verwendung finden. Beispielsweise im Rahmen *geografischer Informationssysteme (GIS)* werden mit raumbezogenen Daten verschiedenste Anwendungen entwickelt, etwa um Einzugsbereiche von Un-

ternehmen zu analysieren und darzustellen.
In der heutigen Zeit des so genannten Web 2.0, in der Nutzer eigenen Inhalt auf vielen Web-Seiten veröffentlichen können, werden raumbezogene Daten ebenso benötigt. So erlauben es manche Seiten, Texte oder Bilder mit Adressen beziehungsweise geografischen Koordinaten zu versehen, um sie dadurch auf einer Landkarte anzeigen zu lassen.

So existieren mittlerweile diverse Anbieter geografischer Verzeichnisse, oftmals auch kostenfrei.
Die *National Geospatial-Intelligence Agency (NGA)* der US-amerikanischen Armee stellt auf ihrer Internetseite Dateien mit geografischen Informationen zu 251 Ländern beziehungsweise geografischen Gebieten[22] auch für zivile Nutzung bereit. Diese umfassen unter anderem die vollständige Bezeichnung, Höhen- und Breitenangaben als auch verschiedene Codes, die etwa Typ und Sprachen kennzeichnen. Die Daten können somit in ein GIS oder eine Datenbank geladen werden, um darauf basierend Analysen durchzuführen.
Weiterhin wurde eine Web-Seite mit dem Titel *GeoNames* [Wic07] aufgebaut, die vergleichbare Daten kostenfrei anbietet. Diese werden hingegen von der Allgemeinheit aufgestellt und gegebenenfalls korrigiert, im Gegensatz zum Angebot der NGA. Außerdem werden hier mehr Informationen bereitgestellt, wie zum Beispiel alternative Bezeichnungen und Einwohneranzahlen.
Einen Überblick über weitere Anbieter (inter-)nationaler, geografischer Informationsverzeichnisse bietet beispielsweise die *Arizona State University*[23].

Auf der Basis von derartigen Ortsverzeichnissen ist demnach ein Abgleich mit zu untersuchenden Zeichenketten möglich. Aus dem Beispiel `University of Leipzig, Leipzig, Germany` könnten demnach `Leipzig` und `Germany` aufgrund der Eindeutigkeit als geografische Elemente und weitere Informationen dazu extrahiert werden. Auf dieser Grundlage muss weiterhin eine Unterscheidung hinsichtlich des Detailgrads der Angaben möglich sein, um in diesem Fall zu erkennen, dass `Leipzig` eine Stadt und im Staat `Germany` inbegriffen ist.
Allerdings wird aus einer Zeichenkette wie zum Beispiel `Brandenburg` nicht ersichtlich, ob es sich um das deutsche Bundesland oder die Stadt handelt. Weiterhin

[22] „Country Files“, earth-info.nga.mil/gns/html/cntry_files.html, 02.12.2007
[23] „Place Name Sites“, www.asu.edu/lib/hayden/govdocs/maps/geogname.htm, 02.12.2007

existieren einige mehrdeutige Namen, so etwa `Amerika` als Staatsname und auch als Name einer deutschen Stadt, auch `Berlin` existiert allein in Deutschland mehrfach als Stadtname.
Vorausgesetzt wird weiterhin, dass es sich bei jedem einzelnen Wort um einen eigenen Identifikator handelt. Der Stadtname `Rio de Janeiro` hingegen besteht aus mehreren Wörtern, die zusammen einen Identifikator bilden. Hier die gesamte Zeichenkette zwischen zwei Begrenzern (hier zumeist Kommata) als zusammengehörenden Identifikator zu nutzen, stellt allerdings keine zufriedenstellende Lösung dar. Wenn die ebenso durch Kommata begrenzte Zeichenkette `Concordia University` nur zusammen betrachtet werden würde, können keine entsprechenden geografischen Informationen den Verzeichnissen entnommen werden.
Demzufolge bietet sich die gemischte Verwendung mit den voran besprochenen Negativlisten an, wodurch `University` gelöscht wird beziehungsweise nicht weiter betrachtet werden muss. Sind wie bei `Rio de Janeiro` nach einem Negativlistenvergleich noch mehrere unbekannte Wörter übrig, muss eine sinnvolle Strategie gefunden werden. Hier bietet sich beispielsweise an, alle Möglichkeiten zu untersuchen und am Schluss die Ergebnisse miteinander zu vergleichen. Möglichkeiten sind dabei die gesamte Zeichenkette als auch alle einzelnen Teilzeichenketten, in vorliegender Reihenfolge anhand der Leerzeichen getrennt (`Rio de`, etc.).
Hieran wird ersichtlich, dass das Erstellen eines Systems zur automatischen Extraktion geografischer Informationen aus einer unbekannten Zeichenkette beliebig komplex und damit aufwändig sein kann.
Darüber hinaus kommt hinzu, dass ein Extraktionssystem mit Rechtschreibfehlern umgehen können, sich dadurch zumindest nicht stark negativ beeinflussen lassen sollte. Weiterhin stellen die verschiedenen Sprachen in den hier erhaltenen Daten eine große Problematik dar. GeoNames hält für einige geografische Gebiete in den Alternativen die Bezeichnungen auch in anderen Sprachen, vor allem im Original bereit, wohingegen die primären Nutzdaten oftmals auf Englisch vorgehalten werden. Eine Möglichkeit in der Bearbeitung dieser Problemstellung wird im nächsten Kapitel 4.1.3 betrachtet.

Ein weiterer Nachteil in der Verwendung solcher Ortsverzeichnisse stellt das lokale Speichern dieser Daten in Datenbanken oder Ähnlichem dar. Dadurch wird einerseits Speicherplatz benötigt. Dieser Faktor besitzt aufgrund heutiger Festplattenka-

pazitäten geringere Bedeutung. Allerdings kommt hinzu, dass mit fortschreitender lokaler Speicherdauer die Aktualität der Daten bezüglich der Originale aus dem Web sinkt. Angemessene Funktionen zur Aktualisierung des Datenbestandes müssten gefunden werden, wodurch der Aufwand wiederum steigt. Geeigneter ist hier die Nutzung von Originaldaten, die entsprechend vom Anbieter bezogen werden müssten. Hierfür werden im Kapitel 4.2 Möglichkeiten aufgezeigt.

4.1.3 Apache Lucene

Um verschiedenen Problematiken bezüglich der Datenheterogenität begegnen zu können, bietet sich die API *Apache Lucene* an.

> *„Apache Lucene is a high-performance, full-featured text search engine library written entirely in Java. It is a technology suitable for nearly any application that requires full-text search, especially cross-platform.“* [apa07]

Die Suchmaschine Apache Lucene (im Weiteren als Lucene bezeichnet) erstellt aus Dokumenten einen Index, um auf dieser Basis Suchfunktionalitäten umsetzen zu können. Ein Dokument kann hier ebenso eine kurze Zeichenkette sein, der Begriff *Dokument* besitzt hier eine andere Bedeutung als auf Betriebssystemebene. Mittels Lucene werden Feldnamen Feldwerte zugeordnet, wobei sich beispielsweise angeben lässt, ob der Wert (hier: Zeichenkette) in seine Bestandteile (Wörter) zerlegt (die so genannte *Tokenisierung*) und diese dadurch einzeln indexiert werden sollen. Häufig auftretende Wörter wie zum Beispiel Artikel werden dabei standardmäßig nicht in den Index aufgenommen, da dies dem Ziel eines Index widersprechen würde. Weiterhin kann angegeben werden, welche Felder indexiert und ob deren Inhalt unverändert mit gespeichert werden sollen. Somit lassen sich zusätzliche Werte speichern, die nicht Inhalt einer Suche sind, sondern im Ergebnis mit erhalten werden.
Über einen erstellten Index kann im Falle einer Suche schnell auf benötigte Dokumente geschlossen werden. Eine Suche kann beispielsweise unter anderem unscharf vorgenommen werden. Das bedeutet, dem Suchbegriff wird in einer bestimmten Syntax ein Wert hinzugefügt, der die Mindestähnlichkeit von Suchbegriff und Ergebnis angibt. Diese basiert auf der *Levenshtein Distance* [Lev65] LD. Mit der Levenshtein Distance wird die kleinste Anzahl an Operationen ermittelt, um eine Zeichenkette

in eine andere zu überführen. Mögliche Operationen sind dabei *Einfügen*, *Löschen* und *Ersetzen* eines Zeichens. Im Rahmen von Lucene findet eine Normalisierung der Ähnlichkeit statt, damit diese einen Wert zwischen 0.0 und 1.0 besitzt. Bei zwei zu vergleichenden Zeichenketten $ZK1$ und $ZK2$ berechnet sich diese Ähnlichkeit sim mit

$$sim = \frac{MIN(LENGTH(ZK1), LENGTH(ZK2)) - LD}{MIN(LENGTH(ZK1), LENGTH(ZK2))} \tag{1}$$

Mit Lucene ließen sich demnach in dieser Arbeit die angesprochenen Ortsverzeichnisse indexieren, wobei jeweils ein Ort einschließlich aller nötigen Attribute (Name, Koordinaten, etc.) einem Dokument in Lucene entspräche. Eine zu untersuchende Zeichenkette ließe sich zerlegen und einzelne Wörter beziehungsweise Wortgruppen mit den durch Lucene indexierten Orten vergleichen. Soll beispielsweise `Leverksuen` untersucht werden, ergäbe eine scharfe Suche (maximale Ähnlichkeit) womöglich kein Resultat, da dieser Name auf keine Ortsbezeichnung passt. Mit einer unscharfen Suche und Mindestähnlichkeit von zum Beispiel 0.8 ließe sich die Stadt `Leverkusen` finden, denn

$$sim^{Leverksuen,Leverkusen} = \frac{MIN(10,10) - 2}{MIN(10,10)} = \frac{8}{10} = 0.8 \tag{2}$$

Hierbei treten allerdings einige Problemstellungen auf. So ist es notwendig, einen sinnvollen Schwellwert zu ermitteln. In diesem Zusammenhang spielt die Länge der Zeichenketten ein Rolle. Sobald in `Austin` ein Buchstabe falsch geschrieben wurde, verringert sich der Ähnlichkeitswert stärker als in einer längeren Bezeichnung wie `Springfield`.

$$sim^{Austln,Austin} = \frac{6-1}{6} = 0.8\overline{3} \tag{3}$$

$$sim^{Springfleld,Springfield} = \frac{11-1}{11} = 0.\overline{90} \tag{4}$$

Wird also die Grenze für die Mindestähnlichkeit zu niedrig angesetzt, werden bei langen Bezeichnungen zu viele Ergebnisse zurückgeliefert, was die Auswertung erschwert.

Hier wäre es möglich, eine absteigende Suche durchzuführen. Wenn der erste Schritt einer scharfen Suche keine Ergebnisse lieferte, wird zum Beispiel eine Mindestähnlichkeit um 0.1 reduziert. Liefert diese Suche kein Ergebnis, wird dementsprechend fortgefahren, bis zu einer definierten minimalen Mindestähnlichkeit oder ersten Ergebnissen. Die minimale Mindestähnlichkeit sollte dabei abhängig von der Länge der zu vergleichenden Zeichenketten sein.

Dieser Ansatz erhöht allerdings die durchschnittliche Anzahl an Suchanfragen und damit die Ausführungszeit. Diese Erhöhung hängt von den zu untersuchenden Inhalten und der Schrittweite der Ähnlichkeitsreduzierung ab.
Weiterhin lässt eine Unterscheidung in einem Zeichen bei kurzen Wörtern nicht unbedingt auf einen Rechtschreibfehler oder Ähnliches schließen. Ein Beispiel dafür stellen die Orte `Austin` und `Mustin` dar.

Mittels der Tokenisierung kann bei der Indexerstellung eine Zeichenkette in seine Teile, die Token zerlegt werden. Somit lässt sich nach den einzelnen Token suchen. Diese Eigenschaft kann auf der einen Seite sinnvoll sein. Bei einer zerlegten und indexierten Zeichenkette `Stadt Brandenburg` kann man mit dem Suchwort `Brandenburg` auf die Stadt schließen. Andererseits stellt dies in anderen Fällen einen Nachteil dar. Bei einer Suche nach `Santa` ist eine Unterscheidung zwischen den Beispielen `Santa Barbara` und `Santa Clara` nur unter zusätzlichem Aufwand möglich. Der Rest der Zeichenkette muss dabei ebenfalls untersucht und die Ergebnisse miteinander verglichen werden.

4.1.4 Zusammenfassung

Das Kapitel 4.1 zeigte verschiedene Formen von Verzeichnissen, deren Einsatzmöglichkeiten als auch Vor- und Nachteile auf. Verzeichnisse können auf unterschiedliche Arten genutzt werden, um einen Algorithmus zur Analyse von Zeichenketten für eine Extraktion von Zugehörigkeitsinformationen (Institute, Orte) zu entwickeln.
Dabei stellen sich vielfältige Problemstellungen heraus, wodurch ein nahezu perfekter Algorithmus aufwändig zu realisieren ist. Dahingehend soll im Weiteren untersucht werden, ob sich Dienste oder Vergleichbares finden lassen, die diese Aufgabe übernehmen können.

4.2 Dienste des Web

In Kapitel 4.1.2 wurde bereits besprochen, dass es möglich ist, verschiedene digitale Medien (Texte, Bilder, etc.) mit Koordinaten auszustatten. Diese so genannte Geokodierung (auch Geocoding oder Geotagging genannt) kann auf verschiedene Arten

erfolgen.
So bietet beispielsweise die Firma *Internetservice WebConsultant.de* mit Geokodierung.net[24] einen kostenfreien Dienst im Web an, bei dem sich Adressdaten (Land, PLZ, Ort, Str.) eingeben und dadurch verschiedene Informationen (Ländercode, Bundesland, Region) und die eindeutigen Koordinaten (Höhen-, Breitengrad) ermitteln lassen, die sich auf das Koordinatensystem WGS84 beziehen, dessen Definition in [Nat00] nachgelesen werden kann.
Für eine automatische Nutzung ist dieser Dienst ungeeignet, da dieser einerseits angesichts der Web-Seite für manuelle Inanspruchnahme gedacht ist. Die Erstellung eines Wrappers zum Ansprechen der Formularfelder und Auswerten der Ergebnisseite ist aufwändig und die Wrapper nicht gegenüber Darstellungsänderung robust. Andererseits ist eine maschinelle Verwendung offensichtlich nicht erwünscht, da hiergegen Sicherheitsmaßnahmen eingerichtet worden. Aus einem Bild muss der Anwender einen Sicherheits-Code entnehmen und in einem Formularfeld eintragen. Eine maschinelle Umsetzung würde hier komplexe Bildverarbeitungsalgorithmen bedingen.
Weiterhin lassen sich bei diesem Dienst Adressdaten über eine interaktiv benutzbare Landkarte ermitteln, ohne hierfür Daten vorher einzugeben. Zu diesem Zweck wurde eine Programmierschnittstelle namens *Google-Maps-API* angewandt, die im Kapitel 4.2.3 näher erläutert wird.

Aufgrund der Nachteile solcher nutzerzentrierten Dienste wurden Verfahren entwickelt, die eine maschinelle Kommunikation zwischen im Web verteilten Anwendungen ermöglichen, bestimmte Dienste anbieten, die so genannten *Web Services*.

4.2.1 Web Services

Das W3C definiert einen Web Service als

> *„[...] a software system designed to support interoperable machine-to-machine interaction over a network.“* [BHM+04]

[24] www.geokodierung.net

Die Interoperabilität zwischen den Maschinen wird einerseits mittels maschinenlesbarer Spezifikation der Schnittstelle zum Web Service ermöglicht. Hierfür wurde die *Web Services Description Language (WSDL)* entwickelt, die aufrufbare Methoden und deren Parameter beschreibt. Mittels *SOAP* werden XML-Dokumente als eine Art selbstbeschreibende Nachricht deklariert, über die Daten zwischen den Maschinen übertragen werden. Die Übertragung dieser Nachrichten erfolgt mittels Internetstandards, wie zum Beispiel dem *Hypertext Transfer Protocol (HTTP)*. Bei einem Verzeichnisdienst wie *Universal Description, Discovery and Integration (UDDI)* können solche Web Services registriert und anschließend aufgefunden werden [BRS03].
Web Services sind im Endeffekt Softwaresysteme, deren Implementierungssprache als auch -struktur nach außen hin irrelevant sind. Bedeutend sind primär die Methoden, die ein Web Service anbietet und welche Aufgaben damit gelöst werden können. Web Services können somit unabhängig voneinander umgesetzt werden, in einer Anwendungskomposition auch zusammen arbeiten und damit abhängig voneinander sein.

Die angesprochenen, grundlegenden Technologien werden häufig im Zusammenhang mit Web Services genutzt beziehungsweise in der Literatur genannt. Legt man allerdings einzig die genannte Definition des W3C zugrunde, lassen sich die im Weiteren angesprochenen Dienste ebenfalls als Web Services betrachten.

