Diplomarbeit

im Diplomstudiengang *Informatik*
Technische Universität Berlin
Fakultät für Informatik und Elektrotechnik

Ein webbasiertes System zur Recherche, Auswahl und interaktiver Strukturierung von Literatur

Sebastian Kolbe
Matr. Nr: 164618

05. März 2008

Gutachter:

Prof. Dr. Bettina Berendt (HU-Berlin)
Dr.-Ing. habil. Thomas Santen (TU-Berlin)

Danksagung

Ich möchte mich bei allen Leuten bedanken, die direkt oder indirekt an der Entstehung dieser Arbeit mitgewirkt haben:

- Prof. Dr. Bettina Berendt für Betreuung und Ratschläge.
- Barbara Nusser für ihre Geduld, Motivation, Tipps, Fehlerkorrekturen und natürlich Versorgung.
- Dmitri Naumov für viele gute Hinweise und Fehlerkorrekturen.
- Den Teilnehmern der Evaluation, die durch Vorschläge und Hinweise geholfen haben, das Programm zu verbessern.
- Johann Pachelbel für den Kanon in D.

Kurzzusammenfassung

In dieser Arbeit wird die Entwicklung eines webbasierten Programmes vorgestellt, das wissenschaftliche Literatur sucht, als Ergebnisliste darstellt und nach thematischen Kriterien zu Gruppen ordnet. Ziel dieses Programmes ist es die Literaturrecherche von Studenten und Wissenschaftlern zu verbessern.

Es werden zunächst gängige Literaturdatenbanken vorgestellt mit dem Ziel eine Datenquelle von Zitationsdaten für das Programm zu wählen und eine Übersicht über ähnliche Systeme zu gewinnen. Eine detaillierte Vorstellung von Algorithmen zur Analyse Dokumentenähnlichkeit schließt bibliographische wie auch textuelle Methoden ein. Es wird angenommen, dass eine solche Analyse vergleichbar mit einer direkten inhaltlichen Analyse ist.
Darauf aufbauend werden Anforderungen und technische Grundlagen vorgestellt mit einem Überblick über die in einem webbasierten Programm einsetzbare Technologie, sowohl auf Server- als auch auf Clientseite. Eine konkrete Auswahl dieser Techniken und kurze Vorstellung der eingesetzten Daten schließen das Kapitel ab.
Auf diesen Grundlagen wird eine Software realisiert, die es Nutzern ermöglicht mit Hilfe eines Webbrowsers Literatur zu suchen, auszuwählen und in Gruppen einzuteilen. Dabei wurde besonderer Wert auf ein möglichst einfach zu benutzendes System gelegt. Zusätzlich wurde ein Beispieldatensatz ausgewählt, an dem die interne Arbeitsweise der Software detailliert nachvollzogen werden kann.

Abschließend wird die Software einer Benutzerstudie unterworfen, um die Gebrauchstauglichkeit beurteilen zu können. Gleichzeitig wird damit eine Einschätzung über die Nützlichkeit des Programms für den vorgesehenen Nutzerkreis von Studenten und Wissenschaftlern möglich. Die Ergebnisse dieser Benutzerstudie lassen den Schluss zu, dass die nach inhaltlichen Kriterien vorgestellte Gruppierung, sowie die hier angedachte Präsentation von wissenschaftlicher Literatur für die Zielgruppe Studenten und Wissenschaftler eine sinnvolle Hilfe sein kann.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1 Einleitung

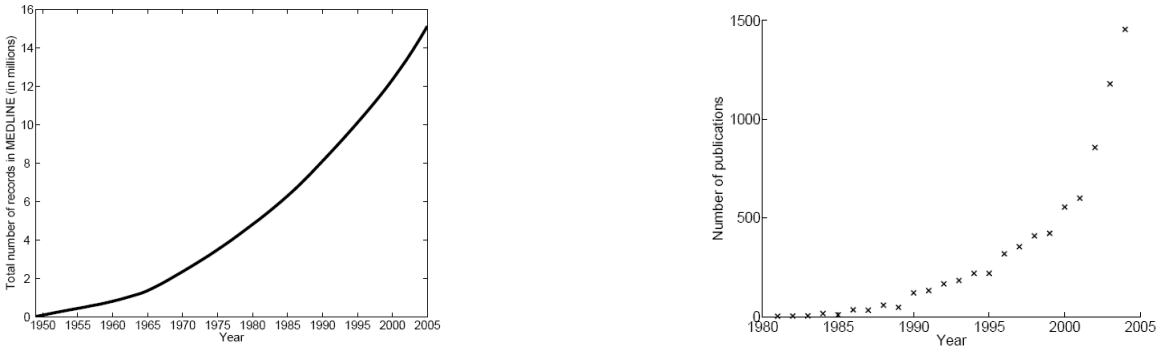
> „**Wissenschaft:** Die Erzeugung und Publikation von Wissen, das bis zum Zeitpunkt der Erkenntnis noch nicht publiziert war. Dazu gehört zwangsläufig die Überwachung und Verwaltung des bereits vorhandenen Wissens, das permanent auf seine Richtigkeit hin überprüft und gegebenenfalls falsifiziert werden muss."
>
> Aus: W. Umstätter: Semiotischer Thesaurus[1]

Die Suche nach geeigneter Literatur ist für Wissenschaftler ein wichtiger Teil der Forschung. Für Studenten gehört die Suche und Auswahl von relevanter Fachliteratur zur Ausbildung und ist spätestens bei der Abschlussarbeit essenziell.
Untersuchungen zeigen, dass nicht nur die Ausbildung zu einer effektiven Literatursuche unzureichend ist, sondern auch die Literatursuche als solche von vielen unterschätzt und nicht effektiv angegangen wird [Pap07].
Deswegen will dieses Projekt Hilfestellung geben mit einem Programm zur Literaturrecherche und -verwaltung.

1.1 Motivation und Zielgruppe des Projekts

Die Probleme der Literaturrecherche zu wissenschaftlichen Quellen sind vielfältig für Studenten und Wissenschaftler. Die Aufgabe, relevante wissenschaftliche Literatur zu einem bestimmten Thema zu finden, ist für viele ein komplexes Problem. Die Auswahl der „falschen" Literatur kostet unter Umständen viel Zeit und kann womöglich zu falschen Schlussfolgerungen führen.
Einige der in Studien aufgezeigten Schwierigkeiten sind auf Mängel in der Ausbildung, unzureichende Zugänglichkeit bzw. fehlende Kenntnis der Existenz von Suchsystemen zurückzuführen oder durch mangelhafte Aufbereitung der Daten entstanden (vgl. [Pap07]). Die schiere Masse an Webseiten, Dokumenten, Artikeln[2] und Büchern führt zu einer wachsenden Unüberblickbarkeit. Dieses Problem verstärkt sich durch das exponentielle Wachstum des (publizierten) menschlichen Wissens, das gleichzeitig auch zu zunehmender Fragmentation von Wissensgebieten führt. Eine solche Entwicklung wurde schon recht früh z. B. von einem der Begründer der modernen Bibliometrie, Derek J. de Solla Price, erkannt und beschrieben [SP63].
Beispielhaft ist in Abb. 1.1 das Wachstum der wissenschaftlichen Publikationen im medizinischen (MEDLINE-Datenbank) und im Bioinformatik-Bereich Daraus lässt sich abschätzen, wie zunehmend unmöglich es für eine einzelne Person sein muss, auf konventionelle Art (bspw. durch Fachzeitschriften) einen Überblick über ihr Wissensgebiet zu erlangen.
Nun lassen sich verschiedene Verbesserungsmöglichkeiten für dieses Problem finden. Dazu gehören zum einen Verbesserungen in der Ausbildung von Schülern, Studenten und Wissenschaftlern. Diese Option ist die am ehesten als nachhaltig zu bezeichnende, löst aber das Problem nur teilweise und für

[1] http://www.ib.hu-berlin.de/~wumsta/infopub/semiothes/lexicon/default/index.html
[2] Oft werden wissenschaftlichen Veröffentlichungen auch als *Paper* (aus dem Amerikanischen) bezeichnet.



(a) Dokumentenbestand in MEDLINE

(b) Publikationen in Bioinformatik

Abbildung 1.1: Wachstumsrate wissenschaftlicher Veröffentlichungen – Quelle: [Jan07, S. 3,100]

den akuten Fall gar nicht.
Eine andere Möglichkeit ist es, Werkzeuge zu entwickeln, die die Literatur zu bestimmten Wissensgebieten ausfindig machen und dabei helfen zusammenhängende Wissensgebiete mit Fachliteratur zu erschließen. Ein solches Werkzeug könnte für Studenten und Wissenschaftler bei der Literaturrecherche eine Hilfe sein. Im vorliegenden Projekt soll nun, implementiert als Computersoftware, ein Ansatz einer solchen Hilfestellung entwickelt werden.

Ein Ansatz zur Literaturübersichtsgewinnung hat sich bei vorangegangenen Arbeiten als sehr erfolgsversprechend für die Zielgruppe erwiesen: Die Repräsentation von Dokumenten in Gruppen von zusammenhängenden, nach bestimmten Kriterien ähnlichen Dokumenten (Cluster) (vgl. [BDH06, BH07]). Eine nicht unwesentliche Frage dabei ist, wann ein Dokument einem anderen ähnlich ist und wie eine konkrete Repräsentation dieses Zusammenhangs einem Benutzer dargestellt werden kann. Als ein gutes Kriterium für die Bewertung der Ähnlichkeit zweier Dokumente stellte sich dabei die Analyse der Literaturreferenzen (Zitationsgraph) sowie die Untersuchung der verwendeten Wörter (Textähnlichkeit) heraus.

1.2 Einordnung des Projekts in Bezug zu vorhandener Forschung

Diese Arbeit soll die Arbeit der *Open Access* Bewegung unterstützen, die sich die freie Zugänglichkeit wissenschaftlicher Literatur zum Ziel gesetzt hat.
Das *Open Access* Protokoll bzw. dessen Umsetzung soll die Abhängigkeit von Wissenschaftlern und insbesondere Bibliotheken von Fachverlagen bzw. Zeitschriften lösen, indem Veröffentlichungen frei zugänglich als elektronische Publikation im Internet oder über öffentliche Einrichtungen verfügbar gemacht werden.
Die Verfügbarmachung von Publikationen in Fachzeitschriften wird, insbesondere von Bibliotheken, zunehmend als finanziell nicht mehr tragbar angesehen. Daher wird die Nutzung von, für Leser frei zugänglichen, Veröffentlichungsmöglichkeiten empfohlen.
Ein Problem bleibt der Qualitätsanspruch an wissenschaftliche Veröffentlichungen. Ein großer Vorteil der Publikation in etablierten Fachzeitschriften ist z. B. ein organisiertes *Peer Review* Verfahren; für andere Publikationswege ist dies ein noch nicht abschließend geklärtes Problem, zumal das *Peer Review* Verfahren nicht unumstritten ist (vgl. [Smi97]). Lösungsansätze versprechen hierfür beispiels-

weise *Open Access* Zeitschriften, die ein *Peer Review* durchführen, allerdings die Kosten für die Veröffentlichung dann meist den Autoren aufbürden. Daher ist dies ein nicht von allen Wissenschaftlern gern genutzter Weg.
Eine Lösung für allgemein frei verfügbare wissenschaftliche Literatur kann die Analyse der Zitationen bringen, wenn die Annahme zutrifft, dass ein öfter zitiertes Paper eher die Zustimmung anderer Wissenschaftler trifft, als eine wenig oder gar nicht zitierte Arbeit. Die Analyse von Zitationen ersetzt bzw. komplementiert die bisher klassische Form des *Peer Review*. (Vgl. [BH07])

Dieser andere Ansatz, alternativ oder komplementär zu der Veröffentlichung von wissenschaftlichen Arbeiten in Fachzeitschriften, wird auch durch die zahlreichen, u. a. in dieser Arbeit vorgestellten, wissenschaftlichen Literaturarchive repräsentiert. Dieser Repräsentation von Wissen, durch *Open Access* sehr gefördert, liegt eine lange Tradition der internationalen Informations- und Dokumentationsbewegung zugrunde, die eine Zentralisierung der Wissenssammlung bzw. ein zentrales Ordnungskriterium für Veröffentlichungen fordert. Populäre Vertreter dieser Bewegung finden sich bereits zu Beginn des 20. Jh., als große wissenschaftliche Entdeckungen gemacht wurden. In Deutschland zählt dazu insbesondere Wilhelm Ostwald, dessen Ideen erst heute realisiert werden. (Vgl. [Hap04])

Diese Arbeit soll nun, neben der oben beschriebenen Repräsentation von Dokumenten, die Übersichtlichkeit innerhalb der frei verfügbaren Literatur erhöhen und dem Benutzer Angaben zu Zitationen eines Dokumentes liefern.

1.3 Angestrebter Erkenntnisgewinn

Der angestrebte Erkenntnisgewinn leitet sich direkt aus der Realisierung des Systems ab, zu welcher im wesentlichen Modellierung, Implementierung und Testreihen gehören.
In Bezug auf die technischen Aspekte der Software werden die folgenden Forschungsfragen formuliert:

- An welchen Stellen können Optimierungen bei der Verarbeitungsgeschwindigkeit, z. B. durch Vorkalkulation von Werten, implementiert werden?
- Eignet sich der Ansatz für die Einbindung von weiteren Datenbanken? Welche Eigenschaften des Systems sind nötig, um andere Wissensgebiete/Datenbanken zu erschließen?

In Bezug auf die Anwendbarkeit lautet die Forschungsfrage:

- Bietet das hier entwickelte Tool die notwendigen Möglichkeiten und Funktionen, um die Literaturrecherche der Zielgruppe (Studenten und Wissenschaftler) zu verbessern?

1.4 Beschreibung des Systems

Vor dem in Kapitel 1.1 bis 1.3 vorgestellten Hintergrund kann eine Systembeschreibung entwickelt werden, die die Umrisse des zu entwickelnden Programms skizziert:
Das Ziel dieser Arbeit ist die Entwicklung eines Such- und Recherchesystems, das wissenschaftliche Literatur nach Stichwörtern (Suchbegriffen) ermitteln und ordnen kann. Die Ordnung der gefundenen Dokumente soll dabei durch verschiedenste Hilfsmittel vom Benutzer interaktiv gestaltet werden können, wobei „klassische“ Verfahren (wie sortieren und löschen) zur Auswahl stehen und die Möglichkeit besteht Dokumente automatisch in Gruppen einzuordnen. Bei dieser Art der Einordnung, sollen die

oben erwähnten Kriterien (bibliographischer Zusammenhang durch Zitationsgraph und / oder Textähnlichkeit) zum Einsatz kommen.
Das ganze System soll für den Anwender möglichst frei von zusätzlichen Voraussetzungen, wie der Installation von zusätzlicher Software, sein. Darüberhinaus wird es als hilfreich angesehen, wenn sich die Handhabung des Programms an der allgemein üblichen Bedienung von Suchmaschinen und / oder vergleichbaren Systemen orientiert, um einen Wiedererkennungseffekt zu erreichen. Aus diesen Gründen soll hier eine Webapplikation entwickelt werden. Dabei wird angenommen, dass ein Webbrowser sowie eine Internetverbindung allgemein verfügbar sind. Durch diese technische Ausgestaltung als Client-Server System besteht die Möglichkeit mit großen Datenmengen zu arbeiten, ohne diese beim Benutzer hinterlegen zu müssen.

Das System muss nicht alle Möglichkeiten der Recherche und Analyse bieten, die möglicherweise in bestehenden Systemen zu finden sind, da das Hauptaugenmerk auf der Evaluation der Dokumentengruppen und dem Umgang potentieller Nutzer mit diesen liegen soll. Die Entwicklung eines vollwertigen Literaturrecherchesystems ist deshalb nicht Schwerpunkt dieser Arbeit.

Eine konkretere Aufschlüsselung der einzelnen Anforderungen an das System findet sich in Kapitel 4.1.

1.5 Vorgehen und Entwicklungsabfolge

Nun ist es unmöglich, dass innerhalb dieser Arbeit eine vollständige Lösung für das Problem der Literaturrecherche bei wissenschaftlichen Arbeiten gefunden wird, bzw. die entwickelte Software alle vorhandenen Probleme behandelt. Daher soll hier ein prototypisches Programm entwickelt werden, das dem oben genannten Ansatz der Literaturaufbereitung folgt und für eine ausgewählte Zielgruppe von Nutzen sein kann.

Die Zielgruppe wurde auf Studenten und Wissenschaftler der Computerwissenschaften und angrenzender Fachgebiete festgelegt (Abschnitt 1.1), da angenommen wird, dass diese aufgrund notwendiger Erfahrung mit Onlinequellen sowie allgemeiner Computererfahrung am ehesten zur Nutzung eines solchen Programms bereit sind.
Ein weiterer Grund für diese Auswahl war die gute Verfügbarkeit von Datenbanken mit online zugänglicher Literatur und Zitationsdaten in diesem Fachgebiet.

Zunächst werden in Kapitel 2 verschiedene Literaturdatenbanken und diesem System ähnliche Systeme vorgestellt. Diese dienen zum einen als Inspiration für das vorliegende Projekt und liefern zum anderen die Datenbasis, mit der dieses System arbeiten wird. Eine konkrete Auswahl dieser Datenquellen ist in Kapitel 5.5 beschrieben.
In Kapitel 3 finden sich sowohl die theoretischen Grundlagen der hier vorgenommenen Datenaufbereitung, als auch die Grundlagen einer möglichen Evaluierung des Systems durch Benutzer. Eine beispielhafte Betrachtung und Auswertung der ausgewählten Daten ist in Kapitel 6.3 zu finden.

Um die obigen Forschungsfragen beantworten zu können, wurde ein Programm entwickelt in der folgenden Vorgehensweise:
Zunächst wurde ein Prototyp entwickelt, der zwar grundsätzliche Fähigkeiten der Software zeigte, aber z. B. in der grafischen Ausgabe und einiger Analysen beschränkt war. Mit diesem Programm wurden

dann weitere Anforderungen mittels Benutzerstudie gewonnen. Eine Übersicht über die detailierten Anforderungen an die Software ist in Kapitel 4.2 gezeigt, die Auswahl der Programmiersprache und weiterer Bibliotheken ist in Kapitel 5 beschrieben.
Mit den Ergebnissen der Evaluation und den grundsätzlichen Anforderungen an das Programm konnte daraufhin ein verbessertes System entwickelt werden (Kapitel 6), das die Anforderungen erfüllte, die zur Beantwortung der oben genannten Forschungsfragen nötig waren.

2 Literaturdatenbanken

Um zunächst einen Überblick über die zur Zeit vorhandenen Möglichkeiten im Bereich von elektronischen Literaturdatenbanken und zur Verwendung der in dieser Arbeit vorgestellten Daten zu bekommen, werden hier einige Datenbanken vorgestellt.
Einen Schwerpunkt der hier vorgestellten Systeme bilden frei zugängliche Datenbanken mit wissenschaftlicher Literatur. Dabei wird unterschieden zwischen Systemen, die zwar in der Benutzung frei sind, deren enthaltenen Dokumente aber kostenpflichtig sind, und solchen, bei denen sowohl Benutzung, als auch enthaltene Dokumente frei zugänglich sind.

2.1 Begriffe und Schnittstellen

Um die Funktionsweise und Interaktionsmöglichkeiten mit digitalen Bibliotheken zu verstehen, werden zunächst einige Begriffe und Schnittstellen erläutert.

2.1.1 Bibliographische Metadaten

Unter Metadaten werden im Allgemeinen Daten verstanden, die nicht zwangsweise in einer Datenmenge enthalten sind, aber trotzdem untrennbar mit dieser verbunden sind. Dies können zum Beispiel Eigenschaften der Datenmenge wie die Anzahl der Objekte sein. (Vgl. [bro05])
Unter bibliographischen Metadaten versteht man die Angaben von Autor, Titel, Jahr der Publikation, Regelungen zum Urheberrecht, Verlag etc. Die Aufnahme dieser Angaben losgelöst von dem Werk selbst ist eine jahrhundertealte bibliothekarische Praxis, während die explizite Angabe dieser Daten bei elektronischen Werken (z. B. Webseiten) noch relativ neu ist.
Um die Angabe von bibliothekarischen Metadaten zu standardisieren gibt es verschiedene Ansätze. Der bekannteste dieser Ansätze für elektronische Medien ist die Angabe nach der *Dublin Core* (DC)[3] Spezifikation, aber auch Angaben nach BIBTEX Standard haben eine große Verbreitung gefunden.
Unterschieden werden muss bei solchen Angaben des Weiteren noch zwischen der Datenspezifikation und der Repräsentation (also der konkreten Darstellung bzw. Schreibweise). Beispielsweise lassen sich Angaben nach dem *Dublin Core* Schema sowohl in HTML-Dateien einbetten, als auch gesondert in XML-Dateien (Beispiele: Dateien im RDF- (Resource Description Framework) oder OpenDocument-Format[4]) darstellen.

Viele Literaturdatenbanken haben sich mit ihren Formaten an dem *Dublin Core* und BIBTEX Schema orientiert, sehr oft sind diese aber mit eigenen Feldern erweitert. In Anhang C ist beispielhaft ein Datensatz nach *Dublin Core* abgebildet, wie er über das OAI-PMH Protokoll abgefragt werden kann.

[3] Die *Dublin Core* Metadata Initiative wurde 1994 in Chigago gegründet und hat sich zum Ziel gesetzt, ein Kernschema von Metadaten zu definieren, mit dem zunächst Resourcen im Internet / Webseiten ausgezeichnet werden konnten. Im Zuge der Weiterentwicklung wurde dieses Schema auch zur Beschreibung von Metadaten allgemeiner Dokumente verwendet.
[4] Dateiformat der Textverarbeitung *Open Office*

2.1.2 Schnittstellen

Die meisten Online-Literaturdatenbanken bieten elektronische Schnittstellen zur Datenabfrage von Metadaten nach einem anerkannten Standard. Dabei sind diese Schnittstellen zusätzlich zur „normalen“ Internet-Benutzerschnittstelle (Webseiten) verfügbar und von unterschiedlichem Leistungsumfang.

2.1.2.1 Open Archives Initiative (OAI)

Unter der Open Archives Initiative (OAI)-Schnittstelle wird zumeist eine Schnittstelle nach dem *OAI Protocol for Metadata Harvesting* (OAI-PMH)[5] verstanden. Technisch basiert eine solche Schnittstelle auf dem HTTP-(P)rotokoll, wobei Daten über normale HTTP-GET Parameter entgegengenommen werden, als Rückgabe aber eine XML-Datei geliefert wird.[6]
OAI unterscheidet grundsätzlich zwischen den beiden Teilnehmern einer solchen Kommunikation:

- **Data Provider:** „Datenlieferanten“, z. B. eine digitale Bibliothek, stellen Daten mit dem OAI-PMH Protokoll zur Verfügung.
- **Service Provider:** Verwenden das OAI-PMH Protokoll, um Daten abzurufen. Dabei stellen sie für den Endanwender einen Service zur Verfügung.

Das OAI-PMH Protokoll gibt Dokumentmetadaten mindestens im *Dublin Core* Schema zurück, viele Bibliotheken haben aber weitere Schemata im Angebot, um zusätzliche Attribute (z. B. Zitationsdaten) anzuzeigen. Dabei ist zwingend vorgeschrieben, dass Dokumente mit einer eindeutigen Identifikation bezeichnet werden, die innerhalb des *Data Provider* konstant bleibt.
Zu beachten ist, dass das OAI-PMH Protokoll keine Suchabfragen im herkömmlichen Sinne (also z. B. Suche nach Autoren, Titeln oder ähnliches) unterstützt. Im Allgemeinen wird davon ausgegangen, dass ein *Service Provider* den kompletten Datenumfang (oder zumindest einen großen Teil davon) lädt.

2.1.2.2 Z39.50

`Z39.50` ist ein Datenübertragungsprotokoll für bibliographische Daten, das federführend von der US-amerikanischen *Library of Congress*[7] entwickelt wurde. Mit diesem Protokoll sind direkte Datenabfragen, insbesondere auch konkrete Suchabfragen, möglich. Das Protokoll erlaubt dabei die Abfragen von mehreren Servern gleichzeitig, um höhere Abfragegeschwindigkeiten zu ermöglichen.
Das `Z39.50` Protokoll ist technisch relativ komplex, da es auf einem eigenem Kodierungsverfahren beruht und die Dokumentenmetadaten in einem speziellen Schema (*US-MARC*) gespeichert sind. Nichtsdestotrotz ist die Verbreitung entsprechender Server bei Bibliotheken sehr hoch.

Es existiert ein offizieller Nachfolger dieses Protokolls, welcher allerdings noch keine hohe Verbreitung gefunden hat: *ZING*. Dieses verwendet wie OAI-PMH auch HTTP als Übertragungsprotokoll und gibt Antwortdaten im XML-Format zurück. (Siehe [Wik07b])

[5] Siehe Spezifikation unter `http://www.openarchives.org/OAI/openarchivesprotocol.html`, Stand 2008-02-16
[6] Eine solche Architektur wird oft auch als *REST* (Representational State Transfer) Webservice bezeichnet.
[7] `http://www.loc.gov/z3950/`, Stand 2008-02-16

2.2 Übersicht über existierende, verwandte Systeme

Eine Übersicht über nicht verbundene und nicht zentral verwaltete Literatursysteme kann immer nur unvollständig sein. Dennoch soll hier eine knappe Zusammenfassung versucht werden, die die wichtigsten derartigen Systeme umfasst.
Dabei bietet sich eine weitere Unterteilung an, die die Datenaufbereitung des untersuchten Systems einbezieht. Damit unterscheiden sich Systeme, die eingegangene Dokumente ausschließlich archivieren (möglicherweise im Format konvertieren) und Systeme, die weitergehende Analysen durchführen und diese dem Benutzer anbieten bzw. mit den Daten die Suche bzw. Navigation erleichtern.
Daneben gibt es Systeme, die eher als Suchmaschine bzw. Meta-Suchmaschine bezeichnet werden können, da diese selber keine oder wenig Dokumente vorrätig haben, aber die Dokumente anderer Systeme in ihrem Index aufführen.

Eine weiteres Unterscheidungsmerkmal ist der Zugriff auf die Literaturdatenbanken: Es gibt Systeme, deren Benutzung frei zugänglich ist, und die die enthaltenen Dokumente frei zugänglich machen. Daneben gibt es Systeme, die zwar auch frei benutzbar sind, die aber keine frei nutzbaren Dokumente beinhalten. Meist sind die Dokumente solcher Systeme dann bei Verlagen oder anderen kostenpflichtigen Stellen hinterlegt. Weiterhin existieren kommerziell orientierte Systeme, in deren Index sich meist ein sehr großes Spektrum von verfügbaren Dokumenten befindet oder die sich auf eine bestimmte Wissenschaftsdisziplin spezialisiert haben.

Es werden hier ausschließlich Systeme aufgeführt, bei denen der Großteil der vorhandenen Dokumente frei verfügbar ist und die ohne aufwendige, datenintensive Anmeldung funktionieren. Dies ist beispielsweise bei Archiven der Fall, die nach dem *Open Access* Gedanken handeln. Hierzu findet man an verschiedenen Stellen auch zentrale Verzeichnisse, die zumindest „offizielle“ von einer Organisation unterstützte Systeme auflisten[8] Am Ende des Abschnitts sind beispielhaft aber auch einige kommerzielle Systeme vorgestellt. Darüberhinaus sind ausschließlich Datenbanken aufgeführt, deren primäres Ziel die Bereitstellung von Literatur ist und deren Daten durch maschinelle Prinzipien gewonnen wurden. Eine andere Herangehensweise sind Datenbanken, deren Inhalte durch partizipative Mechanismen gewonnen werden. Dies kann bspw. durch die Analyse sog. Bookmarklisten und / oder Tags bzw. Beschreibungen von Literatur einer Vielzahl von Benutzern geschehen. Beispiele dazu, die hier aber nicht weiter betrachtet werden sollen sind *CiteUlike*[9] und *BibSonomy*[10].

2.2.1 „Datensammelnde“ Systeme

Zu den „datensammelnden“ Systemen gehören all diejenigen Systeme, die zwar große Mengen von wissenschaftlichen Dokumenten sammeln, aber keine weitergehenden Analysen dieser Dokumente bereitstellen.
Um eine klarere Abgrenzung zu erlauben, sind die wichtigsten Eigenschaften hier aufgeführt:

- Wissenschaftliche Literatur wird, optional nach Fachgebieten sortiert, gesammelt und steht über ein elektronisches System zur Verfügung. Dabei wird unterschieden zwischen Systemen, die nur Metadaten (z. B. bibliografische Angaben) erfassen und Systemen, die auch das Dokument selbst speichern.

[8] Eine Beispiel hierfür ist http://www.webometrics.info/top200_rep.asp, mit einer Rankliste der „besten“ Archive. Allerdings ist diese Liste (nach subjektiver Wahrnehmung) weder vollständig noch spiegelt das angegebene Ranking die Realität wieder, da teilweise sind Archive mit sehr kleiner Dokumentenzahl hohe Ranklistenplätze einnehmen.
[9] http://www.citeulike.org/, Stand 2008-02-20
[10] http://www.bibsonomy.org/, Stand 2008-02-20

Möglich sind außerdem Mischformen, die bspw. nur die Zusammenfassung bzw. Abstract eines Dokuments aufnehmen.

- Um eine langfristige Zugänglichkeit zu gewährleisten, ist (falls nötig) das Dokumentenformat konvertiert. Das gewählte Dateiformat ist dabei ein gut dokumentiertes bzw. nach einer anerkannten Norm standardisiertes Format. Dies gilt sowohl für Metadaten, als auch für das Dokument selbst.

- Das System ermöglicht eine Suchmöglichkeit nach Metadaten (mindestens Autor und Titel). Darüberhinaus kann eine Volltext oder Stichwortsuche angeboten werden.

Solche „datensammelnden“ Informationssysteme gibt es in großer Zahl und in verschiedensten Ausprägungen. Daher hier einige ausgewählte Beispiele, die repräsentativ für viele ähnliche Systeme stehen:

- **ArXiv**[11]
  Das *ArXiv* System war eines der ersten Installationen, die zum Zweck des wissenschaftlichen Austauschs von Literatur, hauptsächlich von sog. Preprints[12], geschaffen wurde. Das Projekt wurde 1991 am US-amerikanischen Los Alamos Forschungszentrum für Physik gestartet, um den damals üblichen teuren und aufwendigen Austausch von Preprints als Fotokopie zu umgehen. Das System entwickelte sich schnell weiter, da andere Institute und Forschungseinrichtungen vor ähnlichen Problemen standen. (Vgl. [Gin95])
  Heute wird das System von der Cornell Universität im US Bundesstaat New York federführend betrieben und enthält über 451 000 Dokumente aus den Fachgebieten Physik, Mathematik, Informatik, Quantitative Biologie und Statistik. Den Hauptteil der Daten stellen nach wie vor Arbeiten aus der Physik dar.
  Neben den Dokumenten selbst werden Metainformationen wie Autor, Titel, Stichworte usw. gespeichert, nach denen gesucht werden kann. Das Dateiformat war in der Anfangsphase auf LaTeX beschränkt, mittlerweile können auch andere gängige Formate wie PDF, *Postscript* oder HTML verarbeitet werden. Jedes Dokument erhält eine eindeutige *ArXiv*-spezifische Kennung (ähnlich einer URN), mit der dieses Dokument identifiziert werden kann.
  Eine weitere Verarbeitung, außer der Eingruppierung in ein Verzeichnis mit vorgegebenen Sachgebieten (vom Autor angegeben) erfolgt nicht. Dokumente werden aber, soweit möglich und konvertierbar, als PDF-Dokument zum Download angeboten. Darüberhinaus sind die Metadaten des Archivs über das OAI-PMH Protokoll verfügbar.
  Da die Autoren selbst aktiv ihre Dokumente in das System einstellen müssen, bleibt der Zuwachs an Dokumenten begrenzt. In der Hauptzielgruppe ist *ArXiv* allerdings so weit verbreitet, dass viele Autoren das System nutzen. Der Zuwachs betrug im Jahr 2007 insgesamt 57 300 Arbeiten, wovon 39 945 ($\approx 70\%$) aus dem Fachgebiet Physik waren.
  Bemerkenswert ist, dass seit etwa 4 Jahren Arbeiten von neuen Autoren nur dann aufgenommen werden, wenn ein vorhandener, aktiver Autor die „Echtheit“ des neuen Autors bestätigt. Darüberhinaus werden regelmäßig Publikationen wieder aus dem Index genommen, wenn sich bspw. ein Plagiatsvorwurf erhärtet hat. Diese Maßnahmen sollen vor allem der Verbesserung der Qualität dienen.