4.2.2 Geonames-Service

In Kapitel 4.1.2 wurde bereits das Ortsdatenverzeichnis von Geonames [Wic07] angesprochen. Darüber hinaus bietet die Web-Seite verschiedene Dienste an, die mit *Web Services* betitelt worden sind. Mehr als ein Dutzend Dienste können über eine jeweils spezifische URL aufgerufen werden. Das bedeutet, über eine URL kann nur eine Art der Problemlösung erreicht werden, im Gegenteil zum Verständnis eines Web Service, bei dem im Allgemeinen mehrere Probleme bearbeitet werden können. Die Beschreibung der Dienste von Geonames und dem Aufbau der jeweiligen Schnittstelle erfolgt für Menschen lesbar auf der Web-Seite, nicht durch eine spezielle maschinenverarbeitbare Sprache wie zum Beispiel WSDL.
Bei den angebotenen Diensten handelt es sich beispielsweise um die Auflistung von

Orten, die eine bestimmte Postleitzahl besitzen, oder auch Funktionen der so genannten *inversen Geokodierung*. Hierbei sind Koordinaten bekannt und Informationen, wie zum Beispiel der jeweilige Ländercode, werden hierzu gesucht.
Die möglichen Dateiformate der Ergebnisausgaben sind XML, unstrukturierter Text, das Datenaustauschformat *JavaScript Object Notation (JSON)* als auch *Resource Description Framework (RDF)*, das der Beschreibung von Ressourcen dient.

Der *Geonames Search Webservice*[25] bietet den für diese Arbeit interessantesten Dienst an. Über funktional ähnliche Parameter lässt sich dem Dienst ein Term übergeben, der daraufhin analysiert wird. Dabei kann angegeben werden, ob die Analyse über alle Felder (Orts-, Land-, Kontinentname, etc.) oder nur dem Ortsnamen geschehen soll. Weiterhin kann eine Auswertung anhand einer scharfen oder unscharfen Suche erfolgen. Über weitere Parameter lassen sich die Sprache der Ausgabe ändern oder die maximale Anzahl an dargestellten Ergebnissen anpassen.
Ein Beispiel soll die Funktionsweise veranschaulichen. Über die URL

> http://ws.geonames.org/search?maxRows=5&lang=EN&type=XML
> &style=FULL&fclass=A&fclass=P&q=Cambridge+Massachusetts

erhält man das Ergebnis in Abbildung 6, wobei zwecks Übersichtlichkeit nur das erste Ergebniselement dargestellt wird.

Mit der Anfrage-URL wurde nach dem Term `Cambridge Massachusetts` gesucht. Die weiteren Parameter geben an, dass maximal fünf Elemente ausgegeben werden sollen, die Ausgabe erfolgt auf Englisch mittels XML-Dokument. Es sollen weiterhin alle verfügbaren Informationen geliefert werden. `fclass=A` und `fclass=P` geben an, dass nach einem Landes-, Staats-, Regions- beziehungsweise Stadt-, Dorfnamen oder dergleichen gesucht wird.

Das Ergebnis gibt zunächst an, welche Informationsfülle das Dokument darstellt (`style="FULL"`) als auch wie viele Ergebnisse insgesamt gefunden wurden (`<totalResultsCount>6</totalResultsCount>`). Es schließen sich die Ergebniselemente mit dazugehörigen Informationen an, wie zum Beispiel den Koordinaten, alternativen Bezeichnungen in unterschiedlichen Sprachen und verschiedene Codes.

[25] www.geonames.org/export/geonames-search.html

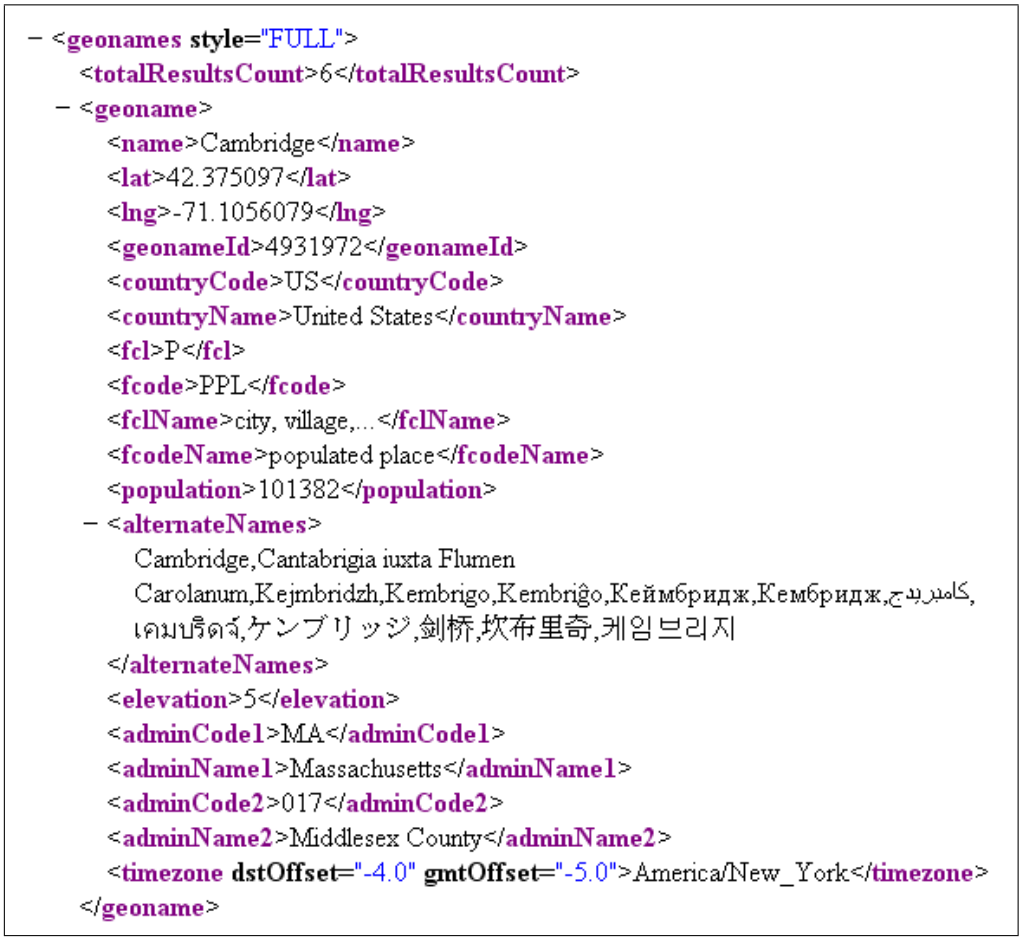


Abbildung 6: Beispielergebnis des Geonames-Search-Webservice (Auszug)

Der Geonames Search Webservice nimmt eine interne Rangordnung der Ergebnisse vor, was bedeutet, dass der Suchanfrage am meisten entsprechende Resultate an oberster Stelle im Dokument ausgegeben werden. Für das obige Beispiel sind die weiteren Ergebnisnamen `Cambridge`, `East Cambridge`, `North Cambridge`, `Old Cambridge` und `Town of Cambridge`. Da diese Resultate nicht jederzeit den Erwartungen entsprechen, muss bei einer Nutzung dieses Dienstes eine sinnvolle Strategie gefunden werden, um die Ergebnisse zu evaluieren. Einerseits ist es möglich, alle Ergebnisse zum gesuchten Term zu speichern. Andererseits könnte nur das erste Element in Betracht gezogen werden.

Da dieser Dienst allein auf geografischen Daten aufbaut, stellt sich hier ein Nachteil heraus, wenn nach einem Term mit nichtgeografischen Wörtern wie zum Beispiel `University of Leipzig` gesucht wird. Dann liefert der Geonames Search Webservice keine Ergebnisse.

Für die Inanspruchnahme der Geonames Dienste kann man fernerhin eine API nutzen. Diese liegt momentan in Version 0.5[26], einzig für die Programmiersprache Java vor. Mit dieser Programmierschnittstelle werden die oben besprochenen Funktionalitäten für Java-Programmierer bequemer zur Verfügung gestellt, bieten darüber hinaus aber keine weiteren Funktionen, beispielsweise für den Umgang mit nichtgeografischen Wörtern.

Sowohl die Dienste über URL-Aufruf als auch die API sind für Anwender und Entwickler unter der *Apache License, Version 2.0*[27] zugänglich. Diese sieht kaum Begrenzungen für die Benutzung vor.

4.2.3 Google-Maps-Service

Der Kartendienst *Google Maps*[28] der Firma Google Inc. ermöglicht das Suchen von Orten weltweit mittels einer Benutzerschnittstelle im Web-Browser. Dieser ist nicht für eine automatische Verarbeitung gedacht. Vielmehr existiert eine Programmierschnittstelle, die Google-Maps-API[29], mittels derer so genannte Mashups erstellt werden können. Dies sind Web-Anwendungen, die hauptsächlich durch verschiedenartigen, importierten und vermischten Inhalt einen neuen Verwendungszweck generieren [RK07]. So lässt sich auch der Kartendienst Google Maps auf Web-Seiten einbinden. Der im Kapitel 4.2 beschriebene Dienst Geokodierung.net nutzt beispielsweise diese Technologie, indem er Anfragen bezüglich der Adress-Auflösung an Google Maps weiterleitet und das Ergebnis im eigenen Format darstellt.
Interessant an diesem Dienst erscheint, dass er zum Teil mit nichtgeografischen Ausdrücken, zum Teil auch mit Eigennamen umgehen kann. So liefert er auf die Anfrage nach der Universität `M.I.T.` die Adresse mit Stadt-, Bundesstaat- und Staatsnamen. Aufgrund des Erfolgs der Firma Google Inc. mit deren Suchmaschine *Google*[30] konnten offensichtlich eine breite Datenbasis angesammelt sowie nützliche Algorithmen zur Analyse eines Suchterms aufgestellt werden.

[26] „GeoNames Source Code“, www.geonames.org/source-code, 09.12.2007
[27] „Apache License, Version 2.0“, www.apache.org/licenses/LICENSE-2.0.html, 09.12.2007
[28] maps.google.com
[29] code.google.com/apis/maps
[30] www.google.com

Obwohl die API eine Verwendung des Dienstes auf anderen Web-Seiten vorsieht, so ist ebenso eine HTTP-Anfrage über eine bestimmte URL und eine Ergebnisverarbeitung unabhängig von einer Web-Seite möglich. Das so erhaltene Resultat zu dem Beispiel `M.I.T.` wird in Abbildung 7 visualisiert.

Abbildung 7: Google-Maps-Web-Service Beispielergebnis

Das Ergebnis liegt im Google-spezifischen Dateiformat *Keyhole Markup Language (KML)* vor, das speziell für den Austausch von geografischen Daten entwickelt worden ist. Es nutzt eine XML-Struktur mit Elementen und Attributen. Das Element `name` enthält nochmals den Suchausdruck, ein Statuscode lässt auf korrekte oder fehlerhafte Ausführung schließen. Die Elemente namens `Placemark`, von denen der Dienst bis zu zehn zurückliefert, enthalten jeweils eine eindeutige Kennung und stellen eine Ergebnis-Adresse dar. Das Unterelement `address` gibt diese in einer kurzen Zusammenfassung wieder, die restlichen Elemente enthalten detailliertere Angaben. Das Element `coordinates` beinhaltet die jeweiligen Koordinaten, die sich wiederum auf WGS84 ([Nat00]) beziehen.

Der Google-Maps-Dienst kann zudem eingeschränkt mit Rechtschreibfehlern umgehen, zu dem Suchterm `Heideiberg` liefert er die korrekte Adresse der Stadt Heidelberg. Mit den genannten Beispielen wird außerdem ersichtlich, dass der Dienst mit verschiedenen Sprachen umgehen kann.

Dieser Dienst kann allerdings nicht ganz ohne Einschränkungen genutzt werden. Zum einen wird ein so genannter *API-Key* benötigt, eine eindeutige Zeichenkette, die von Google Inc. vergeben wird. Erst mit diesem Schlüssel erhält man vom Google-Maps-Dienst Ergebnisse, er muss bei jedem Aufruf an die URL angefügt werden. Um an einen API-Key zu gelangen, muss eine Person ein Google-Konto besitzen, darüber kann man den Schlüssel anfordern. Somit kann der Schlüssel einem Benutzer zugeordnet und eventuellem Missbrauch vorgebeugt werden.
Bei der Nutzung des Dienstes besteht zum anderen die Restriktion, dass pro Schlüssel maximal 50.000 Aufrufe am Tag durchgeführt werden dürfen, demnach durchschnittlich alle 1.728 Sekunden ein Aufruf. Bei einem Verstoß hiergegen wird seitens Google die Benutzung des Dienstes zuerst einen Tag, bei einer Wiederholung auch längerfristig gesperrt.

Auf der SourceForge.net-Web-Seite existiert ein Projekt, welches sich *Geo-Google - Free Address Standardizer*[31] nennt. Dieser Titel klingt hinsichtlich der Aufgabenstellung vielversprechend. Letztendlich stellt es jedoch einzig eine API für die Programmiersprache Java dar, mit der der Google-Maps-Dienst, wie oben besprochen, aufgerufen werden kann. Das Ergebnis (KML-Dokument) wird schließlich eingelesen und objektorientiert zur Verfügung gestellt. Mit diesen Objekten können die Ergebnisse abgefragt werden. Hierbei finden demnach keine inhaltlichen Änderungen am Suchterm oder dem Ergebnis statt, die im Projekttitel enthaltene Adress-Standardisierung basiert vollkommen auf dem Google-Maps-Dienst.

[31] sourceforge.net/projects/geo-google/, 11.12.2007

4.2.4 Zusammenfassung

Aufgrund der Nachteile hinsichtlich der Erstellung eines Programmteils zur Informationsextraktion mittels Verzeichnissen wurden im Kapitel 4.2 Dienste des Web untersucht. Mit dem Google-Maps-Dienst wurde ein Dienst gefunden, der in der Lage zu sein scheint, mit der im Kapitel 3.6 aufgezeigten Datenheterogenität umzugehen. Aus diesem Grund wird dieser Dienst bei der Implementierung eingesetzt. Wie im folgenden Kapitel 5 beschrieben wird, finden aber auch Verzeichnisse zur Unterstützung der Informationsextraktion Anwendung.

Aufgrund der dargelegten Problemstellungen wird die Textsuchmaschine Apache Lucene nicht angewandt. Die Dienste von Geonames wurden ebenso nicht in die Implementierung einbezogen, da der Google-Maps-Dienst leistungsfähiger ist.

5 Implementierungsdetails

Dieses Kapitel hat die Details der Umsetzung der zuvor besprochenen Theorie zum Inhalt. Es beschreibt die Implementierung einer Anwendung, die eine Forschungsinstitutsdatenbank automatisch aufbauen kann. Diesbezüglich wird in einem ersten Kapitel eine Begriffsabgrenzung vorgenommen, die zum grundlegenden Verständnis der anschließenden Beschreibung des Programmaufbaus und spezieller Algorithmen nötig ist. Außerdem werden Rahmenbedingungen hinsichtlich der Umsetzung aufgezeigt. Ein folgender Kapitel beschreibt den allgemeinen Programmfluss. Anschließend erfolgt die Erläuterung der einzelnen Komponenten zur Daten- und Informationsextraktion als auch der endgültigen Datenspeicherung.

5.1 Begriffsabgrenzung

Im Rahmen des Kapitels Implementierungsdetails wird der Begriff *Komponente* verwendet und alternativ zu einem Programmteil verstanden. In Bezug auf den Bereich der Softwareentwicklung besitzt er hingegen eine andere Bedeutung. Das komponentenbasierte Softwareengineering beschäftigt sich mit der Erstellung einzelner Komponenten hinsichtlich der einfachen Wiederverwendbarkeit. Diese Komponenten stellen eigene Applikationen dar, die somit allein lauffähig und über öffentliche Schnittstellen ansprechbar sind [Som04].
In dieser Arbeit liegt der Bedeutungsschwerpunkt des Begriffs Komponente hauptsächlich darauf, dass ein unselbstständiger Programmbestandteil eine spezielle Aufgabe erfüllt und diesbezüglich zusammenhängend betrachtet wird. Eine Komponente stellt hier somit eine Komposition an Funktionalitäten dar, die diesbezüglich unabhängig zu anderen Komponenten behandelt wird, jedoch mit diesen algorithmisch verbunden ist und nicht autonom lauffähig ist.

5.2 Rahmenbedingungen

Im Rahmen der Implementierung stell sich zunächst die Frage nach der einzusetzenden Programmiersprache. Hierfür wurde aufgrund der im Weiteren beschriebenen

Aspekte *Java* der Firma *Sun Microsystems*[32] gewählt.
Zum einen handelt es sich hierbei um eine Programmiersprache, die eine ständige Weiterentwicklung erfährt. Aufgrund ihrer Popularität besteht eine breite Unterstützung im Umgang damit. Diesbezüglich existieren zahlreiche freie Programmierschnittstellen für unterschiedlichste Aufgabenstellungen, von denen ein Teil in dieser Arbeit genutzt werden soll.
Zum anderen fügt sich damit das entstehende Programm in die Fachkenntnis der Abteilung Datenbanken ein. Dadurch werden Anwendung und eventuelle Weiterentwicklung erleichtert.

5.3 Steuerung des Programmflusses

Wie schon in der Einleitung aufgezeigt, ist das Programm logisch in Komponenten geteilt, die bereits theoretisch besprochen wurden. In den folgenden Kapiteln wird die Realisierung dieser Komponenten behandelt. Diese werden in Abbildung 8 nochmals aufgezeigt, wobei der Programmablauf mittels dicken Pfeilen und darin einem Farbverlauf von hell nach dunkel visualisiert ist.

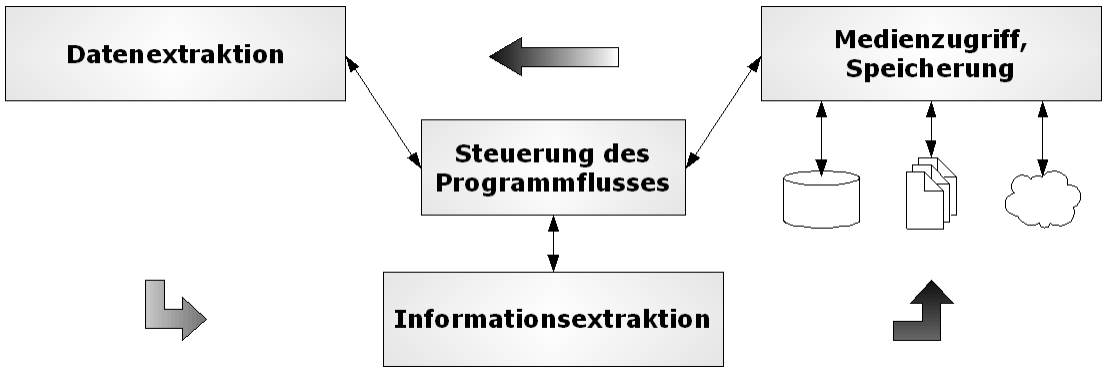


Abbildung 8: Grober, logischer Programmaufbau mit Verlauf

Während der Programmausführung ist es nötig, auf unterschiedliche Informationsträger zuzugreifen. Hierbei werden beispielsweise Start-URLs aus Dateien ausgelesen. Diese URLs verweisen auf Internetseiten, die wiederum analysiert werden sollen. Diesbezüglich muss hierbei auf lokale Dateien zugegriffen werden, aber auch die Existenz einer Datenbankanbindung wird überprüft.

[32] www.sun.com

Mittels der URLs wird über die Medienzugriffs-Komponente auf die Internetseiten zugegriffen und diese geladen. Die Seiten werden der Datenextraktions-Komponente zur Verfügung gestellt, die im anschließenden Kapitel 5.4 näher erläutert wird. Über die Datenextraktions-Komponente werden aus den Internetseiten die benötigten Metadaten aber auch weitere URLs extrahiert, je nachdem was dem Quelldokument entnommen werden kann. Neue URLs verweisen somit auf weitere Internetseiten, von denen Daten geladen werden können. Auf diese Internetseiten wird somit im weiteren Verlauf über die Medienzugriffs-Komponente gleichermaßen zugegriffen. Diese Technologie wird als *crawling*, dies durchführende Programme zum Beispiel als *Crawler* oder *Spider* betitelt. Bezeichnet wird damit das Sammeln neuer Verweise, ausgehend von wenigen Startverweisen [RV03]. Crawler werden beispielsweise von Suchmaschinen eingesetzt, die darauf angewiesen sind, möglichst viel Inhalt des Web zu erfassen.
Die mittels der Datenextraktions-Komponente erfassten Daten werden der Informationsextraktions-Komponente strukturiert zur Verfügung gestellt. Ein Teil dieser Daten wird für die Extraktion der Zugehörigkeitsinformationen genutzt. Die genaue Funktionsweise schildert das Kapitel 5.5.
Sämtliche Daten zu wissenschaftlichen Publikationen und Informationen zu den Zugehörigkeiten der Autoren werden anschließend in einer Datenbank zentral gespeichert. Damit wird die Forschungsinstitut-Datenbank inhaltlich gefüllt, die Datensätze stehen anschließend einer Auswertung oder weiteren Verwendungen zur Verfügung. Kapitel 5.6 beschreibt den Zugriff auf die Datenbank als auch das hier verwendete Schema.

Letztlich besteht die Umsetzung der Programmfluss-Steuerung aus einer Klasse namens *Controller*. Die Kommunikation zwischen den einzelnen Komponenten findet komplett über die Controller-Instanz statt. Diese leitet die Methodenaufrufe der verschiedenen Komponenten in der Regel entsprechend weiter, besitzt demnach kaum aufwändige Logik. Die zentrale Methode *runExtraction* stellt den Einstiegspunkt in die Anwendung dar. Sie initialisiert das Einlesen der Start-URLs aus Dateien, dass der Zugriff auf die mit den URLs referenzierten Dokumente erfolgt und für ein jeweiliges Dokument die Datenextraktion durchgeführt wird.

5.4 Komponente: Datenextraktion

In Kapitel 3.2 wurde die so genannte Wrapper-Technologie eingeführt, die hier zum Einsatz kommen soll. Dabei stellen die hier umgesetzten Wrapper keine eigenständigen Programme, sondern leichtgewichtige, integrierte Teile des Programms dar.
In dieser Arbeit wird jeder Wrapper durch eine eigene Klasse repräsentiert. Aufgrund von Gemeinsamkeiten in der Funktionalität zwischen den verschiedenen Wrappern können diese zusammengefasst werden. Über allgemeine Klassen und Vererbung lässt sich somit für jeden Wrapper eine Spezialisierung durchführen, so dass kein Quelltext doppelt umgesetzt und gewartet werden muss. Die Abbildung 9 zeigt das Klassendiagramm für die hier umgesetzten Wrapper-Klassen.

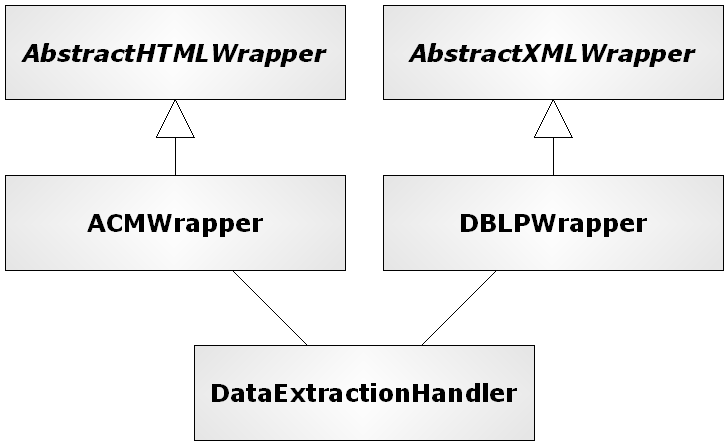


Abbildung 9: Klassendiagramm der Datenextraktions-Komponente

AbstractHTMLWrapper und *AbstractXMLWrapper* stellen zwei abstrakte Klassen dar, die Grundfunktionalitäten implementieren und damit die Grundlage für weitere Unterklassen bilden [Oes06]. Die beiden Klassen lassen darauf schließen, dass zwei verschiedene Arten von Wrappern umgesetzt wurden. Einerseits handelt es sich um Wrapper für Abstract-Seiten, die dem Web entnommen wurden und somit im HTML-Format vorliegen. *ACMWrapper* stellt die Beispielimplementierung für eine Unterklasse hierfür dar, siehe dazu Kapitel 2.7. Andererseits können Wrapper erstellt werden, die mit XML-Dokumenten umgehen können, wie in Kapitel 2.4.1 beschrieben. Für diese Art stellt der *DBLPWrapper* eine Beispielimplementierung einer Unterklasse dar.
Eine so genannte *DataExtractionHandler*-Klasse dient als Schnittstelle zwischen der

zentralen Programmsteuerung und weiteren Inhalten der Datenextraktions-Komponente. Das bedeutet, diese Klasse erfüllt die Aufgabe, die korrekte Wrapper-Instanz auszuwählen. Dies geschieht mit Hilfe überlieferter URLs, die die jeweiligen Dokumente referenzieren. In der Beispiel-URL `http://portal.acm.org/toc.cfm?[...]` kann anhand des *Hostnamens* `acm`, sicherheitshalber zusätzlich mit der *Top-Level-Domain (TLD)* festgestellt werden, dass für das Dokument der *ACMWrapper* genutzt werden sollte, da ACM HTML-Abstract-Seiten anbietet.

Die Klasse *ACMWrapper* definiert unterschiedliche Methoden zur Untersuchung einer Internetseite, da verschiedene Arten möglich sind. Wie in Abbildung 10 zu erkennen, kann eine Internetseite von ACM ein Archiv darstellen, in der Inhaltsverzeichnisse einzelner Jahrgänge aufgelistet sind. Diese Inhaltsverzeichnisse wiederum verweisen auf die einzelnen Abstract-Seiten der jeweiligen Ausgabe.

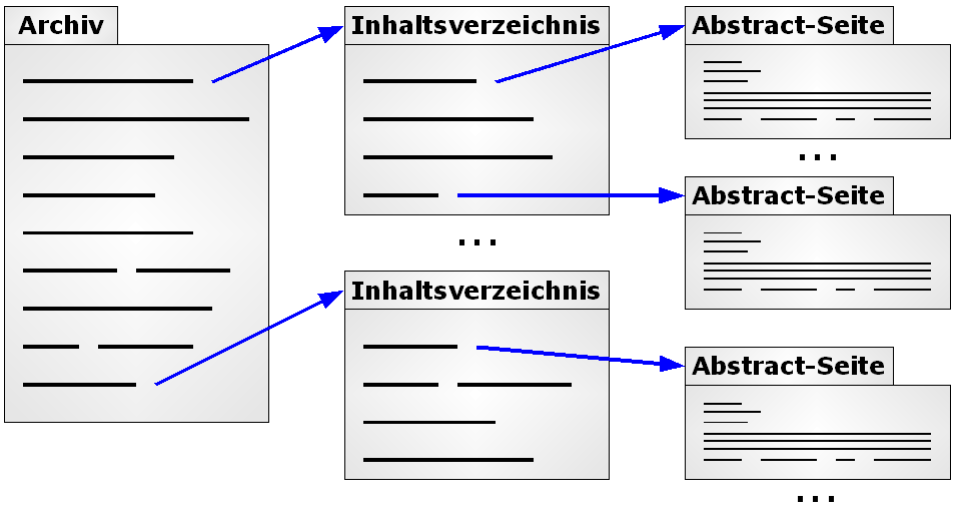


Abbildung 10: Hierarchische Einbindung der Abstract-Seiten

Die Untersuchung nach der Art der Seite ist wichtig, damit die jeweils korrekte Extraktionsmethode ausgewählt wird. Für jede Art existiert eine Methode zur Extraktion der Daten, damit aus Archiven die URLs zu den Inhaltsverzeichnissen und aus denen wiederum die URLs zu Abstract-Seiten entnommen werden. Eine weitere Methode extrahiert die Metadaten zu Publikationen aus den wichtigen Abstract-Seiten.

Der Algorithmus zur Untersuchung einer Internetseite überprüft zunächst, ob die vorliegende Internetseite eine Abstract-Seite ist. Ist dies nicht der Fall, wird auf ein Inhaltsverzeichnis dann auf ein Archiv hin überprüft. Diese *bottom-up*-Reihenfolge

wurde gewählt, da die Anzahl an Abstract-Seiten der Anzahl an Inhaltsverzeichnissen und Archiven in der Regel weitaus übersteigt. Dadurch ist die Wahrscheinlichkeit größer, eine Abstract-Seite vorliegen zu haben.
Da Abstract-Seiten die Grundlage für die Datenextraktion bilden, muss eine Untersuchung in Bezug auf diese Seiten stattfinden. Dass überhaupt auf Archive und Inhaltsverzeichnisse getestet wird und diese bearbeitet werden können, stellt eine Vereinfachung in der Programmanwendung dar. Aufgrund dessen ist es für einen Anwender möglich, lediglich eine Start-URL anzugeben, die auf eine Archiv-Seite beispielsweise einer Zeitschrift verweist. Das Programm sammelt dann alle Publikationsdaten der angeführten Ausgaben.

Für die Wrapper in der Datenextraktions-Komponente werden die in Kapitel 3.3 und 3.4 angesprochenen Technologien (reguläre Ausdrücke, DOM) zum Bearbeiten der HTML- und XML-Dokumente angewandt.
Aufgrund gleicher Darstellung vieler HTML-Dokumente, die von einem Anbieter stammen, lassen sich hierfür Muster finden, die mittels regulären Ausdrücken überprüft beziehungsweise abgefragt werden können. Ein Beispiel hierfür ist der Ausdruck `<small> ([^<]*?)</small>`. Um einen Ausdruck herum müssen die Merkmale außerhalb der Klammern stehen. Der gesuchte Ausdruck hier muss also von den Tags `<small>` und `</small>` umgeben und zusätzlich mit ` ` (HTML-Kodierung eines Leerzeichens) angeführt sein. Innerhalb der Klammern wird ein Ausdruck gesucht, der alle Zeichen außer `<` zulässt. Aufgrund der Klammern kann letztendlich genau dieser Ausdruck abgefragt werden.
Mit diesem Muster erhält man bezüglich der Abbildung 2 auf Seite 14 (Ausschnitt einer ACM-Abstract-Seite) je Autor jeweils die Zeichenkette `University of Leipzig, Germany`.
Die regulären Ausdrücke können mittels der Standard-Java-API[33] seit der Version 1.4 genutzt werden.

Für die Bearbeitung von XML-Dokumenten kann zum Beispiel die freie Java-API *JDOM*[34] angewandt werden. Mit Kenntnis des Dokumentenaufbaus, beispielsweise aufgrund der Existenz einer DTD, kann auf einzelne Elemente zugegriffen werden.

[33] java.sun.com/javase/6/docs/api/
[34] www.jdom.org

Die Methoden zur Extraktion der Daten erstellen eine Instanz einer Klasse, die dafür konzipiert wurde, sämtliche Daten zwischen zu speichern, um sie über die Informationsextraktion letztendlich der Speicherungs-Komponente zu übertragen.

5.5 Komponente: Informationsextraktion

Die Informationsextraktions-Komponente stellt gewissermaßen die zentrale Komponente des Programms dar, da sie die wichtige Funktionalität bezüglich der Aufgabenstellung umsetzt. Sie definiert Algorithmen zur Extraktion von Informationen zu Einrichtungstypen und der Orte aus gegebenen Zeichenketten, für die im Kapitel 3.6 Beispiele aufgeführt wurden. Für diese Informationsextraktion werden verschiedene der in Kapitel 4 besprochenen Technologien (Verzeichnisse, Web Services) eingesetzt.
Die Abbildung 11 soll beginnend einen Überblick über die involvierten Klassen geben, die die Informationsextraktions-Komponente bilden. Die Klasse *InformationExtraction* implementiert den zentralen Algorithmus zur Informationsextraktion, der auf verschiedene Listen (siehe Kapitel 4.1) als auch den Google-Maps-Dienst (siehe Kapitel 4.2.3) zugreift.

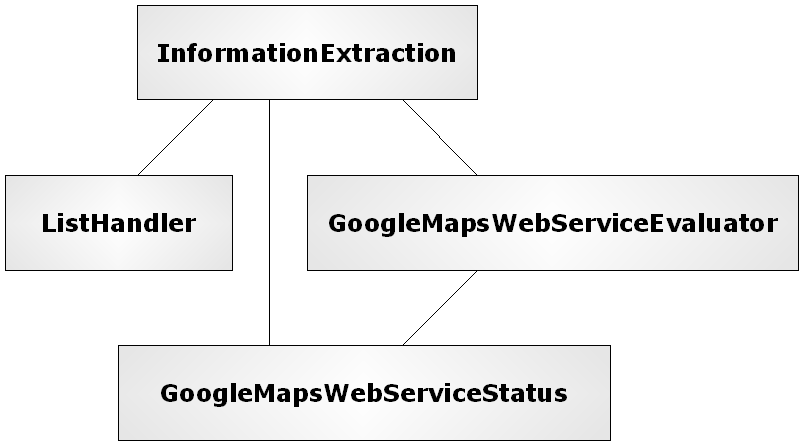


Abbildung 11: Klassendiagramm der Informationsextraktions-Komponente

Der Methode `evaluate` der *GoogleMapsWebServiceEvaluator*-Instanz wird von der *InformationExtraction*-Instanz eine Zeichenkette zur Untersuchung übergeben. Diese Methode erstellt damit die entsprechende URL für den HTTP-Aufruf mit einem API-Key. Der Aufruf selbst geschieht über die Medienzugriffs-Komponente. Das ent-

sprechende Antwort-Dokument wird ausgewertet, bei Erhalt von Ergebnissen werden diese für die weitere Verarbeitung gespeichert und eine Instanz der Klasse *GoogleMapsWebServiceStatus* erstellt. Diese enthält den Status der letzten Abfrage, der für die *InformationExtraction*-Instanz notwendig ist, um auf Erfolg der Abfrage prüfen zu können.
Für die Informationsextraktion wurden verschiedene Verzeichnisse erstellt, die lokal gespeichert sind und über die Medienzugriffs-Komponente geladen werden. Eine Instanz der Klasse *ListHandler* hält Listen vor, um darauf basierende Funktionalitäten anbieten zu können. Diese werden von der *InformationExtraction*-Instanz zusätzlich genutzt und im Laufe dieses Kapitels näher erläutert.

Die Informationsextraktion gliedert sich logisch in eine Vorverarbeitung, auf der die beiden Phasen der Orts- und Institutsextraktion folgen. Diese sollen im Folgenden weiter ausgeführt werden.

5.5.1 Vorverarbeitung

Da eine Extraktion von Informationen ungenau sein kann, wird versucht, diese Ungenauigkeit messbar zu machen. Hierzu wurde ein Parameter namens *locationValidity* eingeführt, der Gleitkommawerte zwischen 0 und 1 annehmen kann und somit eine Wahrscheinlichkeit für die Korrektheit und damit Aussagekraft der erhaltenen Ortsinformationen darstellt. Dieser Parameter wird zu Beginn auf den Wert 0.99 gesetzt, der symbolisiert, dass es keine hundertprozentige (entspricht 1) Zuverlässigkeit gibt. Im Laufe der Informationsextraktion wird der Parameter je nach Ergebnis geändert, um auch ihn letztendlich der Datenspeicherungs-Komponente zu übergeben.

Aufgrund der hohen Datenheterogenität wird in einem weiteren Schritt versucht, die zu untersuchenden Zeichenketten zuvor zu normalisieren. Hierbei wird mit einer Kopie gearbeitet, um die Originale für die Speicherung zu erhalten. Die Normalisierung enthält die folgenden Schritte:

- Leerzeichen zu Beginn und am Ende der Zeichenkette werden gelöscht. Wenn zwischen Wörtern mehrere Leerzeichen aufeinander folgen, werden diese zu einem einzigen gekürzt. Der letzte Schritt wird ebenso für Bindestriche durchgeführt.