[11] http://www.arXiv.org, Stand 2007-11-30

[12] Zur Veröffentlichung in Zeitschriften bestimmte wissenschaftliche Artikel (Papers). Unter Preprint wird meist ein Vorabdruck vom Verlag oder eine Kopie des Artikels verstanden, der vor einer Veröffentlichung, bspw. in einer Zeitschrift, verfügbar gemacht.

- **RePec**[13]
  Unter dem Namen *RePec* sind eine Reihe von Projekten angesiedelt, die auf eine gemeinsame Datenbasis zugreifen. Diese Datenbasis besteht aus Dokumentmetadaten in einem speziellen *RePec* eigenem ASCII-basierten Format, dem *Research Documents Information Format*[14].
  Die Datenbank ist so organisiert, dass Dokumente und Metadaten dezentral bei der jeweiligen Ursprungsorganisation verbleiben, *RePec* selbst bietet nur eine regelmäßige Zusammenführung der Metadaten und eine Suchfunktion an. Die Dokumente stammen aus einer großen Anzahl von Instituten, Organisationen und Firmen; insgesamt sind ca. 564 000 Dokumente im Datenbestand aufgeführt, von denen ca. 80% online verfügbar sind.
  Bei der Sammlung liegt der Schwerpunkt auf wirtschaftswissenschaftlichen Texten, vereinzelt auch aus den Bereichen Mathematik und Informatik. Die Einsortierung ist dabei aber nicht fachgebietsorientiert, sondern hierarchisch nach Organisationen geordnet. Wie auch schon bei *ArXiv* müssen Autoren bzw. Organisationen selbst für das Einstellen von Dokumenten sorgen.
  Es gibt mittlerweile ein in *RePec* organisiertes Projekt, das die gelisteten Dokumente analysiert in Bezug auf Zusammenarbeit von Autoren und das bibliographische Angaben, unter anderem auch Zitationen bzw. Referenzen, extrahiert mit den Algorithmen des *CiteSeer* Projekts (siehe unten). Da dieses Projekt noch relativ weit am Anfang steht, ist diese Zitationsdatenbasis von *RePec* für eine umfassende Analyse noch nicht geeignet.

- **Sammlungen von Veröffentlichungen und Abschlussarbeiten an Universitäten:**
  Viele Universitäten und Organisationen haben in den 1990er Jahren angefangen die Abschlussarbeiten von Studenten und Wissenschaftlern in eine elektronische Sammlung aufzunehmen. Bis auf sehr wenige Ausnahmen blieben diese Bemühungen aber immer auf die jeweilige Institution beschränkt.
  Diese Sammlungen hatten und haben unterschiedliche Ausprägungen, Umfänge und Ziele, je nachdem wie die entsprechende Universität die Benutzung des digitalen Archivs forcierte und / oder weitergehende Ziele mit der Arbeit verfolgte. In vielen Universitäten in Deutschland blieb die Existenz eines solchen Archivs aber eine kaum bekannte und verwendete, zudem freiwillige, Möglichkeit, so dass der Umfang dieser Sammlungen eher klein blieb (vgl. auch [SS08]). Sehr oft ist auch zu beobachten, dass entsprechende Projekte an Universitäten gestartet wurden und nach einigen Jahren wieder eingeschlafen sind.
  Viele solcher Archive sind mit dem *Open Access* Gedanken verbunden und haben ihre Sammlungen vernetzt. Es ist allerdings zu beobachten, dass Konzentrationsbestrebungen bestehen, um zumindest gemeinsame Verzeichnisse (verbunden durch die OAI Technik) zum Suchen und Recherchieren von relevanter Literatur zu schaffen (vgl. [SS08]). Beispiele für Aktivitäten einzelner Institutionen sind:

  - **Zentrales Digitales Archiv der Technischen Universität Berlin**[15]
    Dieses Archiv enthält hauptsächlich Dissertationen aus verschiedenen Fachbereichen, scheint aber viele Lücken zu enthalten. Ausschließlich akzeptiertes und angebotenes Dateiformat ist PDF.
    Die Technische Universität hat sich mit vielen anderen Universitäten zusammengeschlos-

[13] http://www.repec.org, Stand 2007-12-07
[14] Beschreibung findet sich hier: http://openlib.org/acmes/root/docu/redif_1.html, Stand 2008-02-16
[15] http://opus.kobv.de/tuberlin/ – 1615 Dokumente, Stand 2007-11-30

sen um Veröffentlichungen über ein einheitliches System (*OPUS*)[16] abfragen zu können. In diesem System sind Dokumente mit einer eindeutigen ID (URN) gekennzeichnet, die ein späteres Zitieren einfacher und eindeutiger gestalten sollen.
Die Anzahl der seitens der TU-Berlin eingestellten Dokumente lag zum Zeitpunkt der Recherche bei 1 677 Arbeiten.

- **EDOC – Dokumenten und Publikationsserver der Humboldt-Universität zu Berlin**[17]
  Engagiertes Projekt um sowohl Dissertationen, Habilitationen und Diplomarbeiten, als auch diverse andere wissenschaftliche Dokumente zu archivieren. Es werden verschiedene Dateiformate von Autoren akzeptiert, die intern in ein eigens entwickeltes XML-Format (DiML) konvertiert werden. Darüberhinaus versteht sich das Projekt als Sammelstelle für Dokumente verschiedenster Art von historischen Aufsätzen über Materialien zu Vorträgen bis hin zu Forschungspublikationen einzelner Fachbereiche.
  Bemerkenswert ist, dass im Gegensatz zu den meisten anderen Archivsystemen nicht nur das Dokument als solches (mit zusätzlichen Metadaten) gespeichert wird, sondern besondere Anforderungen an die Erstellung des Dokumentes erhoben werden, um später zusätzliche Informationen aus diesem Dokument verarbeiten zu können. Diese Anforderungen beinhalten beispielsweise die obligatorische Benutzung einer eigens für diesen Zweck erstellten Dokumentvorlage bei Verwendung von *Microsoft-Word* oder *OpenOffice*.
  Die Gesamtanzahl der verfügbaren Dokumente (nur Abschlussarbeiten und abgelegte *Open Access* Veröffentlichungen) lag zum Zeitpunkt der Recherche bei insgesamt 1 841 Publikationen.
- **DissOnline**[18]
  Ein Projekt der Deutschen Nationalbibliothek und der DFG (Deutsche Forschungsgemeinschaft) um das digitale Publizieren und Verfügbarmachen von Dissertationen und Habilitationen zu ermöglichen. Dabei wird eine Kopie des Dokuments auf dem Server der Deutschen Nationalbibliothek vorgehalten. Jedes Dokument wird dabei durch eine eindeutige ID (URN) identifizierbar und zitierbar gemacht. Dazu ist, als erweiterter Service, durch eine Kooperation mit Fachverlagen ein Druck des Dokuments als Buch möglich.
  Der interne Austausch von Informationen über ein Dokument geschieht durch ein eigens entwickeltes XML-Format (*XMetaDiss*), das aber ausschließlich die Meta-Informationen eines Dokumentes erfasst (im Gegensatz zu *EDOC*). Die Dokumente selbst stehen lediglich im Format PDF zur Verfügung, wobei oft bei Suchergebnissen noch andere Quellen vermerkt sind, die möglicherweise auch noch andere Formate dieses Dokumentes bereitstellen.
  Dieser Dienst der Nationalbibliothek ist beschränkt auf Dissertationen und Habilitationen von Wissenschaftlern an deutschen Universitäten. Im Test drängte sich der Eindruck auf, dass die Menge der gefundenen Dokumente auf eine eher geringe Nutzung des Systems schließen lässt, auch wenn keine offiziellen Zahlen zu finden waren.

[16] Abfragen können über: http://elib.uni-stuttgart.de/opus/gemeinsame_suche.php (Stand: 2007-11-30) gemacht werden. Dokumente werden dabei nicht zentral gesammelt oder verwaltet, sondern von den jeweiligen Datensystemen der entsprechenden Hochschule abgefragt.
[17] http://edoc.hu-berlin.de/, Stand 2007-12-08
[18] http://www.dissonline.de/, Stand 2007-12-06

- **Weitere Archive:**
  Es existieren im allgemeinen noch eine Vielzahl weitererer Archive mit frei zugänglichem wissenschaftlichen Material an deutschen Hochschulen. Beispielhaft hier zusammengetragen für die Technische Universität Berlin.
  * **Datenbank Hochschulbibliographie**[19]
    Datenbank mit Autoren, Titeln, Fachbereichen und Veröffentlichungsorten von Publikationen, die an der Universität entstanden sind. Da diese Datenbank im Laufe der Jahre in verschiedensten organisatorischen Teilen angesiedelt war, ist sie in entsprechend viele Teile zergliedert.
    Interessant ist der Fakt, dass überhaupt erst seit 1961 solche Informationen gesammelt wurden. Die Daten der Jahre vor 1993 sind nur in gedruckter Form zugänglich.
  * **Spezielle Datenbanken**[20]
    Darüberhinaus betreibt die Universität eine Reihe von Datenbanken, die teilweise Literaturhinweise und / oder weitere Daten zu speziellen Themen oder für ausgesuchte Fachbereiche speichern.
    Allen diesen Datenbanken ist gemeinsam, dass sie meist nur für eine spezielle Nutzergruppe gedacht und auch nicht über eine gemeinsame Oberfläche ansprechbar sind.

    Darüberhinaus gibt es in den einzelnen Fachbereichen dezentral gepflegte Datenbanken mit Publikationen und Abschlussarbeiten. Da dies jeder Fachbereich selbst handhabt und teilweise sogar jeder Lehrstuhl seine eigene Sammlung aufbaut, gibt es für diese Sammlungen keinerlei einheitliches System oder übergeordnete Suchmöglichkeit.

- **Digitalisierungsprojekte**
  Einige Projekte befassen sich mit der Digitalisierung und Zugänglichmachung von „alten“ Dokumenten, Büchern und sonstigen Materialien. Dadurch können viele Dokumente überhaupt erst von einem größeren Nutzerkreis verwendet werden und möglicherweise auch auf anderere Art erschlossen werden z. B. durch die Möglichkeit der Volltextsuche.
  Ein ungelöstes Problem ist allerdings das Urheberrecht, weshalb dieser Ansatz, nach bisherigem Kenntnisstand, nur für eine begrenzte Anzahl von Dokumenten in Frage kommt.
  Zusätzlich kommt hinzu, dass die meisten dieser Systeme nicht mit anderen vernetzt sind, also bspw. Dokumente nicht über ein übergeordnetes Suchsystem gefunden werden können. Dieses Manko versuchen verschiedene Projekte auf unterschiedliche Art und Weise zu beheben. Das wohl am weitesten fortgeschrittene Projekt, nach eigener Selbstdarstellung, ist der kostenpflichtige *FirstSearch* Service der OCLC[21] (Online Computer Library Center) mit ca. 10.000.000 Objekten als Volltext oder Bild. Hier ist zu beachten, dass nicht unterschieden wird zwischen gescannten Medien oder zuvor schon digital vorliegenden Daten. Leider ist auch die Zählweise von Artikeln unklar, weshalb die hohe Zahl der enthaltenen Dokumenten möglicherweise anders zu interpretieren ist.
  Sehr bekannt ist auch das Digitalisierungsprojekt *Google-Books*[22], das zusammen mit einigen Verlagen und Bibliotheken gestartet wurde. Insbesondere die Ankündigung auch urheberrechtlich geschützte Dokumente in begrenztem Umfang verfügbar zu machen, hat dabei für Kontro-

[19] Übersicht unter http://www.ub.tu-berlin.de/index.php?id=336, Stand 2007-11-30
[20] Übersicht unter: http://www.ub.tu-berlin.de/index.php?id=304, Stand 2007-12-07
[21] http://www.oclc.org/firstsearch/default.htm, Stand 2007-12-14
[22] http://books.google.com/, Stand 2007-12-07

versen und andauernden juristischen Streit gesorgt. Der Umfang der bisher geleisteten Arbeit ist unbekannt, es stehen aber schon eine beträchtliche Anzahl Bücher zur Verfügung. Ergebnisse des Digitalisierungsprojekts sind teilweise auch in die normale Websuche von *Google* und in die wissenschaftliche Suche *Google-Scholar* eingeblendet.
In Deutschland sind mehrere kleinere Projekte aktiv, bspw. das *Göttinger Digitalisierungszentrum*[23], der Universitätsbibliothek Göttingen mit z. Z. mehr als 5 Mio. digitalisierten Seiten von im wesentlichen wissenschaftlicher Literatur. Einen gewissen Bekanntheitsgrad hat die dort bearbeitete digitale Version der Gutenberg Bibel erreicht. Im Aufbau befindet sich ein Gesamtverzeichnis von digitalisierten Resourcen in Deutschland.[24]

- **DBLP**[25]
  DBLP (Digital Bibliography & Library Project) ist in erster Linie keine Literaturdatenbank, sondern versucht den Beitrag von Autoren an Veröffentlichungen nachzuvollziehen. Es werden Beiträge zu Konferenzen und einigen ausgewählten Zeitschriften ausgewertet und in die Datenbank übertragen. Laut Beschreibung geschieht ein Hauptteil dieser Arbeit automatisch, wird aber von Hand nachgepflegt.
  Es werden in erster Linie Bibliographien von Personen in der Informatik aufgenommen, dazu sind auch weitere Informationen zu Personen verfügbar, wie z. B. Organisationen und Internetadressen. Insgesamt sind z. Z. etwa 955 000 Artikel gelistet. Erst bei neueren Artikeln wurden auch Referenzen von Veröffentlichungen aufgenommen. In Relation zur Gesamtdatenbasis sind diese Angaben aber erst rudimentär vorhanden.
  Damit stellt *DBLP* eine der größten, freien Sammlungen von Literaturverweisen im Gebiet der Informatik dar. Die Datenbasis steht dabei für wissenschaftliche Zwecke, in Form einer XML-Datei, auch öffentlich zur Verfügung.

- **The Collection of Computer Science Bibliographies**[26]
  Die *Collection of Computer Science Bibliographies* ist eine Sammlung von BibTeX Einträgen aus „verschiedenen“ Quellen, unter anderem *DBLP* und *CiteSeer*. Zusätzlich können einzelne Autoren und Organisationen ihre Dokumente anmelden. Insgesamt stehen über 4 Mio. Datensätze in der Datenbank, von denen allerdings zwischen 30% und 50% Duplikate sind (vgl. [Wik08c]).[27]
  Es werden ausschließlich Dokumente aus dem Bereich Informatik und Mathematik aufgenommen, teilweise stehen noch gesonderte Informationen zu einzelnen Personen zur Verfügung (bspw. zugehörige Organisationen, Internetadressen etc.). Die Suchfunktionen des Systems sind einfach, es steht im Prinzip nur eine Suche nach einzelnen Wörtern im Namen oder Titel zur Verfügung.
  Das Archiv arbeitet ausschließlich mit den Metadaten, die in BibTeX Einträgen gespeichert sind. Dabei sind ca. 25% der Einträge mit einem kurzen Abstract gespeichert und mehr als 65% der Einträge haben mind. eine URL mit weiteren Informationen zu einem Dokument.
  Die gesamte Datenbasis ist in Form einer komprimierten Datei (ca. 600Mb) für wissenschaftliche Zwecke zum Download verfügbar.

[23] http://www.gdz-cms.de/, Stand 2007-12-07
[24] http://www.zvdd.de/, (Stand 2007-12-07)
[25] http://dblp.uni-trier.de/, Stand 2008-02-06
[26] http://liinwww.ira.uka.de/bibliography/, Stand 2008-02-20
[27] Die Angaben variieren.

- **ScientificCommons**[28]
  Ein relativ neues Angebot der Universität St.Gallen (CH), dass sich zum Ziel gesetzt hat eines der weltweit größten Kommuniktionsmediendatenbanken für wissenschaftliche Informationsprodukte aufzubauen. Zusätzlich besteht die Zielsetzung, soziale bzw. arbeitstechnische Verknüpfungen zwischen Autoren darzustellen. Dazu sollen im weiteren Teil des Projets Ontologien entwickelt werden, um Verknüpfungen verschiedener Fachgebieten über die Zusammenarbeit einzelner Personen zu ermitteln.
  Zur Zeit sind etwa 17. Mio. Publikationen und 7 Mio Autoren verzeichnet zu verschiedensten Fachgebieten ohne expliziten Schwerpunkt. Die Dokument-Metadaten werden dabei aus unterschiedlichen OAI-Quellen bezogen, aber auch per Crawler gesucht. Anschließend wird, falls möglich, das Orginaldokument für eine Volltextindizierung verwendet. Autoren bzw. Institutionen können eigene Archive bzw. OAI-PMH konforme Server zur Indizierung anmelden.
  Die z. Z. vorhandenen Möglichkeiten des Archivs sind die Suche nach allgemeinen Stichwörtern bzw. Autorennamen, wobei auf ein vorhandenes Suchergebnis weitere Filter eingesetzt werden können. Sofern das Dokument vom Dienst als Volltext analysiert wurde, steht eine Kopie des Dokumentes für den Benutzer zur Verfügung.

[28] http://www.scientificcommons.org/, Stand 2008-02-20

2.2.2 Analysierende Systeme

Unter analysierenden Systemen sind all diejenigen Systeme zu verstehen, die außer dem reinen Sammeln von Dokumenten auch weitergehende Analysen bereitstellen. Zu diesen Analysen können eine Inhaltsbestimmung, beispielsweise für eine Eingruppierung zu einem bestimmten Fachgebiet, oder auch eine komplexe Zitationsanalyse mit Bestimmung der referenzierten und referenzierenden Dokumente gehören.
Auch hier gilt, dass die folgende Auflistung keinen Anspruch auf Vollständigkeit erhebt, da es keine globalen Verzeichnisse solcher Dienste gibt. Es ist außerdem zu beachten, dass insbesondere die in diesem Bereich aufgeführten Systeme als „Forschungsobjekte“ entwickelt und konzipiert wurden. Das bedeutet auch, dass die Entwicklung und der Fortbestand meist von sehr wenigen Personen abhängig ist und nur „projektweise“ weitergeführt wird.

Allen diesen Projekten ist gemein, dass technisch Methoden angewendet werden, die in Kapitel 3.1 genauer beschrieben sind. Hier sollen aber die Systeme selbst im Vordergrund stehen.

- **CiteSeer**[29]
  Die *CiteSeer* Literaturdatenbank fokussiert hauptsächlich auf die Fachgebiete Informatik und Mathematik. Die Dokumente, die wohl hauptsächlich im Internet per Crawler[30] gefunden werden, sind im Ursprungsformat, bzw. konvertiert nach PDF, als Download erhältlich. Zum Zeitpunkt dieser Recherche waren 767 558 Dokumente indexiert.
  Alle Dokumente werden automatisch und autonom (sog. *Autonomous Citation Indexing*, vgl. [GBL98]) aufgearbeitet und Autor, Titel, Abstract (oft auch in Form der Einleitung des Dokuments), Erscheinungsjahr und angegebene Referenzen werden extrahiert. Da dieser Prozess sehr fehleranfällig ist, besteht die Möglichkeit für angemeldete Benutzer die erkannten Dokumentendaten zu korrigieren.
  CiteSeer referenziert im Weiteren die gefundenen bibliographischen Angaben eines Dokumentes automatisch gegen die in der Datenbasis vorhandenen Dokumente und registriert diese Referenzen. Angaben zu Dokumenten, die nicht in der Datenbasis vorhanden sind, werden bei *CiteSeer* gesondert gespeichert.
  Suchergebnisse nach Dokumenten werden standardmäßig nach der Anzahl der Zitationen geordnet dargestellt und an Informationen zu einem Dokument werden Referenzen und ähnliche Dokumente, Abstract sowie eine BibTeX-Formatierung angezeigt. Eine gesonderte Suche nach ähnlichen Dokumenten und der zitierten Textstelle ist möglich.
  Eine Besonderheit im Vergleich zu anderen Literaturdatenbanken ist, dass die Datenbank (zumindest ein großer Teil davon) sowie der Quelltext des eigentlichen *CiteSeer*-Systems öffentlich zugänglich sind. Die Metadaten sind außerdem per OAI-PMH Interface abrufbar, allerdings sind in den hierüber abgerufenen Datensätzen nicht alle Zitationsdaten enthalten.
  Leider ist das *CiteSeer*-System relativ häufig nicht erreichbar und seit einiger Zeit werden keine Dokumente mehr aktualisiert. Es ist jedoch ein Nachfolgesystem *Next Generation CiteSeer* angekündigt, allerdings noch ohne konkrete Aussagen zur Verfügbarkeit oder Zeitplanung.

  Darüberhinaus existiert ein „Schwestersystem“ **BizSeer**[31] mit derselben Technik und ähnlicher Bedienung aber mit dem Schwerpunkt auf wirtschaftswissenschaftlichen Themen.

[29] http://citeseer.ist.psu.edu/, Stand 2007-12-13
[30] Es ist Autoren auch möglich eine URL für die Dokumentensuche anzugeben.
[31] http://bizseer.ist.psu.edu/index.html/, Stand 2008-02-06
Das System war früher unter dem Namen *SmealSearch* bekannt.

- **Citebase**[32]
  Citebase ist eine Literaturdatenbank, ähnlich dem *CiteSeer* System. Dokumente sind im System meist als PDF-Datei hinterlegt und können so von Benutzern direkt verwendet werden.
  Citebase extrahiert „halb-automatisch“[33] alle Zitationen eines Dokuments und referenziert diese mit den Dokumenten in der Datenbank. Dabei werden unbekannte (nicht in der Datenbasis vorhandene) Dokumente mit einem Link zu einer Suchmaschine angezeigt.
  Neu an diesem System ist die Einbeziehung von Downloadzahlen einer „einfachen“ Literaturdatenbank (*ArXiv*, siehe oben) in das Ranking von Dokumenten. Dabei sind die einzelnen Downloads nach Ländern, Daten und Organisationen aufgeschlüsselt. Durch dieses Merkmal sind für einen interessierten Benutzer weitere Informationen ermittelbar.
  Die Datenbasis von *Citebase* ist nach Angaben der Betreiber aus dem *ArXiv*-System übernommen, zusätzlich wurden über OAI-PMH frei zugängliche Quellen mit wissenschaftlicher Literatur indexiert. Es ist leider nicht ersichtlich, wieviele Dokumente insgesamt und zu welchen expliziten Fachgebieten Dokumente erhältlich sind; eine knappe Liste mit zusätzlichen OAI Quellen lässt auf einen Schwerpunkt in Medizin bzw. Bioinformatik zusätzlich zu den *ArXiv*-Daten schließen. Weiterhin wird von den Betreibern angegeben, dass die Datenbasis noch sehr unvollständig sei.
  Da das Indexieren und Extrahieren von Informationen automatisch erfolgt, sind Fehler nicht auszuschließen. Daher wird Autoren ermöglicht, nach Anmeldung, solche Fehler zu beseitigen oder auch selbst neue Dokumente in das System zu stellen.

  Die Datensätze von *Citebase* sind nicht öffentlich zugänglich, allerdings geben die Betreiber an, dass diese kostenfrei für wissenschaftliche Zwecke zur Verfügung gestellt werden können. Eine andere interessante Möglichkeit ist die Verwendung von OAI-PMH zur Datenabfrage, da *Citebase* unterschiedliche Datenformate über die OAI-Schnittstelle anbietet, inklusive der Möglichkeit alle Referenzen eines Dokumentes im Detail abzurufen.

- **Cora/Rexa**[34]
  Cora bzw. dessen Nachfolger *Rexa* sind bzw. waren als Literaturdatenbank im Stil von *CiteSeer* geplant. Allerdings hat *Rexa* den zusätzlichen Anspruch Objekte im Sinne von Personen und Organisationen aber auch Ereignisse wie Konferenzen, Forschungszusammenarbeiten und Veröffentlichungen zusammenfassen zu können und gemeinsam präsentieren zu können. Damit würden z. B. bei der Suche nach einer Person auch Personen gefunden, die mit dieser zusammengearbeitet haben, was bei der Suche nach für ein Thema relevanter Literatur hilfreich sein kann.
  In der Praxis kämpft jedoch auch dieser Ansatz mit den üblichen Problemen wie Schreibfehlern, unkorrekten Referenzen und Fehlern bei der Aufarbeitung von digitalen Medien. Daher sind Autoren oft mit unterschiedlicher Schreibweise mehrfach aufgeführt und nicht alle Veröffentlichungen eines Autors sind auch diesem (in der Suche nach Personen) zugeordnet. Nichtsdestotrotz sind zu der Person des Autors sehr viel mehr Informationen verfügbar, als in anderen Systemen; zu einigen Autoren sind sogar Bilder abgelegt.
  Für ein Dokument sind die üblichen Informationen wie Autor, Titel, Seitenangaben, etc. verfügbar und ein Link zu einer im System gespeicherten elektronischen Version des Dokumentes ist

[32] http://www.citebase.org/, Stand 2008-02-04
[33] Dieser Ausdruck ist der Programmbeschreibung entnommen, die teilweise auch den Begriff „automatisch“ verwendet. Eine weitere Erklärung findet sich in der Beschreibung nicht. Eventuell beziehen sich die beiden Begriffe darauf, dass die Autoren (und Betreiber) Einträge korrigieren können.
[34] http://rexa.info/, Stand 2008-02-04

angegeben.[35] Darüberhinaus sind die verwendeten Literaturreferenzen aufgeführt und bekannte Zitationen zu einem Dokument angegeben. Die Liste mit Referenzen ist laut *Rexa* nicht vollständig. Allerdings ist unklar, wo genau Referenzen nicht angegeben werden, da auch solche Referenzen aufgeführt sind, die nicht als Dokument im System bekannt sind (solche Dokumente werden als „Pseudodokument“ z. B. bei Suchanfragen mit aufgeführt).
Eine Besonderheit ist die automatische Themen- bzw. Schlüsselworterkennung, die zum einen Teil auf angegebenen Schlagwörtern und zum anderen Teil auf einer N-gram Analyse (Häufigkeitsanalyse von Wortkombinationen, hier meist von zwei Wörtern, sog. Bi-gramme) beruhenden Auswertung von Dokumenten beruht. Eine weitere Besonderheit ist die Möglichkeit einzelnen Dokumenten freie „Tags“ zu vergeben und sie damit auszuzeichnen. Diese Tags sind für alle Benutzer sichtbar und änderbar. Mit Hilfe der Suche nach Tags können thematisch passende Dokumente gefunden werden.
Die Datenbasis von *Rexa* beruht hauptsächlich auf einer eigenen Websuche (Crawler) und von Hand eingegebenen URLs. Es gibt keine Möglichkeit im System vorhandene Dokumente zu verbessern oder zusätzliche Informationen zu hinterlegen. Zum Zeitpunkt dieser Recherche waren 379 011 Dokumente als Gesamtbestand angegeben und insgesamt 7 050 439 Referenzen (im Sinne von Verknüpfungen) hinterlegt, wobei der Hauptfokus der Datenbank im Bereich Informatik und Mathematik liegt. Ein Teil der Daten des Systems ist außerdem per Download erhältlich.
Unklar ist, welchen Status diese Literaturdatenbank hat. Grundsätzlich ist der Zugang nur nach vorheriger Registrierung[36] möglich. Die angebotenen Dokumente scheinen auch schon etwas älter zu sein und es ist unklar, ob Pläne bestehen das System weiterbestehen zu lassen oder weiterzuentwickeln.[37]

- **The Smithsonian/NASA Astrophysics Data System**[38]
  Diese Datenbank besteht aus drei nach Fachgebieten getrennten Bereichen: Astronomie und Astrophysik, Physik und Dokumenten aus *ArXiv*. Insgesamt sind über 6,2 Millionen Dokumente aufgenommen, zu denen allgemeine Metadaten wie Autor, Titel, Journaltitel und Seitenangaben sowie eine Liste von Referenzen und Zitationen aufgeführt sind. Die Liste der Referenzen ist dabei nicht immer vollständig (wird auch so angegeben), übersteigt aber deutlich die Anzahl der in der Datenbasis vorhandenen Dokumente. Der inhaltliche Schwerpunkt der Datenbank ist die Astrophysik.
  Eine Besonderheit dieser Literaturdatenbank liegt in dem großen Umfang der vorhandenen Daten. Dokumente, die nicht elektronisch vorlagen, wurden dafür gescannt und mittels Zeichenerkennung analysiert. Suchergebnisse, die solche Dokumente umfassen, bieten die Ansicht der aus dem Scan entstandenen Bilddateien an. Die Datenbank versteht sich allerdings in erster Linie nicht als Volltextserver, sondern bietet meist nur Abstracts der Dokumente an. Zusätzlich sind meist Links zu weiteren Quellen angegeben (Verlage etc.), von wo der Volltext abgerufen werden kann. Dadurch sind nicht alle Dokumente frei zugänglich.
  Ein Offline-Zugang zu der Datenbank wird nicht angeboten, allerdings ist es erlaubt die Webseite bzw. die Suchfunktionen auch per Programmlogik anzusteuern. Alternativ wird eine `Z39.50` Schnittstelle angeboten, mit der Suchanfragen möglich sind. Dadurch, dass die Literaturdatenbank von einer großen staatlichen Stelle mit eigenem Interesse (*NASA*) unterstützt wird, bietet sie für die Literaturrecherche der Zielgruppe eine hohe Qualität.