- Bestimmte Angaben werden vereinheitlicht. Aus den Termen
  - `... Incorporated`
  - `..., Inc`
  - `..., inc.`

  und so weiter wird `... Inc.` gebildet. Für den Ausdruck `... Ltd.` verhält es sich genauso.
- In den angegebenen Zeichenketten sind gelegentlich E-Mail-Adressen mit angefügt. Um ein Auffinden der Adressen zu erleichtern, wird ebenso versucht, die Verschiedenartigkeit auszugleichen, so dass allein die Teilzeichenkette `, e-mail:` die E-Mail-Adressen vom Rest trennt.

Die Groß-/Kleinschreibung bleibt hingegen unverändert, da sonst essenzielle Informationen verloren gehen könnten. Bei dem Beispiel `IBM` handelt es sich um einen Eigennamen, der nicht zu `Ibm` umgewandelt darf.

In einer weiteren Stufe der Vorverarbeitung gilt es, die jeweilige Zeichenkette daraufhin zu untersuchen, ob mehrere typgleiche Angaben enthalten sind, also mehrere Orte beziehungsweise Einrichtungen. Zunächst wird dabei ermittelt, ob Bindewörter wie das englische `and` in der Zeichenkette vorhanden sind. Ist dies der Fall, wird die gesamte Zeichenkette mit Angaben aus einer Liste verglichen, die anhand von Durchführungsergebnissen manuell erstellt wurde. Die *ListHandler*-Instanz liefert für den Vergleich Zeichenketten, die Einrichtungsangaben darstellen und diesbezüglich Bindewörter beinhalten. Diese Bindewörter trennen in der Regel die Zeichenkette aber logisch nicht in verschiedene Teile. Ein Beispiel hierfür ist `Computer Science and Engineering`. Alle Angaben dieser Art werden entfernt und anschließend wiederum auf verbleibende Bindewörter überprüft.
Auf den Resultaten der Datenextraktions-Durchführung basierend wurden empirisch lineare Funktionen entwickelt, die eine Wahrscheinlichkeit dafür berechnen, ob eine Zeichenkette mehrere unterschiedliche Angaben enthält. Für die beiden Fälle einer Zeichenkette mit und ohne Bindewort existiert jeweils eine Funktion. Dies wird in Abbildung 12 veranschaulicht.

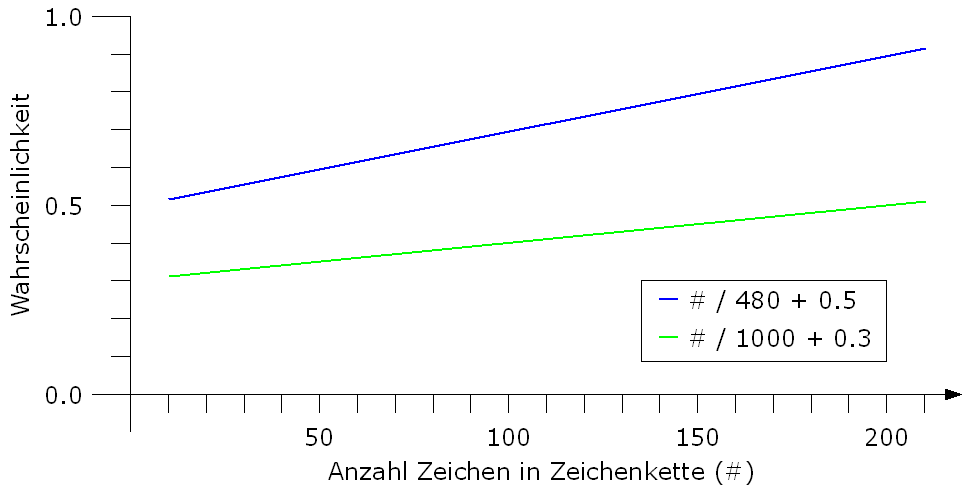


Abbildung 12: Funktionen zur Berechnung der Wahrscheinlichkeit mehrerer Angaben

Die grüne Linie stellt die Funktion für Zeichenketten ohne Bindewort, die blaue Linie für Zeichenketten mit derartigen Wörtern dar. Dass beide Funktionen mit zunehmender Anzahl an Zeichen mehr oder weniger ansteigen, gründet auf der Annahme, dass lange Zeichenketten vergleichsweise eher unterschiedliche Angaben aufweisen könnten.
Diese Funktionen basieren auf Erfahrungswerten, wobei die längste erfasste Zeichenkette 211 Zeichen umfasste. Sollten in weiteren Durchläufen längere Zeichenketten auftreten, muss eventuell eine manuelle Anpassung der Funktionen durchgeführt werden.
Die Berechnung der Wahrscheinlichkeit der Angaben mehrerer Orte oder Einrichtungen kann somit unabhängig vom nächsten Schritt, der Zerlegung in diese einzelnen Angaben, betrachtet und bei Bedarf angepasst werden. Er bildet lediglich eine Entscheidungsgrundlage in Bezug auf die Zerlegung, damit diese nicht bei jeder Zeichenkette durchgeführt werden muss.

Abschließend in der Vorverarbeitung erfolgt eine Zerlegung der Zeichenkette in die verschiedenen Angaben, wenn die Wahrscheinlichkeit 0.5 oder mehr beträgt. Hierbei werden wieder alle Angaben gesucht, die Bindewörter enthalten, die aber keine Aufteilung erfordern. Jedes Zeichen dieser Angaben wird diesmal aber in einer Kopie der Zeichenkette mit jeweils einem Leerzeichen ersetzt. Damit wird es möglich, die Stellen exakt zu identifizieren, wo sich noch Bindewörter befinden. In der Kopie übrig bleibende Bindewörter sind auch im Original an der selben Stellen. Somit ist

bekannt, wo in der originalen Zeichenkette getrennt werden muss.
Ein Beispiel soll dies verdeutlichen. Aus der zusammenhängenden Zeichenkette

```
USAF Wright R & D Center, WRDC/TXI, Dayton, OH and
Wright State University, Dept of Computer Science, Dayton, OH
```

wird

```
USAF Wright       Center, WRDC/TXI, Dayton, OH and
Wright State University, Dept of Computer Science, Dayton, OH
```

gebildet (R & D ist eine Abkürzung für Research & Development und steht in der oben genannten Liste der Einrichtungsangaben mit Bindewörtern). Das Bindewort `and` gibt die Stelle an, an der getrennt werden muss. Somit werden die beiden Zeichenketten

```
USAF Wright R & D Center, WRDC/TXI, Dayton, OH
```

und

```
Wright State University, Dept of Computer Science, Dayton, OH
```

erhalten, das Bindewort wird dabei gelöscht.

Alle so erfassten Zeichenketten können anschließend unabhängig voneinander hinsichtlich der Orte und Einrichtungstypen untersucht werden. Das schließt die Originalzeichenkette mit ein, wenn keine Trennung stattfand.

5.5.2 Ortsextraktion

Dieser Kapitel beschreibt die automatische Extraktion der Ortsinformationen, die wesentlich auf die Eigenschaften des Google-Maps-Dienstes aufsetzt, siehe Kapitel 4.2.4.

Da es häufig nicht ausreicht, die gesamte zu untersuchende Zeichenkette an den Google-Maps-Dienst zu senden, da dieser nicht mit allen Angaben umgehen kann, müssen die relevanten Daten herausgefiltert werden. Für die Zeichenkette

```
Wright State University, Dept of Computer Science, Dayton, OH
```

liefert der Dienst beispielsweise keine Informationen, für `Dayton, OH` allerdings

erhält man welche.
Aufgrund dessen werden verschiedene Funktionalitäten umgesetzt, die solche Extraktionen wenn nötig durchführen. Diese Funktionen werden als erstes beschrieben, um sie anschließend im Gesamtkontext des Extraktionsalgorithmus nur noch nennen zu müssen. Die Funktionen bauen in der Regel darauf auf, dass verschiedene Angaben (Institut, Stadt, Land, etc.) in einer Zeichenkette mittels Kommata voneinander getrennt sind. Außerdem implizieren die Funktionen den anschließenden Aufruf des Google-Maps-Dienstes mit den erhaltenen Teilzeichenketten, um auch diesen Sachverhalt nicht jederzeit wiederholen zu müssen.

Google-Maps-Dienst Beim Aufruf des Google-Maps-Dienstes wird der *GoogleMapsWebServiceEvaluator*-Instanz ein Zeichenkette übergeben. Die jeweilige Methode sorgt für den entsprechenden Aufbau der aufzurufenden URL mit API-Key und passender Kodierung. Der Medienzugriffs-Komponente wird die URL zum Aufruf des Dienstes übergeben und von ihr anschließend das Ergebnisdokument erhalten. Die *GoogleMapsWebServiceEvaluator*-Instanz speichert sämtliche Ergebnisdaten zwischen und legt eine entsprechende *GoogleMapsWebServiceStatus*-Instanz an. Diese wird im Ortsextraktionsalgorithmus genutzt, um abzufragen, ob die Anfrage erfolgreich war und brauchbare Ergebnisse zurückgeliefert wurden. Ist dies der Fall, können die Resultate von der *GoogleMapsWebServiceEvaluator*-Instanz abgefragt werden. Anderenfalls erfolgt eine automatische Fehlerbehandlung.

Einrichtungsidentifikatoren In manchen Fällen, wenn keine ausreichenden Ortsangaben aufgefunden werden können, ist es notwendig, eindeutige Einrichtungsidentifikatoren wie zum Beispiel `Politecnico` zu nutzen. Hierzu wurde ebenso eine Liste manuell erstellt. Mittels der Identifikatoren können die gesamten Zeichenketten zwischen zwei Begrenzern (Kommata, etc.) entnommen werden, die eine bestimmte Einrichtung beschreiben und mit denen wiederum Ortsangaben möglich sind. Aus `Politecnico di Milano, Dipartimento di Elettronica e Informazione` lässt sich so die Zeichenkette `Politecnico di Milano` herauslösen, um diese weiter zu analysieren.

Regulärer Ausdruck Allgemein Es wurde ebenso eine Liste erstellt, die der für die Einrichtungsidentifikatoren gleicht. Allerdings sind dort reguläre Ausdrücke aufgeführt. Mit zum Beispiel `feruniv. ([^ ,]*)` kann somit die Angabe hinter dem Stichwort `Fernuni.` extrahiert werden. Diese regulären Ausdrücke fragen bestimmte Muster ab, bei der Zeichenketten erhalten werden, die wiederum in der Regel Ortsnamen sind. Bei `Fernuni. Hagen` wäre dies demnach die Zeichenkette `Hagen`, mit dem der Google-Maps-Dienst angefragt werden kann.

Regulärer Ausdruck USA 1 Viele Publikationen stammen aus den Vereinigten Staaten von Amerika (USA). Aus diesem Grund gleichen sich Angaben oftmals in ihrer Syntax, für die ein regulärer Ausdruck zur Untersuchung angelegt werden kann. Mit `[^,]+,[ ]?[A-Z]{2}[ ]?$` lassen sich Angaben der Struktur `Boston, MA` (Zeichenkette ohne Kommata, Komma, zwei Großbuchstaben) extrahieren, die am Ende einer Zeichenkette vorkommen.
Derartige reguläre Ausdrücke können hauptsächlich für amerikanische und kanadische Angaben erstellt werden, da die beiden Großbuchstaben hierfür charakterisierend sind. Diese stellen im Allgemeinen Abkürzungen von amerikanischen oder kanadischen Staaten dar.

Regulärer Ausdruck USA 2 Für vorrangig amerikanische aber auch kanadische Angaben gibt es weiterhin noch eine andere Struktur. Mit
`[^,]+,[ ]?[A-Z]{2}[ ]?,[^,]+$` können Angaben extrahiert werden, bei denen zusätzlich am Ende das Land angegeben wurde, wie beispielsweise in `Waterloo, ON, Canada`.

Steigernde Überprüfung Bei einem Teil der zu untersuchenden Zeichenketten fällt eine weitere Übereinstimmung in der Syntax auf. Gewöhnlich folgen die Angaben dem Muster, dass am Ende die allgemeinste Ortsangabe steht, zum Beispiel der Staat, davor eine nähere Angabe wie die Stadt und so weiter, wie beispielsweise in `University of Waterloo, Waterloo, Ontario, Canada`.
Hier ist es möglich, die Zeichenkette anhand der Kommata aufzuteilen und von hinten nach vorn steigernd zu überprüfen. Das bedeutet, der letzte Teil (hier `Canada`) wird mittels des Google-Maps-Dienstes untersucht. Bei einem Erfolg wird der voran-

gegangene Teil hinzugenommen (`Ontario, Canada`) und so fort. Bei einem Misserfolg kann das vorherige Ergebnis durch Zwischenspeicherung oder erneutem Google-Maps-Dienst-Aufruf genutzt werden.
Diese Vorgehensweise ist allerdings aufgrund häufiger Aufrufe bezüglich einer Untersuchung sehr zeitaufwändig, wird daher als eine der letzten Möglichkeiten eingesetzt.

Im Folgenden wird der implementierte Algorithmus erläutert, der obige Funktionalitäten einsetzt. Die erste Aufgabe in der Informationsextraktion hinsichtlich der Orte ist die Extraktion von eventuell vorhandenen E-Mail-Adressen. Einerseits stören diese Angaben bei der Umsetzung der oben beschriebenen Funktionen, andererseits werden sie vorerst nicht benötigt. Für eine abschließende Ergebnisüberprüfung können die E-Mail-Adressen allerdings eingesetzt werden. Dafür und für eine Speicherung in der Datenbank werden sie zunächst zwischengespeichert.
Auf der Basis der Datennormalisierung, siehe Kapitel 5.5.1, können E-Mail-Adressen anhand der Teilzeichenkette `, e-mail:` entnommen werden. Hierbei werden die Adressen ebenso vereinheitlicht, da zum Teil alte HTML-Kodierungen (zum Beispiel `&commat;` für `@`) oder Unterschiede in der Umklammerung mehrerer Namen vorgefunden wurden.
Mit der so erhaltenen Zeichenkette ohne E-Mail-Adresse wird daraufhin über die Medienzugriffs-Komponente zunächst untersucht, ob sie vorher schon analysiert worden ist. In diesem Fall erfolgt keine weitere Ortsanalyse.

Der gesamte Algorithmus zur Extraktion der Ortsinformationen wird in Abbildung 13 veranschaulicht. Kursiv werden in der Abbildung Entscheidungen dargestellt.
Dabei wird zum einen untersucht, ob eine Zeichenkette Kommata beinhaltet. Ist dies nicht der Fall, können einige reguläre Ausdrücke implizit kein Ergebnis liefern. Zum anderen wird die Zeichenkette eventuell anhand der Kommata getrennt und solche Teile entfernt, die Einrichtungsidentifikatoren beinhalten. Wenn daraufhin keine Teile übrig bleiben, ist eine steigernde Überprüfung sinnlos.
Wurden dabei keine Teile entfernt, würde eine Abfrage der übrigen Zeichenkette per Google-Maps-Dienst die selben Ergebnisse wie vorher liefern.