[35] Im Test funktionierte diese Funktion allerdings nicht, daher kann keine Aussage über das Dateiformat angegeben werden.
[36] Eine E-Mailadresse ist aber ausreichend.
[37] Diesbezügliche Aussagen im zugehörigen Blog scheinen jedenfalls überholt zu sein.
[38] `http://www.adsabs.harvard.edu/`, Stand 2008-02-06

- **HubMed**[39]
  Das *HubMed* System verwendet die Datenbasis von *PubMed*[40], einer Literaturdatenbank mit Schwerpunkt Medizin, die in den USA vom Gesundheitsministerium betrieben wird. Dabei ist *PubMed* eine der umfangreichsten Datenbanken zu medizinischen Themen und umfasste zum Zeitpunkt der Recherche ca. 17 Millionen Artikel. Dabei sind die eigentlichen Dokumente meist Zeitschriftenartikel und nicht online einsehbar.
  HubMed stellt zunächst eine Art alternatives Interface zu dieser Datenbank dar, bringt aber interessante Möglichkeiten mit, wie z. B. eine graphische Darstellung des Zitationsgraphen (bzw. einen Ausschnitt davon) und eine Funktion zur Gruppierung der Dokumente in Gruppen (Cluster).
  Die graphische Darstellung stellt ein Dokument in den Mittelpunkt und versucht ähnliche Dokumente („related“) zu diesem Dokument zu finden. Die Clusterfunktion ist offensichtlich auf Textähnlichkeit aufgebaut und produziert ein Ergebnis, wie es auch auf einigen Suchmaschinen[41] zu finden ist. Es werden immer nur die Ergebnisse der ersten Suchseite verarbeitet und in Clustern angezeigt.
  Bei dieser Datenbank bzw. Aufarbeitung ist der Ansatz technisch interessant, die Umsetzung jedoch noch nicht komplett geglückt – bei der hier durchgeführten Recherche funktionierte die Seite beispielsweise dauerhaft nur unzureichend.

2.2.3 Metasuchsysteme und kommerzielle Dienste

Diese Kategorie beinhaltet Systeme, die sich im Kern auf das Suchen in und mit schon vorhandenen Daten beschränken bzw. Daten von anderen Systemen verwenden. Darüberhinaus können einige Systeme auch als Mischsysteme gelten, wenn „analysierende Systeme“ (siehe oben) mit weiteren Daten kombiniert oder ergänzt werden.

- **OAIster**[42]
  Dies ist eine klassische Suchmaschine für Dokumente aller Art, die in an das System angeschlossenen Systemen gesucht werden.
  Dabei werden bei diesem Dienst keine Dokumente oder Metainfomationen dauerhaft gespeichert, sondern es werden Daten aus per OAI Protokoll abgefragten Datenbanken zwischengespeichert. Dadurch steht eine sehr große Menge Dokumente und Materialien zur Verfügung, die dann durch Download von den anderen Systemen, sofern die Nutzungsbedingungen dies zulassen und es online verfügbare Quellen sind, verwendet werden können.

- **Google Scholar**[43]
  Google Schoolar ist offiziell noch in der Erprobungsphase „Beta“, bietet aber schon ein Fülle von Möglichkeiten nach Literatur zu suchen, Volltexte zu finden, Referenzen bzw. Zitationen aufzulösen, ähnliche Dokumente zu suchen und bibliographische Angaben in normierten Formaten (BIBTEX) zu übernehmen.
  Viele der angezeigten Dokumente sind dabei nicht direkt von Google aufgenommen, sondern durch Kooperationen mit Fachverlagen und anderen Diensten (u. a. auch freie Dienste wie *CiteSeer* und *ArXiv*) indexiert. Dies hat zur Folge, dass insbesondere bei den Titeln, die durch Fachverlage aufgenommen wurden, das Dokument nicht direkt zur Verfügung steht, sondern käuflich

[39] http://www.hubmed.org/, Stand 2008-02-06
[40] http://www.pubmed.org, Stand 2008-02-06
[41] Zu nennen wäre hier z. B.: http://clusty.com/ oder auch das Carrot^2 System: http://www.carrot2.org/.
[42] http://www.oaister.org/, Stand 2007-12-07
[43] http://scholar.google.com/, Stand 2007-12-14

erworben werden muss (sofern nicht noch andere Quellen zur Verfügung stehen).
Eine interessante Funktion findet sich bei den Einstellungen: Dort können direkte Links zu einer angeschlossenen Bibliothek mit angezeigt werden, sofern ein Dokument in dieser Bibliothek vorhanden ist. Bei Zugang von einer registrierten IP-Adresse wird dieser Link von Google automatisch in den Suchergebnissen eingebettet.
Es sind von Seiten Googles keinerlei Informationen über Anzahl der indizierten Dokumente oder Deckungsumfang über Fachgebiete veröffentlicht. Eine genauere Betrachtung des Dienstes von Google ist bei [ND06] und [Kad06] zu finden.

- **Microsoft Live Search – Academic**[44]
  Microsofts Antwort auf *Google Schoolar* nennt sich *Live Search – Academic* bzw. in der deutschen Version *Live Search – Wissenschaftlich* und ist ähnlich der *Google*- Version auch als „Beta" gekennzeichnet.
  Die Funktionalität ist sehr ähnlich, wenngleich die Bedienungsoberfläche grafisch aufwendiger (Web 2.0 Style) gestalteter ist. Es gibt zumeist das komplette Abstract zu jedem Dokument zu lesen und für jedes Dokument ist ein Hinweis vorhanden, wo der komplette Datensatz gefunden werden kann. Allerdings sind mitunter viele Suchergebnisse zu Büchern enthalten, zu denen leider kaum weitere Informationen (nur Autor & Titel) angezeigt werden.[45]
  Zitationen zu einem Dokument werden als Zahl angezeigt, allerdings waren zum Zeitpunkt der Recherche selten und wenn dann nur wenige Zitationen (weniger als bei anderen Diensten zum gleichen Paper) zu finden[46]. Metadaten eines Dokuments können in verschiedenen Ausgabeformaten, bspw. BIBTEX ausgegeben werden.

- **Kommerzielle Dienste**
  Neben den bereits erwähnten (teilweise durchaus kommerziell orientierten) Diensten, gibt es noch eine Anzahl von weiteren Organisationen, die solche und ähnliche Dienste ausschließlich kommerziell anbieten. Der Unterschied zu den bereits Erwähnten, liegt dabei in der Zugänglichkeit, die meist nur gegen Bezahlung und / oder Mitgliedschaft in einer Organisation möglich ist.
  Ein Vergleich des *Web Of Science* mit anderen kommerziellen und nicht-kommerziellen Anbietern findet sich bei [ND06]. Dort ist auch eine Übersicht zu weiteren Zitationsdatenbanken zu finden.
  Zu den bekanntesten rein kommerziellen Anbietern gehören:

  - **Web Of Science**[47]
    Durch das *Web Of Science* System bzw. dem Verwandten *Web Of Knowledge* sind eine Vielzahl von Zeitschriftenartikel recherchierbar. Dadurch, dass ausschließlich Beiträge aus einschlägigen Fachzeitschriften aufgenommen werden, ist die Qualität der einzelnen Ergebnisse sehr gut. Der Dienst deckt ein sehr breites Spektrum von Fachgebieten ab, von Sozialwissenschaften bis zu Technik, Naturwissenschaften und Informationswissenschaften.
    Bekannt ist das *Web Of Science* durch die frühere Mitarbeit von Eugene Garfield, einer der Begründer der modernen Bibliometrie. Von ihm stammt die Idee Zitationen von wissenschaftlicher Literatur methodisch zu erfassen und zu analysieren. Diese Arbeit wird durch

[44] http://academic.live.com/, Stand 2007-12-14
[45] Diese Buchergebnisse schienen im Test oft nicht sehr relevant für die Suche.
[46] Dieses Thema wurde auch in diversen Internetforen diskutiert wie z. B. hier: http://recherchenblog.ch/index.php/weblog/zitierhaeufigkeiten_in_live_search_academic/ (Stand 2008-02-04)
[47] http://scientific.thomson.com/products/wos/, (Stand 2007-12-13)

das *Web Of Science* weitergeführt mit regelmässig veröffentlichen Listen (*Science Citation Index*) und Analysen zur Bedeutung von Autoren und Organisationen. Bekannt sind die Publikationen des *Impact Factor*, der als ein Maß für den wissenschaftlichen Einfluß einer Veröffentlichung angesehen werden kann. Allerdings sind Metriken dieser Art nicht unumstritten, da die genaue Interpretation einer Zitationen nicht berücksichtigt wird (bspw. die Motivation, die hinter der Angabe einer Zitation steht) und die Menge der analysierten Daten begrenzt ist (eine Auswahl von ausgewerteten Zeitschriftenartikel). (Vgl. hierzu [Bru98])
Der Zugang zu diesen Daten ist nur über eine kostenpflichtig zu benutzende Webseite möglich. Viele Universitäten bzw. Bibliotheken haben aber für die interne Nutzung Sammelverträge abgeschlossen.

– **ACM**[48]
In den Fachgebieten der Computer Wissenschaften und Mathematik haben die Veröffentlichungen der *Association for Computing Machinery* großen Einfluß, da viele Tagungen und Konferenzen von dieser Organisation initiiert wurden. Die *ACM* ist als Verein organisiert mit vielen Unterorganisationen (Special Interest Groups) und Kooperationen mit internationalen Fachorganisationen.
Die Publikationen der *ACM* sind in einer digitalen Bibliothek nachweisbar und in den meisten Fällen als Volltext im Download erhältlich. Für Nicht-Mitglieder ist die Nutzung bzw. der Download von Artikeln kostenpflichtig.
Die Dokumente sind im allgemeinen gut aufgearbeitet und sowohl Abstract und Referenzen als auch Zitationen auf ein Dokument sind vollständig extrahiert.[49] Zusätzlich zu den Publikationen der *ACM* sind viele Dokumente von *IEEE*-Arbeitsgruppen sowie Normenwerke verfügbar.

– **Zeitschriftenverlage**
Die meisten wissenschaftlichen Zeitschriftenverlage bieten einen Onlinezugang zu ihren veröffentlichten Artikeln. Dort sind die meisten Verlage dazu übergegangen nicht nur Abstracts aus den Artikeln zu extrahieren, sondern auch die Zitationen und Referenzen zu analysieren. Sehr oft werden auch Autoren separat erfasst um z. B. direkte Links und E-Mailadressen zu den Autoren anbieten zu können oder nach weiteren Arbeiten eines ausgewählten Autors zu suchen. Beispiele hierfür sind: *Wiley Interscience*[50] oder *Scirus*[51] und *Scopus*[52] (beide vom *Elsevier B.V.* Verlag).
In den meisten Fällen sind diese Dienste kostenpflichtig bzw. werden auf Subskriptionsbasis abgerechnet. Allerdings haben sich einige, meist kleinere, Verlage[53] der *Open Access* Bewegung verschrieben und bieten ihre Dienste ohne zusätzliche Kosten für den Nutzer an. Bei diesen Verlagen kommen für die Veröffentlichung meist recht hohe Kosten auf die Autoren zu.

[48] http://portal.acm.org/dl.cfm, (Stand 2007-12-13)
[49] Auf eventuelle Fehler wird dabei hingewiesen.
[50] http://interscience.wiley.com/, Stand 2008-02-06, Internetangebot des Verlags John Wiley & Sons, Ltd.
[51] http://www.scirus.com/, Stand 2007-12-13
[52] http://www.scopus.com/, Stand 2008-02-06
[53] Zum Beispiel die *Public Library of Science*: http://www.plos.org/, Stand 2007-12-13
Veröffentlichungen kosten hier pro Artikel zwischen 1200$ und 2750$, je nach Zeitschrift.

2.3 Vorangegangene Projekte und Vorarbeiten

Dieser Arbeit gingen mehrere Projekte voraus, bei denen einige notwendige Vorarbeiten und Untersuchungen gemacht wurden. Diese Projekte waren / sind Teil des *IR-Thesis* Projekts, dass zum Ziel hatte softwaregestützte Hilfsmittel für die wissenschaftliche Recherche von Studenten und Wissenschaftlern zu entwickeln.
Dabei wurde u. a. eine Software entwickelt für die Rechercheunterstützung bei der Suche nach Literatur in digitalen Bibiotheken:

- Im Rahmen der Veranstaltung *Knowledge and the Web* (am Institut für Wirtschaftsinformatik der HU-Berlin bei Prof. Berendt) Sommer 2005 wurde ein Programm entwickelt, dass nach Literatur unter der Zuhilfenahme des Webfrontends von *CiteSeer* sucht und diese mit einem in PHP geschriebenen Webservice für eine MS-Office Applikation (mittels *Visual Basic*) bereitstellt.
  Eine Datenbank mit Zitationsbeziehungen wurde dabei für eine Analyse von Kozitationsbeziehungen verwendet und eine Clusterung wurde zur Gruppierung der Dokumente verwendet. (Siehe [BDH06])

- Das Programm wurde im Rahmen der Forschung mehrfach angepasst. Dabei wurde zum einen das Datenbankschema geändert und mehr Daten lokal vorgehalten. Eine Version mit einfacher Weboberfläche (statt MS-Office) wurde für eine Evaluation entwickelt.

- Eine Evaluation mit zufällig ausgewählten Nutzern wurde 2007 erstellt und verwendete die Webversion des Programm, kam aber zu eher gemischten Ergebnissen (vgl. [Pap07]).
  Insbesondere die Bedienung (Oberfläche, Reaktionszeiten, etc.) wurde als nicht ausreichend empfunden und die Ergebnisse wurden als schlecht eingestuft. Dabei war nicht zu unterscheiden, ob die Ergebnisse (bzw. die „Nützlichkeit“) als schlecht eingestuft wurden, weil die Bedienung unzureichend war, oder ob die Gruppierungen zu keinem interpretierbaren Ergebnis führten.

- Insbesondere aus den Ergebnissen der Evaluation wurde der Schluß gezogen, dass bei einer Weiterentwicklung des System mehr Wert auf *Usability* gelegt werden muss, damit Anwender einen Nutzen daraus ziehen können.

3 Theoretische Grundlagen

In diesem Kapitel werden die theoretischen und empirischen Grundlagen vorgestellt, mit denen das entwickelte System arbeitet und bewertet werden kann.
Die theoretischen Grundlagen umfassen die Entwicklung und Vorstellung von Klassifikationsverfahren, die in der Lage sind Zusammenhänge in Literatur (Kap. 3.1.1) zu entdecken und Gruppen ähnlicher Dokumente zu bilden (Kap. 3.1.2).

Die empirischen Grundlagen (Kap. 3.2) beschreiben die Möglichkeiten einer Nutzerevaluation, um Schlussfolgerungen über die Verwendbarkeit des Systems und weiterer Systemanforderungen zu erlangen.

3.1 Analyseverfahren

Dieser Abschnitt stellt Analyseverfahren für Daten vor, wie sie in typischen Literaturdatenbanken anzutreffen sind. Die Anwendung bibliographischer und textbasierter Verfahren, um ein numerisches Maß für die Ähnlichkeit zweier Dokumente zu definieren, wird in Abschnitt 3.1.1 beschrieben.
Um aus den hier vorgestellten Ähnlichkeitsmaßen berechneten Daten Gruppen sich ähnlicher Dokumente zu bestimmen, hat sich das Verfahren der Clusterung bei verschiedenen, ähnlich gelagerten Anwendungsfällen, bewährt (vgl. [Jan07, ZK02]). In Abschnitt 3.1.2 werden verschiedene solcher Clusterverfahren vorgestellt, mit denen Gruppen gebildet werden können.
In Abschnitt 3.1.3 folgt eine Darstellung der möglichen Kombination mehrerer Verfahren.
3.1.3. Abschliessend folgt eine kurze Erörterung von Möglichkeiten der Generierung von Stichworten bzw. Beschreibungen für Gruppen von Dokumenten.

3.1.1 Algorithmen und Analyseverfahren

Grundlage der Gruppenbildung von Dokumenten ist die Analyse und Erkennung von „ähnlichen“[54] Dokumenten sowie die Zusammenfassung der als „ähnlich“ erkannten Dokumente. Eine anschließende Analyse der in einer Gruppe zusammengefassten Dokumente kann einem Benutzer Aufschluss oder zumindest Anhaltspunkte über die enthaltenen Dokumente geben.
Diese Aufgaben bedingen, im vorgesehenen Einsatzgebiet, die Verwendung von automatischen, nicht (direkt) beaufsichtigten Verfahren (*non-supervised*), die ausschließlich mit den vorhandenen Daten operieren und die benötigten neuen Informationen daraus ermitteln.
Die in Kapitel 2.2.2 und 2.2.3 vorgestellten Systeme basieren auf bzw. nutzen einige der hier vorgestellten Methoden um bestimmte Funktionen zu realisieren, wie etwa die Suche nach ähnlichen Dokumenten.

[54] Der Begriff „ähnlich“ soll hier ausschließlich auf einen inhaltlichen bzw. thematischen Zusammenhang hindeuten. Ein solcher Zusammenhang kann von verschiedenen Personen natürlich unterschiedlich bewertet werden, worüber nur eine Nutzerstudie Auskunft geben kann.

Für das Ziel, eine Aussage über die „Ähnlichkeit“ von Dokumenten machen zu können, werden zunächst verschiedene Metriken vorgestellt, die auf die hier vorhandenen Dokumente angewendet werden können. Dabei wird im Grundsatz unterschieden zwischen bibliographie-basierten / Auswertung des Zitationsgraphen und Text-basierten Ähnlichkeitsmaßen.

3.1.1.1 Bibliometrische Verfahren – Auswertung von Zitationsgraphen

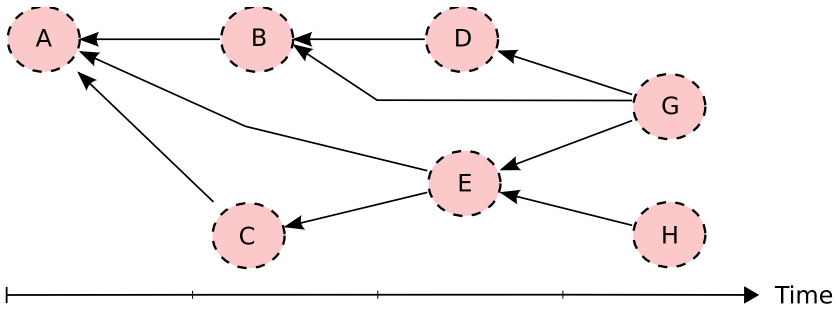


Abbildung 3.1: Einfacher Zitationsgraph als Beispiel

Die vorgestellten Verfahren basieren auf den bibliographischen Metadaten eines Dokumentes, genauer: auf den von den Autoren verwendeten bzw. angegebenen Referenzen im Dokument. Ebenfalls möglich wäre die Auswertung sozialer Beziehungen, z. B. die Zusammenarbeit von Autoren (sog. Koautoren-Analyse) oder Zugehörigkeiten zur selben Organisation (vgl. [Jan07, S. 89]).[55] Diese Art von Beziehungen werden jedoch hier nicht betrachtet. Als Beispiel einer solchen Analyse kann [PVR91] verwendet werden.

Unter dem Begriff Referenzen wird hier zum einen die Liste verwendeter anderer Dokumente (Bibliographie, Literaturverzeichnis) verstanden, als auch (gekennzeichnete) direkte Zitate innerhalb des Dokuments. Zur Vereinfachung wird im Folgenden immer von *Zitation* gesprochen, auch wenn es sich formalsprachlich um eine Referenz (also ein indirektes Zitat) zu einem anderen Dokument handelt.
Ein Dokument d_i, das ein Dokument d_j zitiert, wird in dieser Beziehung als abhängiges (auch abgeleitetes) Dokument von d_i bezeichnet und d_j wird als Vorfahre oder Vorgänger von d_i bezeichnet.

Veranschaulicht wird das Mittel der Zitationsanalyse an einem sog. Zitationsgraphen (Abb. 3.1) $G = (V,E)$, der Zitationszusammenhänge von Dokumenten anzeigt (vgl. [Gar79]). In diesem Graph bilden die Dokumente d die Knoten V und die Kanten werden durch die Zitierungen als gerichtete Kanten $E \subset V \times V$ gebildet. Des Weiteren ist ein solcher Graph (im Allgemeinen) azyklisch.[56]
Im Beispiel (Abb. 3.1 und Formel 3.1) zitiert das Dokument d_G die Dokumente d_B, d_D und d_E, während das Dokument d_A von den Dokumenten d_B, d_C und d_E zitiert wird. Mathematisch können diese

[55] Solche Untersuchungen geben Aufschluß über die Verteilung von Wissenschaftlern in der Welt oder es lassen sich die Anteile eines Landes an wissenschaflicher Forschung aufschlüsseln.
[56] Ausnahmen sind möglich. Beispiel: Ein Dokument, dass in zwei Teilen veröffentlicht wird und beide Teile zitieren einander. In den Weiteren Betrachtungen wird diese Besonderheit aber keine Rolle spielen.

Zusammenhänge kompakt in einer sog. Adjazenzmatrix gespeichert werden, die für jeden Knoten des Graphen die Nachbarschaftsbeziehungen darstellt:

$$\mathbf{M} = \begin{array}{c} \\ d_A \\ d_B \\ d_C \\ d_D \\ d_E \\ d_F \\ d_G \end{array} \begin{array}{c} \begin{array}{ccccccc} d_A & d_B & d_C & d_D & d_E & d_F & d_G \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \end{array} \tag{3.1}$$

Diese Matrix hat, insbesondere bei Zitationsbeziehungen, die Eigenschaft dünn besetzt zu sein (eine sog. sparse Matrix), also nur wenige Einträge zu haben (bzw. sehr viele 0-Einträge). Außerdem können bei Zitationsgraphen grundsätzlich nur die Werte 1 und 0 (zitiert – zitiert nicht) auftreten, was auch als binäre Matrix bezeichnet wird.[57] Mehrfachzitierungen (ein Dokument d_i wird von einem Dokument d_j an mehreren Stellen im Text referenziert) werden dadurch als eine Zitation gewertet.

Um nun auf dieser Basis inhaltlich zusammenhängende Dokumente zu erkennen, werden die folgenden Zusammenhänge angenommen: (vgl. [Sma73, S. 265–268], [Gar01])

- **Direkte Zitation**
  Zwei Arbeiten, die durch direkte Zitation verbunden sind, sind auch miteinander inhaltlich verbunden.
- **Kozitation**
  Zwei Arbeiten, die gemeinsam von einer dritten Arbeit zitiert werden, sind inhaltlich miteinander verbunden.
- **Bibliographische Kopplung**
  Zwei Arbeiten, die gemeinsam das oder die gleichen Dokumente zitieren, sind inhaltlich miteinander verbunden.

Zusätzlich wird angenommen, dass das Maß für den inhaltlichen Zusammenhang, von der Häufigkeit des Auftretens eines solchen Zusammenhangs abhängig ist; also ein höherer inhaltlicher Zusammenhang besteht, je häufiger bspw. zwei Dokumenten von einem dritten Dokument gemeinsam zitiert werden.

Unter Verwendung einer Datenbasis mit Zitationsdaten, lassen sich nun diese Verhältnisse analysieren.

Direkte Zitation

Die einfachste Variante des Zitationszusammenhangs basiert direkt auf der Adjazenzmatrix mit direkten Zitationswerten. Hier wird für jedes Dokument die Menge an Referenzen auf ein anderes angegeben und anschließend ausgewertet. Dabei können sowohl die eingehenden (Dokument wird zitiert), als

[57] Dies ist natürlich erstmal eine sehr einfache Form (bzw. Grundform) der Modellierung des Zitationszusammenhangs. Die Matrix kann z. B. durch Umformungen oder statistische Verfahren auch andere Werte annehmen, was in späteren Schritten berücksichtigt werden muss. Hier soll zunächst erstmal die „Grundform“ verwendet werden.

auch die ausgehenden Zitationen (Dokument zitiert) von einem Dokument bewertet werden. Im Beispiel oben (Abb. 3.1) ist das Verhältnis von Dokument d_G und d_D als eingehende Zitation für d_D zu werten und als ausgehende Zitation für d_G.
Dieses Verfahren ist sehr beschränkt in dem Sinne, dass nur Zitationen zwischen den aktuell betrachteten Dokumenten berücksichtigt werden (was nur bei der Analyse der gesamten Datenmenge kein Problem wäre) und weitergehende Zusammenhänge, wie z. B. gemeinsam zitierte oder zitierende Dokumente nicht in die Wertung einfließen. Außerdem kann bei einer binären Adjazenzmatrix jede Zitation nur mit eins bewertet werden, was eine Auswertung des Maßes des inhaltlichen Zusammenhangs schwierig macht. Daher sollten, wenn möglich, andere Verfahren gewählt werden.

Kozitation – Co-citation (CC)

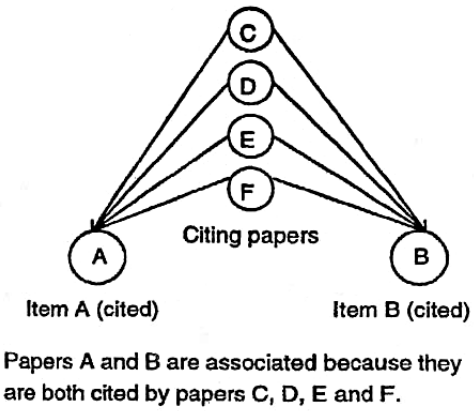


Abbildung 3.2: Kozitationsbeziehung – Quelle: [Gar01]

Bei der Analyse der Kozitation (Abb. 3.2) wird die gleichzeitige Zitation zweier Dokumente durch ein drittes Dokument betrachtet.
Um zwei Dokumente also als „ähnlich“ gelten zu lassen, muss ein bzw. mehrere dritte Dokumente diese beiden Dokumente in ihrer Referenzliste aufführen. Eine Konsequenz (und auch das größte Manko) ist, dass neuere Dokumente durch diese Analyse weit weniger berücksichtigt werden können als ältere Dokumente, da für den messbaren Zusammenhang zwischen den beiden betrachteten Dokumenten ein drittes, neueres Dokument existieren muss. Wenn die beiden betrachteten Dokumente sehr neu sind, dann ist die Wahrscheinlichkeit geringer, dass ein solches, noch neueres Dokument, existiert. Umgekehrt wird es bei älteren Dokumenten mit hoher Wahrscheinlichkeit eine höhere Anzahl von zitierenden Dokumenten geben; die Anzahl von Kozitationen muss aber keineswegs im gleichen Maße steigen, was möglicherweise zu einer anderen Bewertung des inhaltlichen Zusammenhangs durch die Kozitationsanalyse führt. Bei neueren Dokumenten kommt noch hinzu, dass diese möglicherweise einen neuen Ansatz im Fachgebiet verfolgen, so dass eine Adaption anderer Wissenschaftler (messbar in Kozitationen) mehrere Jahre braucht (vgl. [Por77]).
Daraus folgt, dass das hier vorgestellte Evaluierungsmaß für die Kozitation zeitabhängig ist, da die Anzahl von Kozitationen indirekt abhängig vom Publikationsjahr der beiden betrachteten Dokumente ist. Sie eignet sich außerdem nur eingeschränkt für die Analyse von sehr neuen Dokumenten.
Die Kozitationsanalyse wird oft bei Untersuchungen der thematischen und zeitlichen Entwicklung ei-

ner Wissenschaftsdisziplin verwendet, wie sie zum Beispiel vom ISI[58] herausgegeben wird. Außerdem wird dieses Verfahren als geeignet zur Evaluierung der Relevanz und Qualität von wissenschaftlichen Arbeiten angesehen, da es grundsätzlich auf dynamischen Verhältnissen (die Anzahl Dokumente, die auf ein gegebenes Dokument verweisen) beruht und damit potentiell die Meinungen bzw. Relevanzeinschätzungen weiterer Wissenschaftler einfließen. Solche Untersuchungen, die im Allgemeinen zum Fachgebiet der *Szientometrie* gerechnet werden, wurden maßgeblich durch die Analyse der Zitationshäufigkeiten vorangebracht. (Vgl. [Gar01])

Bibliographische Kopplung – Bibliographic Coupling (BC)

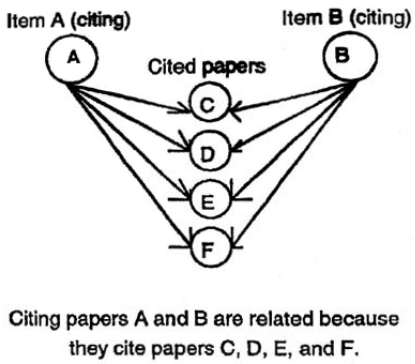


Abbildung 3.3: Bibliographische Kopplung – Quelle: [Gar01]

Die Untersuchung der Bibliographischen Kopplung zweier Dokumente, auch als *Koreferenz* bezeichnet, beruht ähnlich der Kozitationsanalyse auf der Beziehung zu einem dritten Dokument, welches von den beiden betrachteten Dokumenten gleichzeitig zitiert wird. (Abb. 3.3)
Der Vorteil dieser Analyse ist, dass auch neueste Dokumente ausreichend berücksichtigt werden können. Allerdings ist die Bibliographische Kopplung ein statisches Verhältnis, da sich die angegebenen Referenzen eines Dokumenten naturgemäß nicht mehr ändern. Dieser Fakt und die Tatsache, dass die Referenzen (und damit die errechnete Dokumentenähnlichkeit zu einem anderen Dokument) nur durch den Autor eines Dokuments selbst angegeben wird, machen die Analyse der Bibliographischen Kopplung ungeeignet für die Evaluation von wissenschaftlicher Arbeit. Für den Einsatzzweck als Indikator eines inhaltlichen Zusammenhangs zwischen zwei Dokumenten ist die Bibliographische Kopplung allerdings gut geeignet, da sie insbesondere neueste Dokumente besser (als die Kozitationsanalyse) berücksichtigen kann und man annehmen kann, dass Autoren die für sie relevante Literatur korrekt angeben.

Graphenanalyse

Die drei obigen Verfahren berücksichtigen im Zitationsgraphen ausschließlich die dem betrachteten Dokument am nächsten liegenden Dokumente, also die direkten Nachbarn. Darüber hinaus ist natürlich auch eine weitere Analyse der weiter entfernten Dokumente durch eine direkte Auswertung des

[58] Institute for Scientific Information (ISI) – Von Eugene Garfield gegründet für die Analyse von bibliographischen Informationen, ist es heute als *Thomson Scientific* Teil des *Thomson-Reuters* Konzerns. Siehe auch Seite 19.

Zitationsgraphen mittels Graphenanalyse denkbar und möglich.
Ein oft verwendeter Graphenalgorithmus für diesen Zweck ist der „Hubs and Authorities“ (oder auch *Hypertext Induced Topic Search* – HITS) Algorithmus (vgl. [Kle99]), der allerdings in der ursprünglichen Version auf die Suche nach einem Thema (sog. monothematische Analyse) abgestimmt ist. Ein weiterer bekannter Algorithmus dieser Art ist der PageRank Algorithmus, auf dem die Suchmaschine *Google* beruht. Dieses Verfahren basiert auf einem schon früher veröffentlichten Algorithmus für bibliographische Daten (vgl. [PN76] und [Kle99, S. 618]).
Eine Erweiterung des HITS Algorithmus ist möglich durch die numerisch aufwendige Hauptkomponentenanalyse (PCA), mit der weitere Themengebiete erschlossen werden können (sog. polythematische Analyse). Allerdings werden durch das unpassende Statistikmodell in HITS auch unrelevante Themen und Zitierraten erzeugt, die in den realen Daten nur schwer nachzuvollziehen sind. (Vgl. [CC00, Ste07])
Ein durch ein anderes Statistikmodell besser angepasster Algorithmus ist der von [CC00] vorgeschlagene *Probalistic Hypertext Induced Topic Search* – PHITS Algorithmus, der durch Suche nach mehreren Themengebieten bessere Ergebnisse zu liefern verspricht. In [Ste07] findet sich eine komplexere Beschreibung und beispielhafte Implementierung.