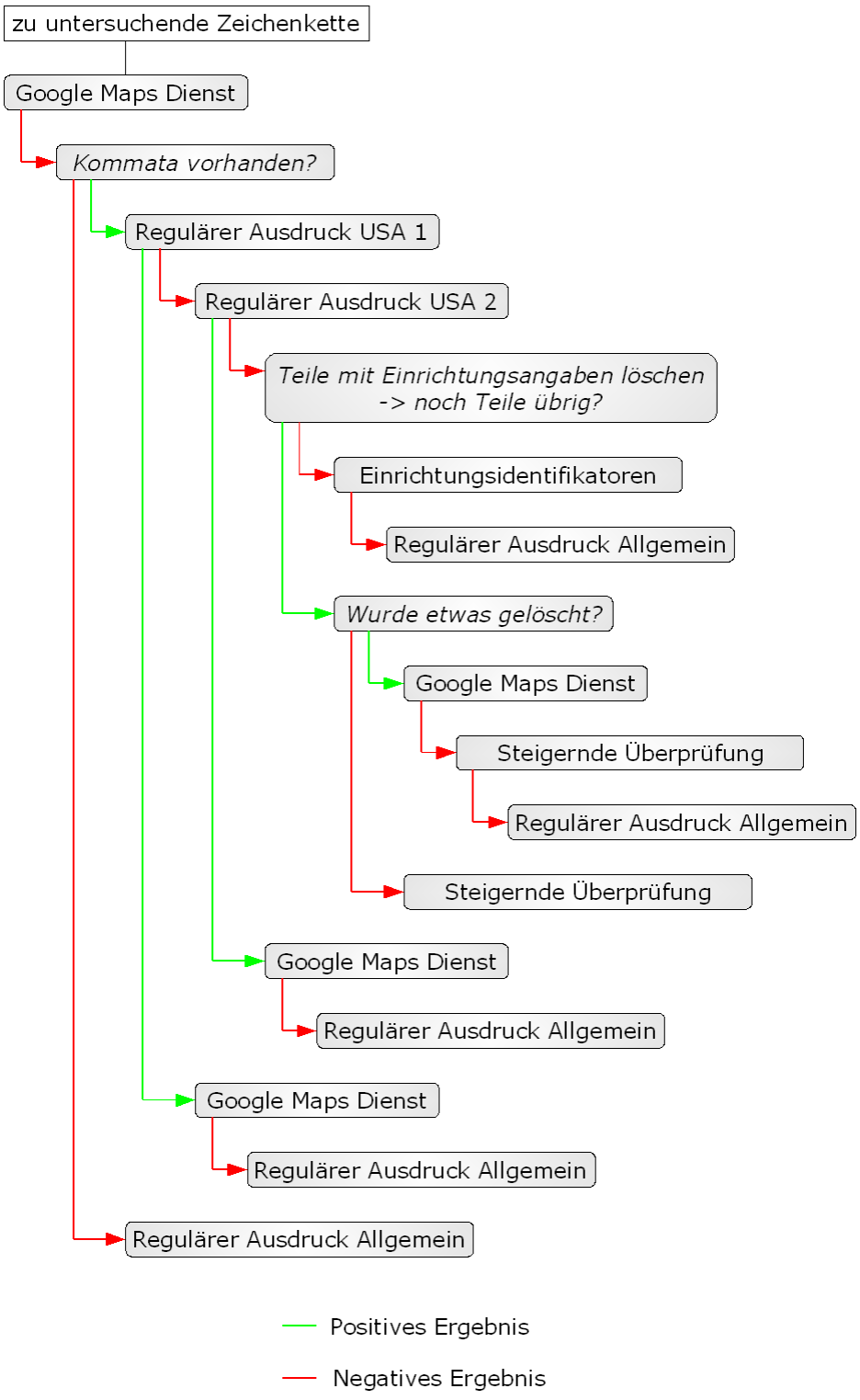


Abbildung 13: Visualisierung des Ortsextraktions-Algorithmus

Wie der Abbildung 13 entnommen werden kann, symbolisiert ein roter Pfeil ein negatives Ergebnis. In solch einem Fall wird der Parameter *locationValidity* verringert, beispielsweise mit 0.8 multipliziert. Denn ein nicht sofort erhaltenes Ergebnis ist erfahrungsgemäß ungenauer.
Wenn Funktionen am Ende einer Entscheidungskette keine Ergebnisse liefern, so werden letztendlich keine Ortsinformationen zu der Zeichenkette gespeichert.
Ein positives Ergebnis einer Google-Maps-Dienst-Anfrage hat zur Folge, dass das Resultat genutzt wird und keine weitere Untersuchung stattfindet. Wenn zu Beginn eine E-Mail-Adresse entnommen und zwischengespeichert wurde, wird diese in einem weiteren Schritt für eine Verifikation des Ergebnisses genutzt. Wenn die TLD dieser Adresse einem Land zugeordnet werden kann, wie zum Beispiel *Deutschland* für *de*, wird dieses Land mit dem im Ergebnis verglichen. Zwei Funktionen wurden ermittelt, die den Parameter *locationValidity* entsprechend gewichten. Bei positiver Übereinstimmung erfolgt die Gewichtung mit $-x^2 + 2x$ (x entspricht dem Wert von *locationValidity*). Kleine Werte erfahren damit gegenüber großen Werten eine höhere Gewichtung, wobei die Grenzen von 0.0 und 1.0 eingehalten werden. Wenn keine Gleichheit besteht, wird mit $0.5x^2 + 0.25x$ gewichtet, große Werte werden hierbei stärker verringert.

5.5.3 Institutsextraktion

Im Anschluss an die Extraktion der Ortsinformationen werden Einrichtungstypen identifiziert. Hierfür wurde eine Liste, wie in Kapitel 4.1.2 beschrieben, mit Einrichtungs-Zeichenketten als auch deren Abkürzungen beziehungsweise verschiedenen Schreibweisen erstellt, die jeweils eine Zuordnung zu einem Einrichtungstyp enthält. Die zu untersuchende Zeichenkette wird anhand von Kommata in Elemente geteilt. Für jedes Element wird daraufhin untersucht, welche Einrichtungs-Zeichenketten es enthält, unabhängig von der Groß- beziehungsweise Kleinschreibung einzelner Zeichen, und deren Typen ermittelt. Die Informationen der erkannten Typen und des jeweiligen Elements werden anschließend zwischengespeichert. Aus

```
Wright State University, Dept of Computer Science, Dayton, OH
```

werden somit der Typ `academy` aufgrund von `University` und der Typ `department` wegen `Dept` extrahiert.

Die zugrunde liegende Liste wurde wiederum empirisch aufgestellt und für die bearbeiteten Daten vervollständigt. Hierbei wurden ebenso verschiedene Sprachen berücksichtigt. Die Abbildung auf einen gemeinsamen Typ, von zum Beispiel `Universität`, `University`, `Università`, `Université` und `Universidad` zu dem Typ `academy`, bewirkt damit eine Vereinheitlichung der Ergebnisdaten.
Dabei ist es ebenso denkbar, die Einrichtungen auf einen numerischen Typ abzubilden. Während Zeichenketten für den Menschen besser interpretierbar sind, lassen sich mit numerischen Werten programmatische Auswertungen einfacher durchführen, um zum Beispiel Durchschnittswerte zu bilden.

5.6 Komponente: Medienzugriff, Datenspeicherung

Für den Umgang mit den verschiedenen Medien wurde eine Komponente implementiert, die in der Lage ist, Daten aus einer Datenbank zu lesen als auch in diese zu schreiben, Dateien auszulesen sowie Web-Seiten aufzurufen. Diese Funktionalitäten werden von den in Abbildung 14 aufgezeigten Klassen umgesetzt.

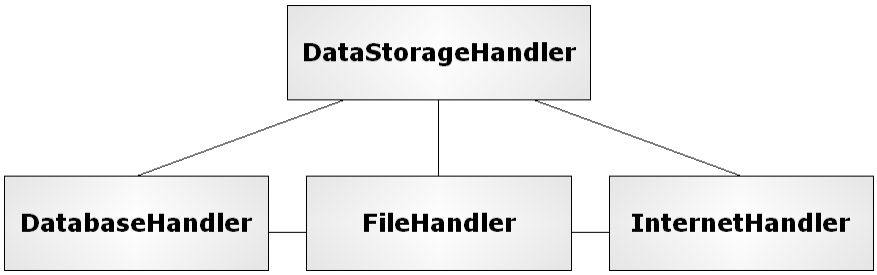


Abbildung 14: Klassendiagramm der Medienzugriffs-, Datenspeicherungs-Komponente

Die *DataStorageHandler*-Klasse dient als zentrale Instanz zum Aufruf der Funktionen, sie delegiert die Anfragen an die entsprechenden Klassen weiter.

Die Klasse *FileHandler* stellt Funktionen zum Laden verschiedener Dateien bereit. Hierbei handelt es sich beispielsweise um die Start-URLs für die Datenextraktion (siehe Kapitel 5.4). Außerdem können die genannten Listen geladen werden, die für

die Informationsextraktion von Bedeutung sind (siehe Kapitel 5.5.2 sowie 5.5.3). Weiterhin können Anwender-Einstellungen für das Programm aus einer Datei gelesen werden. Die Einstellungen definieren zum Beispiel Zugangsdaten zu einer Datenbank oder aber den für den Google-Maps-Dienst notwendigen Google-Maps-API-Key (siehe Kapitel 4.2.3). Außerdem werden mit dieser Klasse einmal aufgerufene Web-Seiten und eine Datei mit Logging-Informationen lokal gespeichert.

Die *DatabaseHandler*-Klasse repräsentiert den wesentlichen Teil bezüglich der Datenspeicherung. Alle Daten zu den bearbeiteten Publikationen mit den dazu extrahierten Orts- und Einrichtungsinformationen werden in eine Datenbank geschrieben. Hierbei stellt eine Instanz der Klasse mittels der Java-API *JDBC* die Verbindung zu einer relationalen Datenbank her, erstellt wenn nötig die im Folgenden beschriebenen Datenbanktabellen und bietet Methoden zum Speichern von Daten. Durch die Verwendung von JDBC lassen sich Anweisungen in der *Structured Query Language (SQL)* an nahezu jede beliebige relationale Datenbank senden [HCF98].
Eine Übersicht des Datenumfangs in Form des Datenbankschemas bietet Abbildung 15.

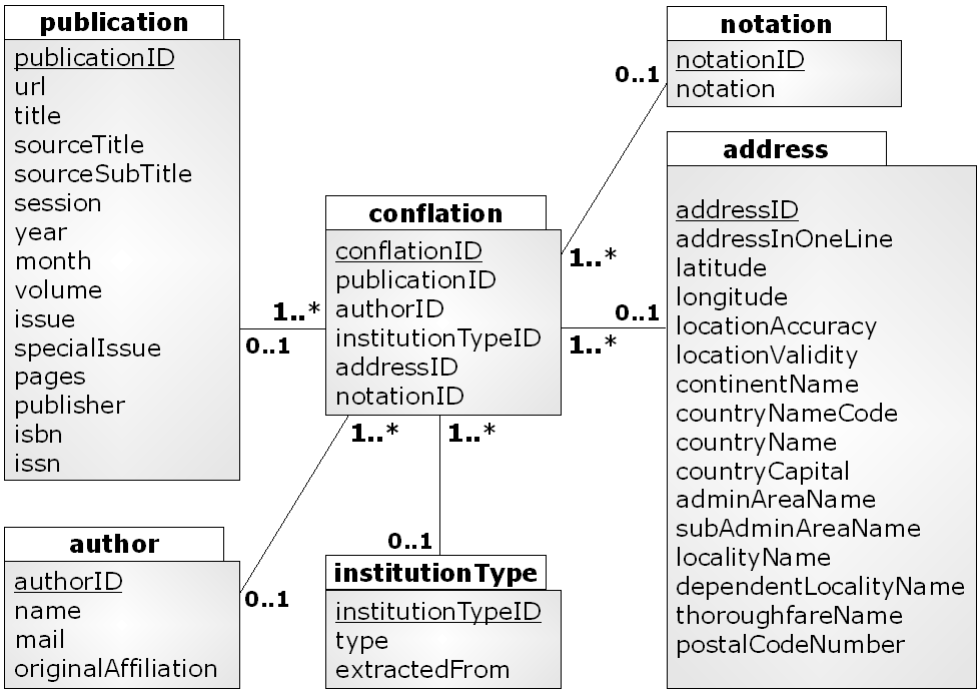


Abbildung 15: Datenbankschema

Eine zentrale Tabelle namens *conflation* (engl. für Zusammenführung) verbindet die eigentlichen Datentabellen über die jeweiligen Fremdschlüssel. Somit wird gewährleistet, dass kein Datensatz mehrfach gespeichert wird und sämtliche Zusammengehörigkeiten gespeichert werden.
Die restlichen Tabellen enthalten Daten zu den Publikationen, aller Autoren, der jeweiligen Adressbezeichnung und den daraus extrahierten Informationen zu den Orten als auch den Einrichtungstypen. Die Tabelle *author* enthält ein Feld *originalAffiliation*, das die ursprüngliche Zeichenkette enthält, die zur Informationsextraktion herangezogen wird. Die Tabelle *notation* hingegen enthält die Teile davon, die aufgrund der Vorverarbeitung erhalten wurden, siehe Kapitel 5.5.1. Die auf dieser Basis ermittelten Adressen werden in der Tabelle *address* gespeichert. Die Felder der *address*-Tabelle entsprechen im Großteil den Elementen des Google-Maps-Dienstes, wie in Kapitel 4.2.3 zu sehen. Während der Informationsextraktion wird der Wert *locationValidity* berechnet, der in Kapitel 5.5.1 einführend besprochen wurde und ebenso in der Datenbank gespeichert wird.

Die Klasse *DatabaseHandler* bietet weiterhin die Möglichkeit, eine Zeichenkette daraufhin zu überprüfen, ob sie bezüglich der gesuchten Informationen zu Orten und Einrichtungstypen bereits untersucht wurde und aus diesem Grund die Zeichenkette und zugehörige Informationen schon in der Datenbank stehen. In diesem Fall muss diese Zeichenkette nicht erneut analysiert werden.

Die *InternetHandler*-Klasse dient dem Aufrufen von Web-Seiten, deren entsprechende URL jeweils an die hierfür zuständige Methode übergeben werden muss. In manchen Situationen ist es notwendig, vor dem Laden von Web-Seiten eine Weile zu warten. Beispielsweise wurde in Kapitel 4.2.3 die Beschränkung auf 50.000 Google-Maps-Dienst-Aufrufe besprochen. Außerdem kann bei der Datenextraktion, wie in Kapitel 5.4 beschrieben, das pausenlose Laden von Abstract-Seiten bei dem selben Anbieter Probleme verursachen. Denn auf Anbieterseite wird versucht, programmatisches Auslesen zu blockieren, weil dadurch ein erhöhtes Datenaufkommen entsteht. Die *InternetHandler*-Klasse speichert die Zeiten des letzten Zugriffs auf eine Web-Seite eines Anbieters. Ein in einer Programm-Konfigurations-Datei anzugebender Wert legt eine Mindestpause in Millisekunden fest, die bis zum nächsten Aufruf eingehalten werden muss. Ein Wert von 10.000, der demnach 10 Sekunden entspricht, hat sich als Pause für das Aufrufen von ACM-Abstract-Seiten als sinnvoll erwiesen.

Für den Fall eines nicht erfolgreichen Aufrufs lassen sich vor dem Programmstart ebenso Werte in der Programm-Konfiguration angeben, die Anzahl an und Pause zwischen erneuten Aufrufversuchen festlegen.

5.7 Zusammenfassung

Dieses Kapitel 5 zeigte die Implementierungsdetails des in dieser Arbeit entwickelten Programms, welches anhand weniger Vorgaben (Start-URLs) und Einstellungen (Pause-Werte, Datenbankverbindungsparameter) in der Lage ist, automatisch Dokumente zu laden, Daten und mit diesen weitere Informationen zu Orten und Einrichtungstypen zu erschließen, um diese letztendlich in einer Datenbank abzulegen.

In Abbildung 16 ist dieser Programmablauf in einem Sequenzdiagramm aufgezeigt, das die Interaktionen zwischen den jeweils beteiligten Klassen für ein bestimmtes Szenario zeigt. Der zeitliche Ablauf mit den ausgetauschten Nachrichten wird dabei abgebildet [Oes06]. Das Diagramm stellt den einfachen Anwendungsfall dar, eine URL für eine Abstract-Seite aus einer Datei zu laden, das Web-Dokument aufzurufen, lokal zwischen zu speichern, um anschließend die Daten und Informationen zu extrahieren und diese abschließend in der Datenbank zu speichern. Weitere Zwischenschritte, wie beispielsweise die Ermittlung, ob ein Web-Dokument bereits lokal vorliegt, sind möglich. Wenn außerdem eine Start-URL auf ein Inhaltsverzeichnis verweist, wird dieses Dokument ebenso geladen und weitere URLs daraus entnommen, die weitere Web-Dokumente referenzieren, wie in Kapitel 5.4 beschrieben. Um die Übersichtlichkeit zu wahren, zeigt das Sequenzdiagramm lediglich einen einfachen, allgemeinen Fall der Untersuchung einer einzelnen Abstract-Seite.
Durch das Diagramm wird der Ablauf deutlich, der in den Kapiteln 5.3 bis 5.6 in einzelnen Teilen besprochen wurde. Die Teile der Daten-, Informationsextraktionen und Datenspeicherung wurden deshalb im Diagramm überschaubar zusammengefasst dargestellt.

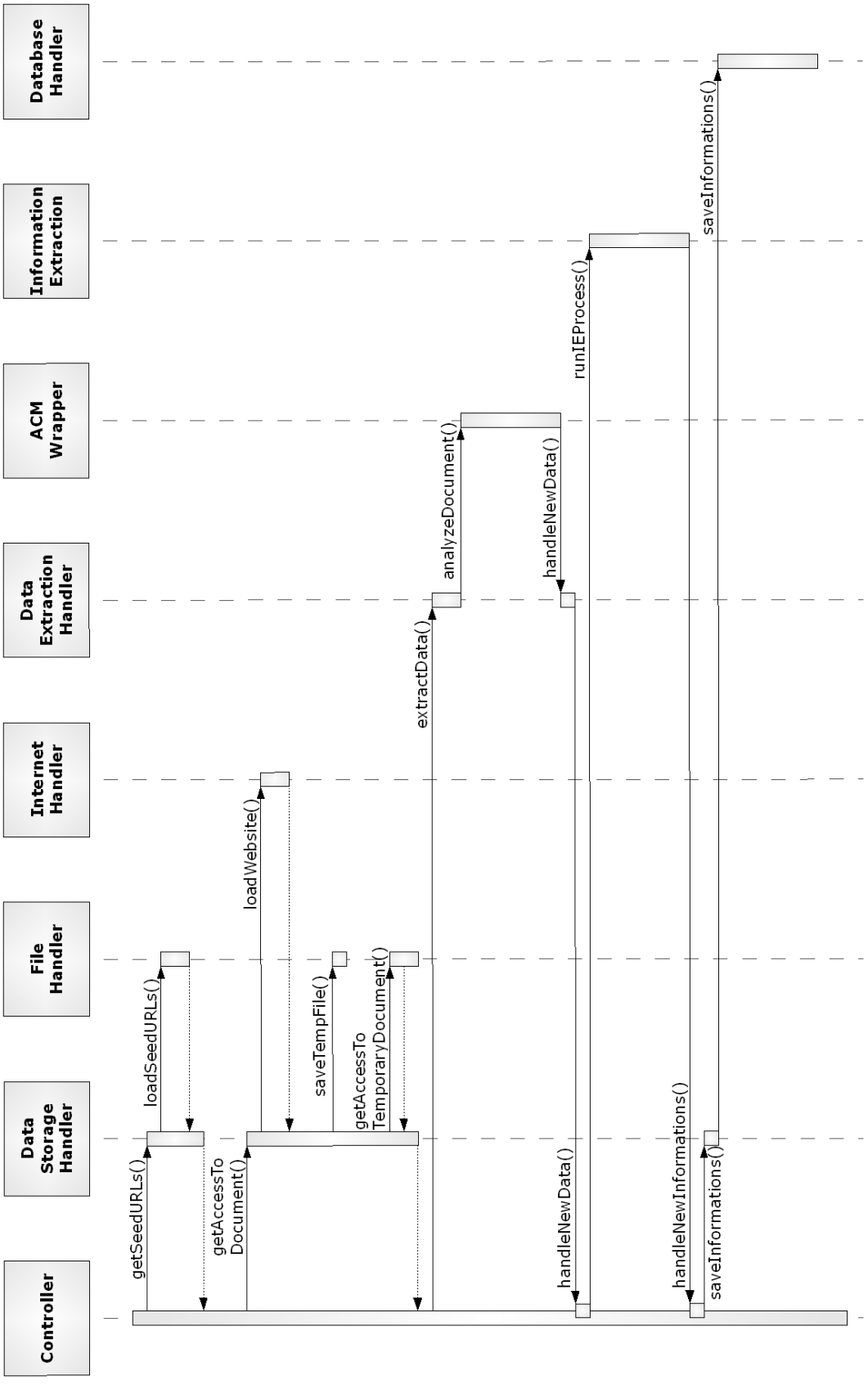


Abbildung 16: Sequenzdiagramm am Beispiel einer Abstract-Seite

Die aufgezeigten Klassen folgen dem Paradigma *Singleton* der objektorientierten Programmierung. Während des gesamten Programmablaufs existiert jeweils nur eine Instanz einer Klasse [GHJV95], die über eine spezifische Methode `getInstance()` aufgerufen wird. Diese statische Methode erzeugt bei Bedarf zunächst die Instanz der jeweiligen Klasse. Somit wird der Ablauf nicht durch unnötiges Instanziieren und Löschen dieser Instanzen verzögert. Weiterhin erleichtert dies die Programmierung, wenn zum Beispiel über Objektattribute Zustände gespeichert sollen und damit weitergearbeitet werden soll.