Aufgrund der hohen Komplexität und aufwendigen numerischen Umsetzung wird die Graphenanalyse im Folgenden nicht weiter betrachtet werden.

Numerische Ähnlichkeitsmaße

Es gibt verschiedene Möglichkeiten die Ähnlichkeiten zwischen zwei Dokumenten mit einem konkreten numerischen Wert auszudrücken. Die einfachste Variante ist das Abzählen von Zitationen bzw. Referenzierungen nach Kozitation und / oder Bibliographischer Kopplung. Der Nachteil dieser Methode ist, dass der Wert direkt abhängig von der Größe der Datenbank ist und Dokumente, die oft zitiert werden, bevorteilt würden. Ein errechneter Ähnlichkeitswert aus diesem Maß wäre nicht normiert und kaum vergleichbar mit einem anderen Ähnlichkeitswert.
Eine bessere, normalisierte Variante für die Messung des Zitationszusammenhangs als Ähnlichkeitsmaß zweier betrachteter Dokument wurde schon von [Sma73, S. 269] bzw. E. Garfield formuliert als das Verhältnis der Menge der nach Kozitation bzw. Bibliographischer Kopplung mit diesen beiden Dokumenten zusammenhängenden Dokumente und der Gesamtzahl der Zitationen zu diesen Dokumenten. Dieses Verhältnis ist unter dem Namen *Jaccard-Quotient* bekannt.

Formal ausgedrückt stellt sich dieses Verhältnis so dar:
Sei D die Menge der Dokumente $D = \{d_1, \ldots, d_n\}$ und $d_i, d_j \in D$. Ferner sei $A = \{x | x \text{ cites } d_i\}$ und $B = \{x | x \text{ cites } d_j\}$, also die Menge der Dokumente, die d_i bzw. d_j zitieren. Damit ist also $A \cap B$ die Menge der Kozitationen von d_i und d_j. Das Ähnlichkeitsmaß nach Jaccard für die Kozitation sieht dann so aus:

$$\text{sim}_{CC_{Jaccard}}(d_i, d_j) = \frac{|A \cap B|}{|A \cup B|}. \tag{3.2}$$

Diese Formulierung entspricht der ursprünglichen, einfachen Variante des Jaccard-Quotienten. Die erweiterte Version des Jaccard-Index ist auch unter dem Namen Tanimoto-Quotient bekannt und ist eine Anpassung für nicht-binäre Ausgangsverhältnisse.
Für den hier betrachteten Fall einer einfachen binären Zitationsbeziehung, ist dieses Maß identisch zum Jaccard-Index, lässt aber eine Vereinfachung zu:

$$\text{sim}_{CC_{Tanimoto}}(d_i, d_j) = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}. \tag{3.3}$$

Das Cosinus-Maß nach Salton (vgl. [SM83]) ist rechnerisch etwas aufwendiger. Hier werden die Zitationen als Vektor betrachtet und das betrachtete Maß ist durch den Cosinus des Winkels, zwischen diesen Vektoren gegeben:

$$\text{sim}_{CC_{Salton}}(d_i, d_j) = \cos_{CC}(d_i, d_j) = \frac{A \cdot B}{\|A\| \cdot \|B\|} \tag{3.4}$$

Für die Betrachtung von Zitationen (im binärer Fall), reduziert sich diese Formel auf:

$$\text{sim}_{CC_{Salton}}(d_i, d_j) = \frac{|A \cap B|}{\sqrt{|A||B|}} \tag{3.5}$$

Die Frage, welches dieser Bewertungsmaße sich für die Literaturanalyse besser eignet, ist allerdings in der Wissenschaft nicht endgültig geklärt (vgl. [Ley08, Ham89, Gmü03]).

Die Berechnung der Ähnlichkeit nach Bibliographischer Kopplung erfolgt analog, allerdings mit der unterschiedlichen Definition von A und B:
$A = \{x|x \text{ is cited by } d_i\}$ und $B = \{x|x \text{ is cited by } d_j\}$.

Neben den hier vorgestellten Maßen gibt es noch weitere, die hier aber nicht weiter behandelt werden sollen, da sie für die Auswertung von Zitationsbeziehungen keine besseren Ergebnisse ermöglichen (vgl. [Ley08, Gmü03]). Dazu gehören z. B.: *Pearson Correlation Coefficient* oder der *Probalistic Activity Index*, die eher für eine vergleichende Analyse einer sehr begrenzten Autorenmenge geeignet sind (vgl. [Gmü03, S. 30]).

Ein weiteres Maß, entwickelt von R. Amsler, welches in [BE80] genauer evaluiert wurde, soll hier noch erwähnt werden, da dieses versucht Kozitation und Bibliographische Kopplung in einem Maß zu vereinen. Bei diesem Maß wird versucht die absolute Zahl an Verbindungen L_i zu einem Dokument d_i (sowohl zitierte, als auch zitierende Dokumente) zu berücksichtigen. Dieses Maß bringt allerdings die Nachteile der Kozitationsanalyse mit sich und kam in der Evaluation ([BE80]) nur auf geringfügig andere, aber vor allem uneinheitlichere, Ergebnisse als die Analyse der Bibliographischen Kopplung alleine.

$$\text{sim}_{Amsler}(d_i, d_j) = \frac{|L_i \cap L_j|}{|L_i \cup L_j|} = \frac{|(A_i \cup D_i) \cap (A_j \cup D_j)|}{|(A_i \cup D_i) \cup (A_j \cup D_j)|}, \tag{3.6}$$

wobei $A_i = \{x|x \text{ is cited by } d_i\}$ (von d_i zitierte Dokumente) und $D_i = \{x|x \text{ cites } d_i\}$ (Dokumente, die d_i zitieren).

Weitere Arten der Gewichtung und Verbesserungen

Auch wenn die Messung der bibliographischen Ähnlichkeit durch den *Jaccard-Quotienten* oder mit dem *Salton*-Maß in der Wissenschaft allgemein anerkannte Technik ist, ergeben sich doch Möglichkeiten bestimmte Effekte minimieren zu wollen. Zu diesen Effekten gehören bspw. die Verteilung der Ähnlichkeitswerte. Diese ist im Allgemeinen durch die oben vorgestellten Maße sehr uneinheitlich. In einer „allgemeinen“ Menge von Dokumenten überwiegen sehr viele Ähnlichkeitswerte mit sehr kleinen Werten und nur sehr wenig hohe Werten (vgl. [Jan07, S. 122f]). Ein anderer Effekt liegt in der verschiedenen Zitierweise von Autoren begründet. Die bisherigen Modelle berücksichtigen nur eingeschränkt die Menge der Zitationen, die von einem Dokument ausgehen. Diskutabel ist bspw., ob angegebene

Referenzen eines Dokumentes mit sehr vielen Referenzen anders zu werten sind, als Referenzen eines Dokumentes, das nur wenige, möglicherweise wichtigere Dokumente, angibt (vgl. [Gmü03, S. 29]).

Es ergeben sich bei Beibehaltung der grundsätzlichen Zusammenhänge verschiedene Möglichkeiten:

- In [Gmü03, S. 37] wird statt der im *Jaccard-Quotienten* vorgesehenen Division mit der absoluten Zahl der Zitationen, eine Division durch den Mittelwert der Zitationen vorgeschlagen:

$$\text{sim}(d_i, d_j) = \frac{|A \cap B|}{\frac{1}{2} \cdot (|A| + |B|)}. \tag{3.7}$$

  Dies entspricht dem sog. *Dice* Koeffizienten, der grundsätzlich dem *Jaccard-Koeffizienten* abgeleitet ist und diesem in den Ergebniswerten ähnlich ist.

- In [Gmü03, S. 40f] wird ein weiteres Maß *CoCit-Score* vorgeschlagen, dass in der dortigen Evaluation gute Ergebnisse erbrachte. Dabei wird die mittlere Zitationshäufigkeit und die minimale Zitationshäufigkeit kombiniert:

$$\text{sim}(d_i, d_j) = \frac{(|A \cap B|)^2}{\min(|A|, |B|) \cdot \frac{1}{2} \cdot (|A| + |B|)}. \tag{3.8}$$

- In [BDH06] wird vorgeschlagen die Publikationsjahreszahlen einzubeziehen. Danach wird für die Berechnung der Gesamtzahl der Zitationen ($|A|$ und $|B|$) nur die Anzahl der Zitationen berücksichtigt, die auf beide Dokumenten zutreffen können.
  Das Ähnlichkeitsmaß nach *Jaccard* ändert sich in der Weise:

$$\text{sim}(d_i, d_j) = \frac{|A \cap B|}{|\hat{A}| + |\hat{B}| - (|A \cap B|)}. \tag{3.9}$$

  mit $\hat{A} = \{x | x \text{ cites } d_i \text{ and pubyear}(d_i) \geq \max(\text{pubyear}(d_i, d_j))\}$, B analog.

- Eine Ansatz, unterschiedlich ausgeprägte Literaturlisten zu berücksichtigen, wird in [SS85, S. 393] vorgeschlagen, mit einem anteiligen Zitationskoeffizienten *Fractional Citation Count*, der eine Zitation eines Dokumentes mit dem Kehrwert der Anzahl aller Zitationen dieses Dokumentes gewichtet.
  Dieses Maß, hat in der Literatur keine hohe Verbreitung gefunden. In [SS85], wird der Haupteinsatzzweck auch eher auf die Auswahl von Dokumenten gelegt.

Grundsätzlich ist noch anzumerken, dass die meisten Untersuchungen hierzu auf Basis der Kozitationsanalyse gemacht wurden. Inwieweit sich diese Ergebnisse auf die Analyse der Bibliographischen Kopplung übertragen lassen, ist noch weitgehend ungeklärt.

Aufbau der Ähnlichkeitsmatrix

Um eine weitere Auswertung bspw. mit Clusterverfahren zu ermöglichen, müssen die Ähnlichkeitswerte für jedes Dokument zu jedem anderen Dokument berechnet werden. In der Praxis wird eine *Dokument–Dokument* Matrix $\mathbf{S} \in \mathbb{R}^{n \times n}$ (sog. *Ähnlichkeitsmatrix*) aufgebaut, die vollbesetzt, symmetrisch und mit nicht negativen Werten besetzt ist:

$$\mathbf{S} = \begin{bmatrix} 1 & & & \\ \mathrm{sim}_{2,1} & 1 & & \\ \vdots & \ddots & 1 & \\ \mathrm{sim}_{n,1} & \dots & \mathrm{sim}_{n,n-1} & 1 \end{bmatrix} \tag{3.10}$$

Dabei entspricht $\mathrm{sim}_{i,j}$ = „Ähnlichkeit von Dokument d_i und d_j“.
Statt Ähnlichkeitswerten können auch Distanzwerte verwendet werden, wie es bspw. bei [Jan07] vorgeschlagen wird (was natürlich in weiteren Auswertungen beachtet werden muss).
Ein Ähnlichkeitsmaß, wie oben vorgestellt, erzeugt im Allgemeinen normierte Werte $x_{ij} \in [0,1]$. Bei anderen Ähnlichkeitsmaßen sollte die Matrix explizit normiert werden um das anschließende Clustern zu ermöglichen und um Vergleichbarkeit zu gewährleisten.[59] Da ein Dokument zu sich selbst immer maximal ähnlich ist, ist die Hauptdiagonale durchgängig mit 1 besetzt.

[59] Es ist bei Clusterprogrammen üblich die Matrix im ersten Schritt zu normalisieren, was bei einer nicht normalisierten Ähnlichkeitsmatrix möglicherweise zu falschen oder irregulären Ergebnissen führen könnte.

3.1.1.2 Textbasierte Verfahren

Im Gegensatz zu den bibliographiebasierten Verfahren arbeiten die textbasierten Verfahren mit dem textuellen Inhalt eines Dokuments. Der Vorteil dieser Verfahren ist, dass sie auf einem Grundsatz von Daten (das Dokument selbst) arbeiten, welcher theoretisch immer vorhanden ist, während andere Metainformationen wie Referenzen erst aus dem Dokument extrahiert werden müssen. Allerdings ist dieser Vorteil gerade bei Literaturdatenbanken, die ausschließlich Metadaten speichern (siehe im Kapitel 2.2) nicht gegeben. Dort bleiben nur die bibliographischen Methoden, wenn nicht zumindest das Abstract oder ein Textauszug vorhanden ist.

Es gibt verschiedene Ansätze mit den Wörtern eines Dokuments umzugehen. Neben einer aufwendigen strukturellen Satzbauanalyse (über sog. *part-of-speech tagger*), bspw. um in der Bedeutung wichtige Hauptwörter zu erkennen, ist das sog. *Bag-of-Words* (BOW) Format sehr verbreitet. Bei dieser Art der Dokumentenanalyse wird das Dokument in einzelne Wörter, auch *Terme* genannt, zerlegt, was grammatikalische Informationen sowie Informationen zur Reihenfolge des Auftretens im Text verwirft. (Vgl. [BFS03, S. 83])
Die einfachste Variante des BOW gewichtet jedes Wort w_j aus dem Vokabular V mit einer binären Funktion: $w_j \in \{0,1\}$ (Term kommt nicht vor / Term kommt vor). Eine erweiterte Variante nutzt die Menge der Vorkommen des j-ten Wortes als Gewichtung. Zusätzlich können beliebige andere Gewichte verwendet werden, bspw. um Terme eines bestimmten Dokuments oder Dokumententeils besonders zu behandeln.[60]
Formale Beschreibung: Ein Dokument $d_i \in D$ wird beschrieben durch die Menge der Termgewichte ω_k, mit $k \in \{\text{Terme von } d_i\}$. Dadurch entsteht ein sog. Termvektor $\vec{\mathbf{d}}_i \in \mathbb{R}^{|V|}$ mit den ermittelten Termgewichten ω_k, $k \in [1..m]$ und $m = |V|$ der Größe des Vokabulars aus D:

$$\vec{\mathbf{d}}_i = \begin{bmatrix} \omega_{1,i} \\ \vdots \\ \omega_{m,j} \end{bmatrix}. \tag{3.11}$$

Die Zusammenfassung aller beteiligten Termvektoren in einer Matrix führt zu der so. *Term-Document Matrix*, wobei jede Spalte der Matrix ein Dokument repräsentiert und durch den Termvektor $\vec{\mathbf{d}}_i$ gebildet wird. Jede Zeile $\vec{\mathbf{t}}_j^{\,T}$ der Matrix repräsentiert die Verteilung eines Terms ω_j zu den einzelnen Dokumenten $d \in \{d_1, \ldots, d_n\}$:

$$\vec{\mathbf{t}}_j^{\,T} = \begin{bmatrix} \omega_{j,1} & \ldots & \omega_{j,n} \end{bmatrix}. \tag{3.12}$$

Die entstehende Matrix $\mathbf{A}$ wird auch mit dem mathematischem Begriff *Vector-Space-Model* (VSM) bezeichnet: (vgl. [SM83])

$$\mathbf{A} = \begin{array}{c} \\ \vec{\mathbf{t}}_j^{\,T} \rightarrow \end{array} \begin{array}{c} \vec{\mathbf{d}}_i \\ \downarrow \\ \begin{bmatrix} \omega_{1,1} & \ldots & \omega_{1,n} \\ \vdots & \ddots & \vdots \\ \omega_{m,1} & \ldots & \omega_{m,n} \end{bmatrix} \end{array} \tag{3.13}$$

Diese Matrix $\mathbf{A}$ wird in den meisten Fällen sehr dünn besetzt sein, da die Anzahl der Terme bei typischen Dokumenten deutlich höher ist als die Anzahl der Dokumente: $|V| \gg |D|$. Dies ermöglicht eine sehr effiziente Speicherung in Computersystemen.

[60] Dies kann sinnvoll sein, um z. B. Überschriften, Stichworte oder ähnliches im BOW besonders auszuzeichnen.