6 Ergebnisse, Auswertung

In diesem Kapitel werden Ergebnisse betrachtet, die Daten zu wissenschaftlichen Publikationen enthalten. Mit dem in dieser Arbeit entwickelten Programm konnte eine umfangreiche Datenbank mit allgemeinen Publikationsdaten und den gewünschten Adress- und Einrichtungstyp-Informationen aufgebaut werden.
In den ersten Kapiteln werden wichtige Details des Ausführungsszenarios geschildert und ein allgemeiner Überblick über die Ergebnisse gegeben. Hierbei finden auch Vergleiche mit der Arbeit [RT05] statt, aus der einige Ausgangspunkte für die hier gestellte Aufgabe entstammen. Anschließend erfolgt eine Bewertung der Adressinformationen, um einen effektiven Einsatz des Programms beispielsweise mit Bezug zum eingeführten Parameter `locationValidity` abzugrenzen. Weiterhin wird untersucht, inwiefern vorhandene E-Mail-Adressen die Ergebnisse beeinflusst haben.

6.1 Szenariobeschreibung

Mit dem entwickelten Programm wurden Daten zu den Publikationen gesammelt, die in der wissenschaftlichen Arbeit [RT05] schon Gegenstand einer Untersuchung waren. Wie in der Motivation (Kapitel 1.1) beschrieben, wurde diese Untersuchung allerdings manuell und deshalb mit einigen Einschränkungen durchgeführt.
Durch die Nutzung der gleichen fünf Bibliografien[35] für die Daten- und damit auch der anschließenden Informationsextraktion wird somit eine Vergleichsmöglichkeit der Ergebnisse geschaffen.

Für die Ausführung wurden jeweils die Web-Seiten von ACM gesucht, die die Archive (siehe Kapitel 5.4) für die genannten Bibliografien darstellen. Die fünf zugehörigen URLs wurden in einer Datei gespeichert, um sie dem Programm als Start-URLs zu übergeben. Das Programm sucht damit die Inhaltsverzeichnisse und daraus wiederum die Abstract-Seiten selbstständig, wie in Kapitel 5.4 erläutert.

Es wurden zwei verschiedene Wege gewählt, um die Art und Weise der Ausführung aufzuzeigen.
Zum einen wurde jede der fünf Bibliografien einzeln und damit die zugehörige Start-

[35] SIGMOD Conference, SIGMOD Record, TODS, VLDB Conference, VLDB Journal

URL gewählt. Die jeweiligen Ausführungsergebnisse zu einer Start-URL wurden in einer eigenen Datenbank abgelegt. Zum anderen wurden dem Programm die gesamten fünf URLs auf ein Mal übergeben, die Ergebnisse ebenso in einer separaten Datenbank gespeichert.
Somit wurden im Januar 2008 sechs unterschiedliche Datenbanken aufgebaut, wobei letztere die Daten der ersten fünf beinhaltet.

6.2 Allgemeine Ergebnisse

Die Tabelle 8 gibt einen Überblick über statistische Informationen zu den Ergebnissen. Die Werte in der Zeile *Summe* geben jeweils die Summe dieser Spalte oberhalb dieser Zeile an, also der fünf Bibliografien. Die Zeile *Zusammen* beinhaltet die Werte für den zweiten Ausführungsweg, alle fünf Start-URLs mit einem Mal zu behandeln. Auf Basis der fünf Start-URLs wurden je Ausführungsart 7.669 Web-Dokumente heruntergeladen und lokal gespeichert. Ein Großteil davon stellt ACM-Abstract-Seiten für jeweils eine Publikation dar.

	html	**addr**	**auth**	**conf**	**inst**	**nota**	**publ**
SIGMOD Conference	2.165	1.764	5.355	7.875	1.242	2.034	2.114
SIGMOD Record	2.280	2.004	5.246	8.076	1.465	2.293	2.226
TODS	704	676	1.400	1.644	498	720	669
VLDB Conference	2.181	629	4.104	6.297	472	715	2.145
VLDB Journal	339	515	986	1.562	430	554	321
Summe	7.669	5.588	17.091	25.454	4.107	6.316	7.475
Zusammen	7.669	3.901	13.303	25.454	2.407	4.382	7.475

html ... Anzahl zugehöriger HTML-Dokumente
addr ... Anzahl erhaltener Adressen
auth ... Anzahl extrahierter Autoren
conf ... Anzahl an Zusammenführungen
inst ... Anzahl erhaltener Einrichtungstypen
nota ... Anzahl verschiedener Ortsangaben
publ ... Anzahl extrahierter Publikationen

Tabelle 8: Allgemeine Ergebnisübersicht

In der Tabelle ist zu sehen, dass die Menge der HTML-Dokumente in Bezug auf die beiden Ausführungswege erwartungsgemäß gleich ist, wie auch die Anzahl der ermittelten Publikationen und der Zusammenführungen.
Die Werte in den restlichen Spalten variieren voneinander. Da keine doppelten Werte in den Datenbanktabellen vorkommen, reduziert sich die Anzahl der Datensätze, wenn man mehrere Bibliografien in einer Datenbank zusammenfasst. Denn in zwei (hier thematisch eng verbundenen) Bibliografien können beispielsweise oft die selben Adressen auftreten. Wenn ein Datensatz schon vorhanden ist, wird der jeweilige Primärschlüssel herausgesucht und dieser in der Tabelle *conflation* eingetragen. Nicht nur die Ausführungszeit des Programms verringert sich durch den Abgleich mit Daten aus der Datenbank, sondern auch der benötigte Speicherplatz ist geringer bei zusammengefasster Ausführung mehrerer Bibliografien.

Weiterhin wird anhand der Tabelle die Verteilung in den Datenbanktabellen auf die einzelnen Bibliografien erkennbar.
Während für die *VLDB Conference* im Vergleich zu *TODS* mehr als die dreifache Anzahl an Publikationsdaten extrahiert wurde (2.145 zu 669), so wurden dennoch etwas weniger Adressen ermittelt (629 zu 676). Bei älteren Publikationen fehlen im Allgemeinen häufig nähere Angaben zu Autoren, weil eine nachträgliche Digitalisierung aufwändig ist. Da beide Bibliografien jedoch etwa seit dem gleichen Zeitraum bestehen (VLDB Conference seit 1975, TODS seit 1976), wird das Alter als Grund ausgeschlossen. Der Extraktionsalgorithmus war für beide Durchführungen der selbe. So dass entweder dieser Algorithmus bei den Daten von *VLDB Conference* weniger gut arbeitet oder bei dieser Bibliografie fehlen häufiger Angaben zu den Autoren. Eine Untersuchung zu diesem Punkt ergab, dass für *VLDB Conference* etwa 56% (2.298) der Autoren ohne Angaben gespeichert wurden, bei *TODS* lediglich knapp 2% (24). Die Datenqualität bei der Bibliografie *VLDB Conference* ist demnach geringer.

In der Arbeit [RT05] wurden nur Publikationen der Jahre 1994 bis einschließlich 2003 betrachtet, obwohl einige Bibliografien viel weiter in die Vergangenheit reichen. Zudem wurden für diese Publikationen weitere Einschränkungen vorgegeben, damit sich der Aufwand verringert.

Mit dem hier vorgestellten Programm konnten die gesamten, zum Ausführungszeitpunkt[36] aktuellen Daten aller zugänglichen Publikationen der angegebenen Bibliografien gewonnen werden. Die Daten erstrecken sich über die Jahre 1969 bis 2008. Abbildung 17 zeigt einen Vergleich der jeweiligen Anzahl an untersuchten Publikationen, auf der linken Seite nach den Daten der Arbeit [RT05], im rechten Diagramm aktuelle Ergebnisse des entwickelten Programms. Die Zeitachse des Diagramms für die aktuellen Daten wurde für den Vergleich an die Zeitachse nach [RT05] angepasst und damit auf die Jahre 1994 bis 2003 beschränkt.

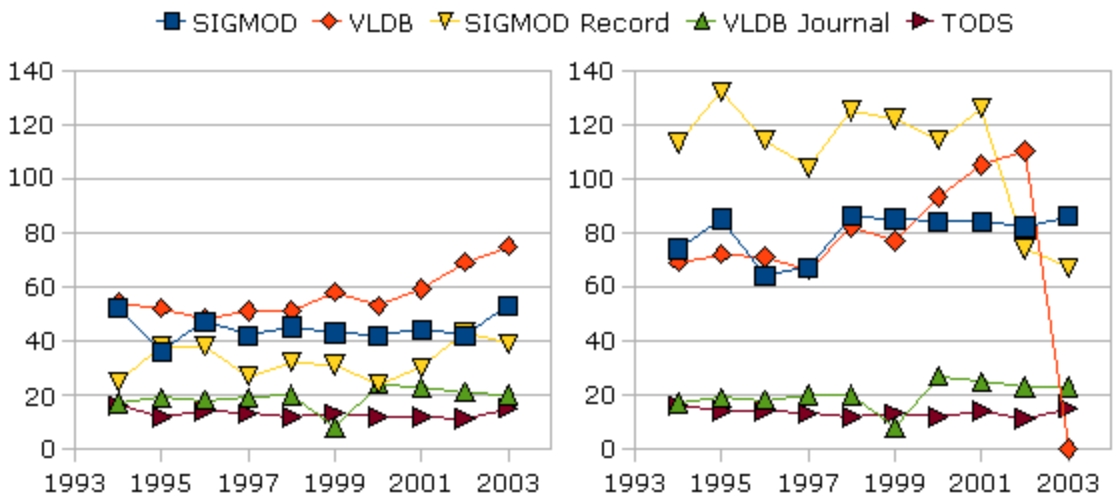


Abbildung 17: Anzahlen untersuchter Publikationen, links nach [RT05]

An der Abbildung 17 werden die voran besprochenen Unterschiede erkennbar. Für *VLDB Journal* und *TODS* sind im Allgemeinen nur wenige Publikationen pro Jahrgang verfügbar, so dass im Vergleich nur marginale Unterschiede festgestellt werden können. Für die restlichen drei Bibliografien sind diese jedoch bedeutend größer. Dass bei der *VLDB Conference* in dieser Arbeit für das Jahr 2003 keine Daten gesammelt wurden, liegt daran, dass dieser Jahrgang im ACM-Archiv nicht vorgehalten wird. Eine Datenextraktion von und damit Abgleich mit weiteren Quellen ist demnach als ein weiterer Schritt denkbar.

[36] 08.01.2007

6.3 Adressinformationen

Die Untersuchung von Quantität und Qualität erhaltener Adressinformationen ist Gegenstand dieses Kapitels.

In Abbildung 18 wird die prozentuale Verteilung der Genauigkeit der erhaltenen Adressinformationen für die jeweiligen Bibliografien als auch deren Zusammenfassung in einer Datenbank aufgezeigt. Die Genauigkeit wird durch die Kenngröße *locationAccuracy* angegeben, die der Google-Maps-Dienst in seinen Antworten mitliefert. In der zugehörigen API-Beschreibung[37] werden die Werte näher erläutert. Je höher der Wert, desto ausführlicher sind die Ergebnisse. Eine 1 bedeutet, dass nur die Angabe des Staates zurückgeliefert wurde, bei dem höchsten Wert 8 enthalten die Ergebnisse Angaben bis hin zur Hausnummer. Das bedeutet gleichzeitig, dass alle Ergebnisse mindestens die Angabe eines Staates beinhalten, laut Aufgabenstellung (Kapitel 1.2) der Schwerpunkt der Informationsextraktion.

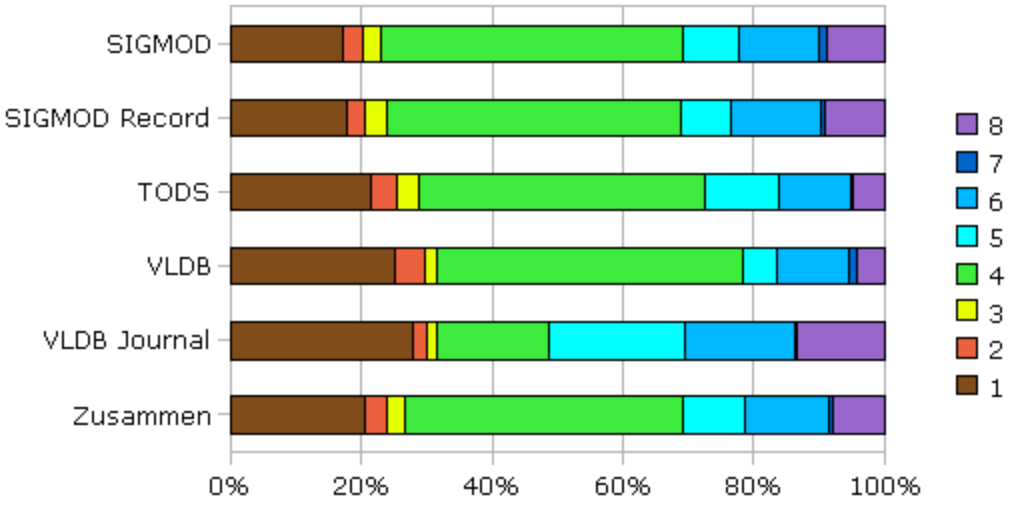


Abbildung 18: Verteilung der Adressinformations-Genauigkeiten

Aus der Abbildung 18 wird ersichtlich, dass die Adressen größtenteils eine Genauigkeit von 4, 1 und 6 besitzen. Die fünf Bibliografien zusammengefasst, bedeutet dies, dass bei etwa 43% Angaben zum Staat, eventuellen Bundesländern oder Ähnlichem bis hin zum Stadtnamen bekannt sind. Zu circa 21% ist nur der Name des Staates vorhanden, bei rund 13% noch zusätzlich zum Stadtnamen die Postleitzahl und ein

[37] code.google.com/apis/maps/documentation/reference.html, 15.01.2008

Straßenname. Die Ergebnisse der Informationsextraktion in Bezug auf Adressen gehen damit weit über die Aufgabenstellung hinaus, zu rund 79% der Ergebnisse sind mehr Angaben als nur der Staatsname verfügbar.
Sämtliche Daten enthalten Koordinaten nach WGS84 [Nat00], also Höhen- und Breitenangaben. Dabei können die Koordinaten für ein Land die gleichen wie zum Beispiel für eine Stadt sein.

Zu 481 der 4.382 verschiedenen Ortsangaben über alle Bibliografien wurden keine Adressangaben gefunden, das entspricht rund 11%. Diesbezüglich ist es sinnvoll, weitere Dienste beziehungsweise Methoden zur Informationsextraktion einzubeziehen, wie im Kapitel 4 beschrieben. Damit könnten für die hier nicht analysierbaren Angaben weitere Adressen bezogen und die vorhandenen verbessert beziehungsweise überprüft werden. Da der Google-Maps-Dienst allerdings im Gegensatz zu den anderen Diensten und Methoden gute Extraktionsergebnisse liefert, wird eine Ergebnisaufwertung mit erheblichem Aufwand verbunden sein.

6.3.1 Evaluation

In diesem Kapitel werden die erhaltenen Adressinformationen untersucht, um eine Bewertung in Bezug auf die Anwendbarkeit durchzuführen. Diese Evaluation erfolgt mit der in dieser Arbeit aufgebauten Datenbank, die die Ergebnisse zu allen hier behandelten fünf Bibliografien zusammenfasst.

Zunächst soll die Aufgabenangemessenheit des entwickelten Aufteilungsalgorithmus betrachtet werden. Der Schritt der Aufteilung von originalen Adressangaben, wenn mehrere Teile enthalten sind, wurde in Kapitel 5.5.1 beschrieben. Aufgrund der Menge an Angaben wurde bei der Betrachtung stichprobenartig vorgegangen. 200 der aufzuteilenden Angaben wurden zufällig ausgewählt und deren Aufteilung überprüft. Die Auswertungsergebnisse werden im Folgenden näher beschrieben.

- 168 der Adressen wurden korrekt aufgeteilt, wie zum Beispiel
  `AT&T Bell Laboratories and IBM Almaden Research Center and Stanford University`
  in die Teile

  - `AT&T Bell Laboratories`
  - `IBM Almaden Research Center`
  - `Stanford University`

- 31 Adressen wurden falsch aufgeteilt. Hierfür ist eine Anpassung des Algorithmus notwendig. Die Probleme hier stellen sich vielseitig dar, so dass eine Algorithmusanpassung nicht angemessen umsetzbar sein kann.

  - So wurde zum Beispiel
    `Cadlab (Coop. Uni.Paderborn & Siemens Nixdorf), Bahnhofstr, 32, 33102 Paderborn, Germany`
    anhand des Et-Zeichens gewissermaßen falsch getrennt, welches in vielen anderen Fällen an Stelle des (hier englischen) Bindeworts *and* eingesetzt wird.
  - Das Beispiel
    `Institute for Advanced Computer Studies and Department of Computer Science, University of Maryland, College Park`
    wurde getrennt, obwohl das angegebene *Institute* als auch das *Department* von einer Universität stammen.
  - `Department of Computer Science and Automation, Indian Institute of Science, Bangalore, India`
    wurde getrennt, da noch kein entsprechender Eintrag in der manuell erstellten Liste mit Einrichtungstypen mit Bindewörtern enthalten war, siehe dazu Kapitel 5.5.1. Eine vollkommene Abdeckung solcher Angaben ist nahezu unmöglich, vor allem wenn neue Daten analysiert werden sollen.

Die hier dargestellten Probleme beeinflussen die auf den Aufteilungsalgorithmus folgende Informationsextraktion nur wenig. Einerseits stellen sie einen geringen Prozentsatz dar. Weiterhin verringert sich dieser im Verhältnis zur Gesamtzahl an Adressangaben weiter, so dass die Anzahl falsch aufgeteilter Adressen vernachlässigbar klein ist. Andererseits werden in der Regel keine Postanschriften geteilt, so dass die Informationsextraktion dennoch umfassende Ergebnisse liefert.

- 1 Adresse wurde fälschlicherweise nicht geteilt. Bei
  `Bell Labs, Lucent Technologies`
  kann kein geeignetes Kriterium zur Erkennung mehrerer Angaben gefunden werden.

Im Weiteren soll überprüft werden, wie sich mit dem in Kapitel 5.5.1 einführend beschriebenen Parameter *locationValidity* die Aussagekraft hinsichtlich der Korrektheit der Adressen abschätzen lässt. Die Tabelle 9 zeigt zunächst die Verteilung der Werte hierfür, in der x für *locationValidity* steht.

locationValidity x	**numerisch**	**prozentual**
$0.0 <= x < 0.1$	8	0.2
$0.1 <= x < 0.2$	128	3.3
$0.2 <= x < 0.3$	40	1
$0.3 <= x < 0.4$	43	1.1
$0.4 <= x < 0.5$	44	1.1
$0.5 <= x < 0.6$	59	1.5
$0.6 <= x < 0.7$	277	7.1
$0.7 <= x < 0.8$	478	12.3
$0.8 <= x < 0.9$	1141	29.2
$0.9 <= x <= 1.0$	1683	43.1

Tabelle 9: Verteilungsübersicht *locationValidity*

Mittels der Tabelle wird erkennbar, dass annähernd 85% der erhaltenen Adressinformationen mindestens eine Aussagekraft von 0.7 vorweisen. Knapp die Hälfte der Angaben erreicht einen Wert von wenigstens 0.9. Wie in Kapitel 5.5 beschrieben, wurden bei den entsprechenden Adressangaben in einer frühen Phase der Informationsextraktion Adressinformationen erhalten beziehungsweise der *locationValidity*-Wert entsprechend gut gewichtet. Somit ist zu erwarten, dass diese Informationen gute Ergebnisse darstellen.

Im Anschluss soll eine Beurteilung der Aussagekraft erfolgen. Hierfür wurden wiederum zahlreiche Adressinformationen untersucht, um deren *locationValidity*-Wert festzustellen und die Korrektheit der jeweiligen Adressinformation zu überprüfen. Die Ergebnisse werden in der Tabelle 10 gezeigt.

Für die beispielhaften Überprüfungen von erhaltenen Informationen auf Kontinent-, Staat- und Stadtebene wurden jeweils 200 zufällige Adressinformationen gewählt, so dass im Endeffekt 600 Adressinformationen (mit eventuellen Überschneidungen) untersucht worden sind.
Da viele Adressinformationen einen hohen *locationValidity*-Wert besitzen (siehe Tabelle 9), wurden mehr Adressinformationen mit einem Wert zwischen 0.9 und 1.0 überprüft als zum Beispiel mit einem Wert unter 0.1. Die Anzahl der jeweils überprüften Adressinformationen wird in der Spalte '#' der Tabelle 10 dargelegt.

locationValidity x	#	**Kontinent**		**Staat**		**Stadt**	
		korrekt	falsch	korrekt	falsch	korrekt	falsch
$0.0 <= x < 0.1$	6	5	1	3	3	2	4
$0.1 <= x < 0.2$	14	12	2	12	2	5	9
$0.2 <= x < 0.3$	10	8	2	8	2	4	6
$0.3 <= x < 0.4$	10	10	0	9	1	5	5
$0.4 <= x < 0.5$	10	9	1	8	2	7	3
$0.5 <= x < 0.6$	10	10	0	8	2	8	2
$0.6 <= x < 0.7$	20	19	1	19	1	15	5
$0.7 <= x < 0.8$	30	29	1	29	1	28	2
$0.8 <= x < 0.9$	40	40	0	38	2	39	1
$0.9 <= x <= 1.0$	50	50	0	49	1	49	1
Summe	200	192	8	183	17	162	38

Tabelle 10: *locationValidity* im Vergleich zur Korrektheit von Adressangaben

Festzustellen ist, dass ein hoher Anteil der erhaltenen Adressinformationen korrekt ist. Allerdings nimmt die Korrektheit mit steigendem Detailgrad der Informationen ab.
Der *locationValidity*-Wert eignet sich für Kontinent-Angaben nur bedingt, die Korrektheit dieser liegt immer bei mindestens 80%. Mit steigendem Detailgrad der Adressen wird der *locationValidity*-Wert allerdings wichtiger. Für Angaben auf Stadt-Ebene sollte dieser Wert nicht kleiner als 0.5 sein, da Angaben mit Werten kleiner 0.5 eine maximale Korrektheit von 70% erreichen. Bei der Betrachtung von Staaten ist eine erhöhte Ungenauigkeit erst bei *locationValidity*-Werten kleiner 0.1 zu rechnen.

Anschließend an die Beurteilung der Aussagekraft des Parameters *locationValidity* erfolgt eine Untersuchung, inwiefern E-Mail-Adressen die Ergebnisse beeinflussen konnten.
Tabelle 8 zeigt, dass für die hier behandelten fünf Bibliografien 13.303 verschiedene Autor-Angaben erhalten wurden. Darin enthalten sind ca. 2% (311) Autoren mit E-Mail-Angaben, die einzig in den ACM-Abstract-Seiten zur Bibliografie *VLDB Journal* aufgefunden wurden. 100 dieser E-Mail-Adressen wurden in Bezug auf die erhaltenen Ortsangaben manuell analysiert. Die Ergebnisse stellen sich wie folgt dar:

- Durch 49 E-Mail-Adressen erfolgte eine positive Gewichtung des *locationValidity*-Wertes, da die TLD einem Land zugeordnet werden konnte und dieses dem Land der erhaltenen Adresse entsprach.
- Bei 3 Adressen erfolgte eine negative Gewichtung, da sich die Länder aus der E-Mail-Adresse und den Adressinformationen nicht glichen.
- Bei 48 Adressen fand keine Gewichtung statt, da der jeweiligen TLD kein Land zugeordnet werden konnte. In der Regel waren dies die TLDs *edu* und *com*.

Eine Gewichtung des *locationValidity*-Wertes mittels E-Mail-Angaben ist der Analyse zufolge sinnvoll, da das zugrunde liegende Ergebnis verifiziert werden kann. Allerdings existiert die Beschränkung, dass nur selten E-Mail-Angaben vorgehalten werden. Weiterhin kann etwa der Hälfte dieser Angaben keine nützliche Information entnommen werden, wodurch keine Gewichtung möglich ist.
Außerdem bezieht sich diese Gewichtung auf das Land und verifiziert damit zusätzlich auch den Kontinent. Falls detailliertere Angaben wie Stadt- bis hin zu Straßennamen vorliegen, kann die Gewichtung in diesem Sinn sogar fehlerhaft sein. Eine positive Gewichtung aufgrund des gleichen Landes in E-Mail-Angabe und erhaltener Adressinformation bedeutet nicht, dass Angaben zur Stadt korrekt sein müssen. Somit sollte eine gemeinsame Betrachtung von *locationValidity* und Adressinformationen nur bis zur Ebene von Staaten geschehen, detailliertere Adressinformationen und der *locationValidity*-Wert stehen nicht miteinander in Verbindung.

6.3.2 Überblick

In diesem Kapitel soll ein Überblick über die Verteilung der erhaltenen Adressinformationen gegeben als auch ein Vergleich dieser Verteilung mit der wissenschaftlichen Publikation [RT05] gezogen werden. Damit sollen die Korrektheit des hier entwickelten Programms aufgezeigt als auch Abweichungen erläutert werden.

Die Verteilung der erhaltenen Adressinformationen für die behandelten Bibliografien über die jeweiligen Kontinente, wird in Tabelle 11 gezeigt.

Kontinent	**Anzahl**	**Kontinent**	**Anzahl**
North America	2299	Oceania	80
Europe	1074	South America	41
Asia	400	Africa	7

Tabelle 11: Adress-Verteilung über Kontinente

In der Tabelle 12 werden die ersten 20 von 63 Staaten mit den meisten Publikationen gezeigt. Dabei werden oftmals Publikationen für mehrere Staaten gezählt. Einerseits können verschiedene Autoren einer Publikation aus unterschiedlichen Staaten stammen, andererseits sind mehrere verschiedene Angaben zu einem Autor möglich.

Staat	**Anzahl**	**Staat**	**Anzahl**
United States	12280	United Kingdom	197
Germany	1179	Turkey	192
Canada	1011	Japan	191
Italy	452	Greece	177
India	411	China	147
France	327	Netherlands	129
Switzerland	313	Israel	110
Hong Kong S.A.R., China	306	South Korea	109
Singapore	287	Belgium	107
Australia	268	Denmark	95

Tabelle 12: Publikationen pro Staat

In der Arbeit [RT05] wurden Zitierungen pro Staat betrachtet, wobei auch die Anzahl an Publikationen angegeben wurde. Dieser Inhalt der entsprechenden Tabelle 5 aus [RT05] wird in Tabelle 13 nochmals aufgezeigt. Die Sortierung erfolgt hier allerdings anhand der Anzahlen an Publikationen, um eine Vergleichsmöglichkeit zu schaffen.

Staat	**Anzahl**	**Staat**	**Anzahl**
United States	567	Greece	12
Germany	64	Switzerland	8
Canada	34	Denmark	7
France	29	Israel	6
Italy	22	Japan	6

Tabelle 13: Publikationen pro Staat nach [RT05]

Die ersten drei Staaten mit den meisten Publikationen sind in beiden obigen Tabellen die selben. Die folgenden Staaten unterscheiden sich in der Platzierung wenig bis relativ stark. Zum einen kann dies daran liegen, dass in [RT05] Einschränkungen gemacht wurden, um nicht alle Publikationen sondern nur die wichtigsten zu betrachten. Zum anderen wurde in [RT05] pro Veröffentlichung nur eine Adresse berücksichtigt. Selten auftretende Staaten werden deshalb wahrscheinlich weniger häufig erfasst.

6.4 Einrichtungstypen

Dieses Kapitel bewertet den Extraktionsalgorithmus für Einrichtungstypen als auch dessen Ergebnisse.

Zunächst wird der Extraktionsalgorithmus näher untersucht, der im Kapitel 5.5.3 beschrieben wurde. Da ein Abgleich von einzelnen Wörtern aus der zu untersuchenden Zeichenkette mit denen in einer manuell erstellten Liste stattfindet, sind nur die Ergebnisse *gefunden* und *nicht gefunden* möglich. Die Liste weist einem Fund einen bestimmten Einrichtungstyp zu.
Es existieren zwei Möglichkeiten der Umsetzung des Extraktionsalgorithmus:

- Einerseits kann der Vergleich so durchgeführt werden, dass *vollständige Wörter* aus der Liste *exakt* in der zu untersuchenden Zeichenkette vorkommen müssen. Mit dem Listeneintrag `Universität` kann dann nicht `Fernuniversität` in einer Zeichenkette aufgefunden werden. Dieses Wort muss demnach zusätzlich der Liste hinzugefügt werden. Die Liste ist somit umfangreich und eine manuelle Erstellung aufwändig, da viele Besonderheiten auftreten können. Wörter (Substantive) können aus verschiedenen Gründen vollständig mit Klein- oder auch vollständig mit Großbuchstaben oder „normal“ geschrieben werden, Abkürzungen können in unterschiedlichen Formen auftreten, außerdem werden unterschiedliche Sprachen genutzt.

- Andererseits besteht die Möglichkeit, dass ein Wort der Vergleichsliste nur *irgendwie* in der zu untersuchenden Zeichenkette *vorkommen* muss, so dass zum Beispiel mit dem Listeneintrag `uni` auch `Fernuniversität` gefunden wird. Hierbei kann auf die Untersuchung von Groß- und Kleinschreibung verzichtet werden. Allerdings besteht die Problemstellung, dass die Liste mit relativ eindeutigen Wörtern zu füllen ist. `universität` ist demnach `uni` vorzuziehen, da letzteres zum Beispiel auch in `Juni` vorkommt. Obwohl in den hier erzielten Ergebnissen nicht vorgefunden, so könnte allerdings auch das Wort `Universitätsstraße` in der zu untersuchenden Zeichenkette vorkommen. Die zu erstellende Liste ist folglich weniger umfangreich, eine manuelle Listenerstellung muss aufgrund der beschriebenen Probleme sorgfältig durchgeführt werden.

Um den Aufwand der Listenerstellung zu reduzieren, wurde in dieser Arbeit der zweite Ansatz gewählt.

Die Ergebnisse der Extraktion von Einrichtungstypen sind von den manuell erstellten Listeneinträgen abhängig. Somit ist es möglich, nur bestimmte Typen abzufragen, die beispielsweise eine Bildungseinrichtung (Universität, Schule, etc.) identifizieren. Es können aber ebenso Firmennamen oder -abkürzungen (Microsoft, IBM, etc.) aufgelistet werden.
In dieser Arbeit wurden Firmenkennzeichnungen nicht berücksichtigt. Statt dessen wurde Wert darauf gelegt, eine umfangreiche Liste von allgemeinen Einrichtungstypkennzeichnungen für verschiedene Sprachen zu erstellen.

Diese Einrichtungstypkennzeichnungen werden jeweils auf einen Typ abgebildet, insgesamt wurden während der Entwicklung empirisch 18 Typen definiert. Eine Übersicht über die Anzahl erkannter Typen für die hier behandelten fünf Bibliografien gibt Tabelle 14.

Typ	**Anzahl**	**Typ**	**Anzahl**
academy	1169	chair	11
association	291	committee	6
department	245	bank	5
institute	233	agency	3
laboratory	226	library	2
center	155	foundation	1
division	60		

Tabelle 14: Häufigkeitsverteilung für Einrichtungstypen

Für die Typen *consortium*, *forum*, *initiative*, *faculty* und *group* wurden keine Angaben gefunden.
Die Ergebnisse zeigen, dass der Typ *academy* in den Angaben zu Autoren überwiegend enthalten war. Dieser Typ fasst die Angaben zu (Berufs-)Akademie, College, (Fern-)Universität, Hochschule, Kollegium und School zusammen, er beschreibt somit im Allgemeinen Bildungseinrichtungen.

7 Schlussbetrachtung

Dieses Kapitel fasst die Arbeit in ihren verschiedenen Ausprägungen zusammen, um anschließend darauf aufbauend Erweiterungs- als auch Verbesserungsmöglichkeiten zu erörtern.

7.1 Zusammenfassung

Ziel dieser Arbeit war es, eine Datenbank aufzubauen, die Daten zu Forschungseinrichtungen beinhaltet. Diese Daten umfassen Adressen als auch Typangaben der Einrichtungen. In diesem Rahmen sollten Daten aus Bibliografiedatenbanken des Web genutzt werden.
Zunächst wurde ein Überblick über mögliche Datenquellen gegeben. Anschließend wurden verschiedene Datenquellen näher betrachtet, die aufgrund unterschiedlicher Faktoren, wie die Qualität vorhandener Daten, von besonderem Interesse waren. Die Untersuchung der Datenquellen erfolgte im Hinblick auf deren Methoden zur Datenbereitstellung, ob die benötigten Daten immer vorliegen und inwiefern eine Zugangsbeschränkung zu den Daten besteht. Hier wurde das ACM Portal als primäre Datenquelle für diese Arbeit festgelegt.
Im Anschluss erfolgte eine Darstellung von Möglichkeiten zur Datenextraktion aus verschiedenen Dateiformaten mittels Wrapper-Technologie. Die Betrachtung dieser Möglichkeiten ist für den Implementierungsentwurf notwendig, damit das Programm mit unterschiedlichen Datenquellen umgehen kann. Einige Beispiele von extrahierten Daten wurden aufgeführt, um die Datenheterogenität aufzuzeigen.
Mit dieser Datengrundlage wurden verschiedenartige Möglichkeiten in Bezug auf die Extraktion von Adress- und Typinformationen zu Forschungseinrichtungen aufgezeigt. Unterschiedliche Verzeichnisarten und Dienste des Web wurden auf ihren Nutzen für die Informationsextraktion hin untersucht und beurteilt.

Auf den theoretischen Grundlagen der Datenquellen, Datenextraktion und Informationsextraktion basierend erfolgte die praktische Umsetzung. Hierzu wurden ausgewählte Implementierungsdetails des entwickelten Programms besprochen. Dieses ist in der Lage, mittels weniger Start-URLs zu vielen wissenschaftlichen Publikationen die notwendigen Daten zu beziehen, damit die hier benötigten Informationen

zu extrahieren und in einer Datenbank zu speichern. Hierbei werden ebenso die erhaltenen Daten zu den Publikationen gespeichert, um eine umfassende Auswertung zu gewährleisten.
Ein abschließendes Kapitel gibt einen Überblick über Quantität als auch Qualität der Daten. Hierzu wurden Vergleiche zu einer für diese Arbeit grundlegenden wissenschaftlichen Veröffentlichung gezogen, eine Auswertung über den Detailgrad der Informationen durchgeführt als auch die eingesetzten Informationsextraktionsmethoden evaluiert.

7.2 Ausblick

Mit dem entwickelten Programm ist ein Einsatz in Bezug auf die Informationsextraktion zu wissenschaftlichen Publikationen möglich. Mittels der Wrapper-Technologie können Daten aus ACM-Abstract-Seiten und DBLP-XML-Dateien extrahiert werden. Möchte man die Datenextraktion erweitern, lassen sich weitere Wrapper für verschiedene Datenquellen und damit unterschiedliche Formate für das Programm entwickeln. Insofern von der Abteilung Datenbanken der Universität Leipzig gewünscht, können zum Beispiel Daten von Scopus käuflich erworben und auch hierfür ein Wrapper erstellt werden.

Die Informationsextraktion ist abhängig vom Google-Maps-Dienst. Sollte sich die Schnittstelle dazu ändern, ist eine Anpassung des Programms notwendig. Mit der Einbindung und damit Kombination mit weiteren Dienste oder anderen Methoden mit ähnlicher Funktionalität wird die Abhängigkeit von nur einem Dienst beseitigt. Weiterhin ist damit theoretisch eine Verbesserung der Ergebnisquantität und -qualität möglich. Dies ist allerdings mit hohem Aufwand verbunden. Einerseits müssen Ergebnisse verschiedener Dienste miteinander verglichen werden, andererseits existiert nichts Gleichwertiges zum Google-Maps-Dienst. Andere Dienste und Methoden sind, wie erörtert, mit umfangreichen Problemen verbunden.

Im Zuge der Informationsextraktion wurden E-Mail-Adressen aus zu untersuchenden Zeichenketten extrahiert. Diese könnten genutzt werden, um entsprechende Homepages der Einrichtungen zu identifizieren und somit eine Analyse dieser Internetseiten durchzuführen, um (zusätzliche) benötigte Informationen zu Orten und Einrich-

tungstypen zu extrahieren. Der Aufbau verschiedener Internetseiten folgt allerdings keinem einheitlichen Muster. Zum Beispiel werden Adressen unter einem Menüpunkt *Impressum* oder eventuell *Kontakt* oder Ähnlichem vorgehalten. Außerdem besitzen viele Einrichtungen mehrere Standorte.
Eine automatische Analyse der Homepages ist demnach aufwändig, eine manuelle Analyse erfordert allerdings aufgrund der großen Anzahl einen hohen zeitlichen Aufwand.

Die Informationsextraktion in Bezug auf die Einrichtungstypen erfolgt anhand einer manuell erstellten Liste mit Einrichtungsidentifikatoren. Diese Liste enthält allgemeine Identifikatoren wie *fakultät*, *laboratory* und *politecnico*. Für eine erweiterte Informationsextraktion können auch Eigennamen in die Liste eingetragen werden. So lässt sich zum Beispiel dem Identifikator *IBM* der Typ *corporation* zuordnen. Außerdem kann eine Abstufung durchgeführt werden, indem beispielsweise der Typ *association, research* für *AT&T Research* und *association* für *AT&T* gespeichert wird.
Diese Liste kann demnach den Bedürfnissen angepasst werden, auch eine numerische Typkodierung kann hier erfolgen.

Für die Daten- und Informationsextraktion kann es hilfreich sein, wenn zukünftig Autoren bei der Veröffentlichung von wissenschaftlichen Arbeiten im Web ihre Zugehörigkeitsdaten transparent in einem semantischen System speichern und die Bibliografieanbieter ihre Übersichten mit diesem System anbieten. Über das Tagging kann somit einfach zwischen den verschiedenen Daten unterschieden werden. Je nachdem, wie detailliert dieses Tagging erfolgt, kann auch die Informationsextraktion davon profitieren, wenn zum Beispiel Staat und Stadt voneinander getrennt angegeben werden.

Mit dem in dieser Arbeit entwickelten Programm wurde eine Datenbank für Forschungsinstitute erstellt, die ausschließlich Informationen zu Orten und Einrichtungstypen und Daten zu wissenschaftlichen Publikationen enthält. Die Datensätze lassen sich in ein Data-Warehouse integrieren, um darauf basierend weitere Analysen zu ermöglichen. Mit zusätzlichen Zitierungsdaten dieser Publikationen in einem Data-Warehouse kann somit eine automatische Zitierungsanalyse erfolgen. Zu diesem Punkt ist möglich, das entwickelte Programm insofern zu erweitern, dass ebenso

Zitierungsdaten aus den Datenquellen extrahiert werden.

Ziel einer Auswertung der mit dieser Arbeit erhaltenen Daten kann sein, den beruflichen Lebensweg über Publikations- und Adressinformationen eines Autors nachzuvollziehen. Hierbei ist allerdings eine Duplikaterkennung von Autoren nötig, je nachdem in welcher Art die Namen angegeben wurden, zum Beispiel mit oder ohne abgekürzten Vornamen. Außerdem kann es sein, dass in der hier erstellten Datenbank ein Name zu mehreren Publikationen angegeben wurde. Das bedeutet allerdings nicht hundertprozentig, dass der Autor immer der selbe ist, denn mehrere Autoren können den selben Namen tragen.
Eine Duplikaterkennung kann ebenfalls bei den Informationen zu Einrichtungen sinnvoll sein. Beispielsweise über einen gleichen Einrichtungstyp, zu den Einrichtungen gehörige geografische Koordinaten und eventuell zusätzlich einem anzugebenden Radius könnten zwei unterschiedlich benannte Einrichtungen wie *Universität Leipzig* und *University of Leipzig* als die selbe Einrichtung identifiziert werden.

Letztendlich können die in dieser Arbeit erhaltenen Daten genutzt werden, um die Standorte einzelner Einrichtungen und deren Anzahl an Publikationen zu visualisieren. Ein Mashup mittels Google Maps bietet sich hierfür beispielsweise an.

Glossar

API Application Programming Interface - Programmierschnittstelle für Zugriff auf verschiedene Funktionalitäten

Cookie Textdateien zum Speichern von Einstellungen und Informationen für einen Browser

CSV Comma Separated Values - Strukturiertes Format zum Speichern von Daten, getrennt durch ein beliebiges Zeichen (z.B. Komma)

DOM Document Object Model - Spezifikation einer Programmierschnittstelle für den Zugriff auf semistrukturierte Dokumente (HTML, XML)

DTD Document Type Definition - Festlegung eines Regelsatzes zur Typdeklaration eines Dokuments

Geokodierung Versehen von Medien (Text, Bilder, Videos, etc.) mit geografischen Koordinaten

GIS Geografisches Informationssystem - System zum Umgang (Erfassung, Analyse, Präsentation, etc.) mit geografischen Daten

Hostname alphanumerischer Teil einer Identifizierungszeichenkette eines Rechners in einem Netzwerk

IP-Adresse Internet-Protocol-Adresse - Adresse zur Identifizierung eines (IP-fähigen) Gerätes in einem IP-Netzwerk

JDBC Java-API zu relationalen Datenbanken; zur Ausführung von SQL-Anweisungen

ISO International Organization for Standardization - internationale Vereinigung zur Erarbeitung von Normen

JSON JavaScript Object Notation - Datenformat zum Datenaustausch zwischen Maschinen

HTML Hypertext Markup Language - Beschreibungssprache für Web-Seiten

HTTP Hypertext Transfer Protocol - Datenübertragungsprotokoll in einem Netzwerk

KML Keyhole Markup Language - spezielles Dokumentenformat zum Datenaustausch

Metadatum Datum zur näheren Beschreibung anderer Daten

OCR Optical Character Recognition - digitale Erschließung von gedrucktem Text mit Zeichenerkennung

OLAP Online Analytical Processing - Hypothesen durch Anfragen an ein Analysesystem verifizieren

PDF Portable Document Format - Dateiformat zur plattformübergreifenden, einheitlichen Darstellung

RDF Resource Description Framework - System zur Beschreibung von Ressourcen; ähnlich zu Metadaten

SAX Simple API for XML - API zur sequentiellen, ereignisgesteuerten XML-Dokumenten-Verarbeitung

SOAP XML-basiertes Protokoll zum Aufbau von Nachrichten; für Kommunikation mit einem Web Service

SQL Structured Query Language - Abfragesprache für relationale Datenbanken

TLD Top Level Domain - höchste Ebene in der Adressierungshierarchie eines Rechners in einem Netzwerk

UDDI Universal Description, Discovery and Integration - Verzeichnisdienst zum Registrieren und Auffinden von Web Services

URI Uniform Resource Identifier - eindeutiger Identifikator einer abstrakten oder physischen Ressource

URL Uniform Resource Locator - eindeutiger Identifikator einer physischen Ressource (Unterbegriff von URI)

W3C World Wide Web Consortium - Vereinigung von Unternehmen, sie sich mit der Erstellung von Standardisierungen und Leitfäden für Techniken des World Wide Web beschäftigt

Web Service Software-System zur Maschine-Maschine-Kommunikation über ein Netzwerk

WSDL Web Service Description Language - Sprache für Schnittstellendefinition eines Web Service

XML Extensible Markup Language - Auszeichnungssprache zur Datendarstellung in einer selbstdefinierbaren, hierarchischen Struktur

Abbildungsverzeichnis

Tabellenverzeichnis

Literatur

[apa07] The Apache Software Foundation: *Apache Lucene*. http://lucene.apache.org/java/docs/index.html (07.12.2007), 2007

[BHM+04] Booth, David ; Haas, Hugo ; McCabe, Francis ; Newcomer, Eric ; Champion, Michael ; Ferris, Chris ; Orchard, David: Web Services Architecture / W3C Web Services Architecture Working Group. Version: 2004. http://www.w3.org/TR/ws-arch/wsa.pdf. 2004. – Forschungsbericht

[BPZ01] Bergmark, Donna ; Phempoonpanich, Paradee ; Zhao, Shumin: Scraping the ACM Digital Library. In: *ACM SIGIR Forum* 35 (2001), Nr. 2, S. 1–7

[BRS03] Badach, Anatol ; Rieger, Sebastian ; Schmauch, Matthias: *Web-Technologien*. München, Wien : Carl Hanser Verlag, 2003. – ISBN 3–446–22149–2

[Eik99] Eikvil, Line: Information Extraction from World Wide Web - A Survey / Norweigan Computing Center. Version: 1999. http://citeseer.ist.psu.edu/eikvil99information.html. 1999 (945). – Forschungsbericht

[Els04] Elsevier B.V.: *SCOPUS - Terms & Conditions*. http://www.scopus.com/scopus/standard/termsandconditions.url (11.01.2008), 2004

[Fri98] Friedl, Jeffrey E.: *Reguläre Ausdrücke*. Köln : O'Reilly Verlag, 1998. – ISBN 3–930673–62–2. – Deutsche Übersetzung von Andreas Karrer

[GBL98] Giles, C. L. ; Bollacker, Kurt ; Lawrence, Steve: CiteSeer: An Automatic Citation Indexing System. In: Witten, Ian (Hrsg.) ; Akscyn, Rob (Hrsg.) ; M., Frank (Hrsg.): *Digital Libraries 98 - The Third ACM Conference on Digital Libraries*. Pittsburgh, PA : ACM Press, 1998. – ISBN 0–89791–9653, S. 89–98

[GHJV95] Gamma, Erich ; Helm, Richard ; Johnson, Ralph ; Vlissides, John: *Design Patterns - Elements of Reusable Object-Oriented Software*. Amsterdam, Netherlands : Addison-Wesley, 1995. – ISBN 0–20163–361–2

[Gün02] Günther, Karsten: *LaTeX GE-PACKT*. 1. Auflage. Bonn : mitp, 2002. – ISBN 3–8266–0785–6

[Goo08] Google: *Über Google Scholar - Was ist Google Scholar?* http://scholar.google.de/intl/de/scholar/about.html (11.01.2008), 2008

[HCF98] Hamilton, Graham ; Cattell, Rick ; Fisher, Maydene: *JDBC - Datenbankzugriff mit Java [Dt. Übers. aus dem Amerikan. von Birgit Krehl...]*. Bonn : Addison-Wesley, 1998. – ISBN 3–8273–1306–6

[HGM+03] Han, Hui ; Giles, C. L. ; Manavoglu, Eren ; Zha, Hongyuan ; Zhang, Zhenyue ; Fox, Edward A.: Automatic document metadata extraction using support vector machines. In: *JCDL '03: Proceedings of the 3rd ACM/IEEE-CS joint conference on Digital libraries*. Washington, DC, USA : IEEE Computer Society, 2003. – ISBN 0–7695–1939–3, S. 37–48

[Kan05] Kanaeva, Zara: Ranking: Google und CiteSeer. In: *Information - Wissenschaft & Praxis* 56 (2005), Nr. 2, S. 87–92. – ISSN 1434–4653

[Koe06] Koester, Bjoern: *FooCA - Web Information Retrieval with Formal Concept Analysis*. Darmstadt : Verlag Allgemeine Wissenschaft - HRW e.K., 2006. – ISBN 3–935924–06–2

[Kor97] Korfhage, Robert R.: *Information Storage and Retrieval*. New York : Robert R. Korfhage, 1997. – ISBN 0–471–14338–3

[Lev65] Levenshtein, Vladimir I.: Binary codes capable of correcting deletions, insertions, and reversals. In: *Doklady Akademii Nauk SSSR* 163 (1965), Nr. 4, S. 845–848

[LN07] Leser, Ulf ; Naumann, Felix: *Informationsintegration - Architekturen und Methoden zur Integration verteilter und heterogener Datenquellen*. 1. Auflage. Heidelberg : dpunkt.verlag GmbH, 2007. – ISBN 978–3–89864–400–6

[MGB+05] Mittelbach, Frank ; Goossens, Michel ; Braams, Johannes ; Carlisle, David ; Rowley, Chris: *Der LaTeX-Begleiter*. 2. Auflage. Pearson Studium, 2005. – ISBN 3–8273–7166–X

[Nat00] National Imagery and Mapping Agency: World Geodetic System 1984 - Its Definition and Relationships with Local Geodetic Systems / National Imagery and Mapping Agency (NIMA). Version: Third Edition, 2000. http://earth-info.nga.mil/GandG/publications/tr8350.2/wgs84fin.pdf. 2000 (8350.2). – Forschungsbericht

[Oes06] Oestereich, Bernd: *Analyse und Design mit UML 2.1 - Objektorientierte Softwareentwicklung*. 8. Auflage. München : Oldenbourg Wissenschaftsverlag GmbH, 2006. – ISBN 978–3–486–57926–0

[pdf06] Adobe Systems Incorporated (Hrsg.): *PDF Reference, sixth edition: Adobe Portable Document Format version 1.7*. Adobe Systems Incorporated, 2006. http://www.adobe.com/devnet/acrobat/pdfs/pdf_reference.pdf

[RK07] Richter, Alexander ; Koch, Michael: *Social Software - Status quo und Zukunft, Technischer Bericht Nr. 2007-01*. www.kooperationssysteme.de/wordpress/uploads/RichterKoch2007.pdf (22.01.2008), Feb. 2007. – Fakultät für Informatik, Universität der Bundeswehr München

[RT05] Rahm, Erhard ; Thor, Andreas: Citation analysis of database publications. In: *SIGMOD Rec.* 34 (2005), Nr. 4, S. 48–53. – ISSN 0163–5808

[RV03] Rahm, Erhard (Hrsg.) ; Vossen, Gottfried (Hrsg.): *Web & Datenbanken - Konzepte, Architekturen, Anwendungen*. Heidelberg : dpunkt.verlag, 2003. – ISBN 3–446–22149–2

[Som04] Sommerville, Ian: *Software Engineering*. Seventh Edition. Boston [u.a.] : Pearson Education — Addison Wesley, 2004. – ISBN 0–321–21026–3

[Wic07] Wick, Marc: *GeoNames*. http://www.geonames.org, 08.12.2007

[Wie92] WIEDERHOLD, Gio: Mediators in the Architecture of Future Information Systems. In: *Computer* 25 (1992), Nr. 3, S. 38–49

[Woh07] WOHLIN, Claes: An analysis of the most cited articles in software engineering journals - 2000. In: *Inf. Softw. Technol.* 49 (2007), Nr. 1, S. 2–11. – ISSN 0950–5849

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

_____

Leipzig, den 31. Januar 2008