Um zwei Dokumente, gegeben durch die Termvektoren $\vec{\mathbf{d}}_i$ und $\vec{\mathbf{d}}_j$ miteinander zu vergleichen und eine Ähnlichkeit dieser auszudrücken, ist die Verwendung des Cosinus des Winkels zwischen den beiden Vektoren zweckmäßig. Dieses Maß ist durch das Skalarprodukt (Dot-Produkt) der beiden Vektoren leicht errechenbar und produziert die gewünschten normierten Ähnlichkeitswerte $\in [0,1]$ (siehe Abschnitt 3.1.1.1).
Mit $\vec{\mathbf{d}}_i$ und $\vec{\mathbf{d}}_j$ den Termvektoren zu Dokument d_i und d_j berechnet sich die Ähnlichkeit wie folgt:

$$\mathrm{sim}_{text}(d_i,d_j) = \cos(d_i,d_j) = \frac{\vec{d}_i \cdot \vec{d}_j}{\|\vec{d}_i\|_2 \, \|\vec{d}_j\|_2} \tag{3.14}$$

Diese Verfahrensweise kann verbessert werden durch veränderte Gewichtungen der einzelnen Terme, Ausfiltern von Termen, deren Informationsbeitrag gering ist und Reduzierung von grammatikalischen Formen, bzw. Rückbildung auf den Wortstamm für die enthaltenen Terme:

- Unterschiede in Klein- und Großschreibung werden am einfachsten durch Umwandlung aller Wörter in Groß- oder Kleinbuchstaben behandelt. Dadurch reduziert sich die Menge der Terme bei nur sehr geringem Informationsverlust.

- In jedem Text gibt es Wörter, die als Füllwörter gelten oder aus grammatikalischen Gründen enthalten sind und damit nur sehr gering zum Informationsgehalt des Textes beitragen. Dazu gehören z. B. Artikel oder Bindewörter wie „und“. Im Allgemeinen ist es praktikabel, diese Wörter mit Hilfe einer sog. Stoppwortliste auszufiltern (vgl. [BYRN+99, S. 167]). Diese Stoppwortliste ist natürlich sprachabhängig. Daher sollte sichergestellt sein, dass nur Dokumente einer Sprache verarbeitet werden, bzw. die Dokumentensprache für die korrekte Anwendung einer solchen Liste gekennzeichnet ist.

- Wortbeugungen führen zu einer Vielzahl von morphologischen Varianten, die auf das gleiche Grundwort zurückzuführen sind. Damit erhöht sich nicht nur die Dimension der Term-Document Matrix, sondern die Ähnlichkeitsberechnung wird auch ungenauer. Eine Lösung bietet sich hier durch den Einsatz sog. *Stemmer* an, die regel- oder lexikonbasiert verschiedene Wortformen auf entsprechende Grundformen abbilden.
  Dabei gibt es eine Vielzahl von Ansätzen (eine Übersicht bringt [Wik08b]), bewährt haben sich reine regelbasierte Verfahren wie der *Porter-Stemmer*[61] (vgl. auch [BFS03, S. 82]), rein lexikonbasierte Verfahren oder Ansätze, die eine regelbasierten Auswertung mit einem lexikonbasierten Verfahren kombinieren um Grenzfälle zu behandeln. Eine grammatikalisch korrekte, aber aufwendige, Variante wird als „Lemmatisierung“ bezeichnet und nutzt meist zusätzliche Informationen aus der Anwendung eines „*part-of-speech taggers*“. (Vgl. [MHSW03])

  Die Anwendung eines Stemming-Algorithmus, insbesondere des *Porter-Stemmer*, birgt möglicherweise Schwierigkeiten in der Interpretation, wenn die dadurch erzeugten Terme einem Benutzer angezeigt werden sollen. Da diese Terme (je nach verwendetem Algorithmus) sehr oft grammatikalisch oder syntaktisch unkorrekte Wortstämme beinhalten, kann dies zu Unverständnis führen. Einen Ausweg bieten „weniger eingreifende“ Algorithmen, wie z. B. der *UEA-Lite Stemmer*[62] (für die englische Sprache).
  Eine Übersicht und Einschätzung der Nützlichkeit von Stemming in IR bietet [GHS01], darüber

[61] http://tartarus.org/martin/PorterStemmer/
[62] http://fizz.cmp.uea.ac.uk/Research/stemmer/

hinaus wird vermutet, dass der Einsatz von *Latent Semantic Analysis* (LSA) (siehe weiter unten, Seite 34) das Stemming nicht oder nur in einigen Fällen notwendig macht (vgl. [DDF+90] und [BYRN+99, S. 168]).

- Die unterschiedliche Länge von Dokumenten führt zu ungleichen Gewichtungen der Dokumente durch die verschiedenen Häufigkeiten eines Terms. Da das Vorkommen eines Wortes w bei kürzeren Texten schon statistisch gesehen niedriger sein muss, bietet sich eine Normalisierung der Worthäufigkeiten über die Länge des Dokumentes (Menge der Terme in d_i) an.
  Dieser Wert wird im Allgemeinen als *Term frequency* (TF) bezeichnet (vgl. [BFS03, S. 84]). Mit $D = \{d_1, \ldots, d_n\}$, $d_i \in D$ und n_{ki} mit den Vorkommen des Wortes w_k in Dokument d_i, wird TF berechnet:
  $$\mathrm{TF}_{ik} = \frac{n_{ik}}{|d_i|}. \tag{3.15}$$

  Das unterschiedliche Auftreten von Termen in vielen Dokumenten lassen einige Terme wichtiger (bzw. mit höherem TF Wert) oder unbedeutender erscheinen, als eigentlich adäquat wäre. Der Effekt lässt sich versinnbildlichen, dass Wörter, die in sehr vielen oder allen Dokumenten verwendet werden, diese Wörter für die Unterscheidung innerhalb dieser Dokumentenmenge nicht sehr aussagekräftig macht.
  Eine übliche Methode ist es daher, den TF-Wert mit einem Dämpfungsfaktor zu kombinieren, der aus dem Logarithmus des Verhältnisses der Dokumentanzahl n und der Anzahl der Dokumente n_k, die einen Term w_k mindestens einmal enthalten, errechnet wird. (Vgl. [BFS03, S. 84][63])
  $$\mathrm{IDF}_k = \log \frac{n}{n_k}. \tag{3.16}$$

  Damit kann ein Term w_k mit der Gewichtung ω_k im Termvektor für ein Dokument d_i mit dem Produkt
  $$\omega_{ik} = \mathrm{TF}_{ik} \cdot \mathrm{IDF}_k \tag{3.17}$$
  (im Weiteren kurz TF-IDF bezeichnet) repräsentiert werden.

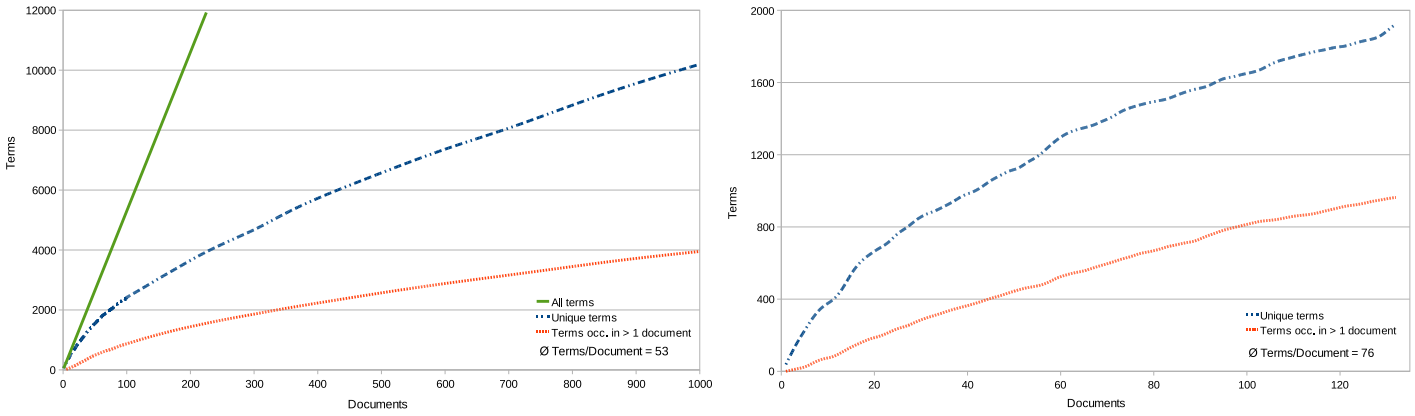
- Eine zusätzliche Reduzierung der Terme lässt sich durch die Anwendung von Zipf's Verteilungsgesetz erreichen (vgl. [BFS03, S. 24+80], [Jan07, S. 39]). Dieses Gesetz sagt aus, mit welcher Häufigkeit Wörter in Texten vorkommen. Demnach ist die Frequenz eines Terms umgekehrt proportional zur Rangfolge in der absteigend geordneten Wortfrequenzliste.
  Daraus kann man ableiten, dass Terme, die weit unten in der Wortfrequenzliste stehen auch nur in wenigen Dokumenten vorkommen und damit möglicherweise weniger Informationswert für die gesamte Dokumentensammlung haben, während Terme, die weit oben stehen, in praktisch allen Dokumenten vorkommen und daher kaum Informationswert für ein einzelnes Dokument haben.
  In der Praxis ist es gebräuchlich Terme auszusortieren, die nur in einem Dokument vorkommen.
  Eine andere Möglichkeit ist es, Terme auszusortieren, die in einer großen Anzahl Dokumenten (bspw. > 50% aller Dokumente) auftreten, da diese vermutlich keine Information zu Spezialisierung eines Dokuments beitragen und eher als „Grundrauschen" anzusehen sind (vgl. [BYRN+99, S. 146]).

  Zu bedenken ist noch, dass sich durch solche Maßnahmen der Ähnlichkeitswert zweier Dokumente ändert, was nicht immer gewünscht ist.[64]

[63] Im Buch ist fälschlicherweise Zähler und Nenner vertauscht. Siehe auch unter: http://ibook.ics.uci.edu/errata.html

[64] Beispielsweise, wenn Ähnlichkeitswerte verglichen werden sollen.

Eine Abschätzung zur Größenordnung der auszusortierenden Terme gibt Abb. 3.4, die einen typischen Textkorpus (bestehend aus Abstract und Titeln) einer Literaturdatenbank (*CiteSeer*) aufzeigt. Demnach ist es möglich, dass fast 50% der Terme ausgefiltert werden können, alleine wenn einzeln vorkommende Terme gefiltert werden.



(a) Zufällige Auswahl von Dokumenten

(b) Auswahl von Dokumenten zu einer bestimmten Anfrage

Abbildung 3.4: Verteilung und Wachstum von Termen in Relation zu der Anzahl der Dokumente.

Typischerweise wird die *Term-Document Matrix* (Formel 3.13) mit den TF-IDF Werten der entsprechenden Terme gefüllt (vgl. [BFS03, S. 84ff]). Aus diesen lässt sich nun über das oben vorgestellte Cosinus-Ähnlichkeitsmaß (Formel 3.14) eine Dokument-Dokument Ähnlichkeitsmatrix (Formel: 3.10, S. 30) aufstellen.

Verbesserung der Term-Document Matrix durch Latent Semantic Analyis

Bei der maschinellen Analyse von Texten gibt es u. a. das Problem der Synonyme[65] und der Polysemie[66] von Wörtern. Bei der Textanalyse ist es jedoch wünschenswert, dass Texte zum gleichen Thema, auch in der Analyse als „ähnlich“ erkannt werden.

Die bisher dargestellte textuelle Analyse ist zufriedenstellend, sofern in den zugehörigen Texten ausschließlich dieselben Wörter für eine Bedeutung verwendet werden. Im Zuge sprachlicher Varianz werden von Autoren jedoch gerne Synonyme verwendet und Forschergruppen verwenden möglicherweise unterschiedlichen Fachworte für dieselben Objekte. Hier würde die bisherige Analyse keine Übereinstimmung entdecken.

Um diesem Problem zu begegnen kann die LSA angewendet werden, die, wie der Name schon andeutet, verborgene, latente semantische Strukturen analysiert. Dabei wird ein statistisches Verfahren auf die Term-Document Matrix angewendet, um weitere, bislang nicht erkannte Zusammenhänge aufzudecken. Solche Zusammenhänge können unter anderem durch Synonyme und Polysemie, also durch

[65] *Synonyme* (aus dem Griechischen: gleichnamig, gleichbedeutend), sind verschiedene Begriffe, die in der Bedeutung die gleiche oder eine eng sinnverwandte Sache bezeichnen. (Vgl. [Ulr02])

[66] *Polysemie* (aus dem Griechischen: Viel-Zeichen, Mehrdeutig), bezeichnet den Umstand, dass mit einem Begriff mehrere Bedeutungen verbunden werden können. Ein Polysem hat also mehrere Deutungen, die allerdings ähnlichen Ursprungs sind. Der Fall, dass die Bedeutung des Wortes keine Ähnlichkeit aufweist, wird stattdessen als *Homonymie* bezeichnet. *Homonyme* sind gleichlautende Wörter mit unterschiedlicher und unabhängiger Bedeutung. Im Deutschen, wird diese Eigenschaft umgangssprachlich auch als „Teekesselchen“ bezeichnet. (Vgl. [Ulr02])

die Wortwahl, hervorgerufen werden. Dieses Verfahren, dass 1990 von [DDF+90] vorgestellt wurde[67], arbeitet mit der mathematischen Methode der Singulärwertzerlegung (*Singular Value Decomposition* (SVD)).

Die SVD stellt eine beliebige Matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ mit Rang r als das Produkt dreier spezieller Matrizen $\mathbf{U}$, Σ und $\mathbf{V}$ dar (Formel 3.18). Dabei ist insbesondere die Matrix Σ von Interesse, da in ihr nur die Hauptdiagonale $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_n)$ besetzt ist (die singulären Werte, ähnlich Eigenwerten), mit den Eigenschaften: $\sigma_1 \geq \ldots \geq \sigma_n$ und $\forall i \leq r \ : \ \sigma_i > 0$ und $\forall j > r \ : \ \sigma_j = 0$.
Die anderen beiden Matrizen sind unitäre Matrizen, für die die Beziehung $\mathbf{U}^T \cdot \mathbf{U} = \mathbf{I}$ und $\mathbf{V}^T \cdot \mathbf{V} = \mathbf{I}$ gilt. Sie können außerdem als die in Matrizen zusammengefassten Eigenvektoren aufgefasst werden.

Bei der LSA macht man sich also zunutze, dass eine beliebige Matrix, z. B. die oben vorgestellte Term-Document Matrix, in eine Anzahl von Werten zerlegt werden kann. Dabei sind die singulären Werte in Σ anhand ihrer Wichtigkeit auf die ursprüngliche Matrix geordnet. Das heisst, hohe Zahlenwerte in Σ haben auch einen entsprechend hohen Einfluß auf die ursprüngliche Matrix $\mathbf{A}$. Daher kann, wenn kleine Werte aus Σ vernachlässigt werden, eine Rang-reduzierte Annäherung $\mathbf{A}_k \approx \mathbf{A}$ erreicht werden, wobei $k \in \mathbb{N}$ und $k \leq r$ die Anzahl der singulären Werte in Σ angibt.
Das Produkt der Matrizen $\mathbf{U}$, Σ und $\mathbf{V}^T$

$$\mathbf{A} = \mathbf{U}\,\Sigma\,\mathbf{V}^T \tag{3.18}$$

wird approximiert durch $\mathbf{A}_k$ mit $k \in \{1, \ldots, \min(m,n)]$, wobei
$n = |D|$ (Anzahl der Dokumente) und $m = |V|$ (Anzahl der Terme, Vokabular):

$$\mathbf{A}_k = \mathbf{U}_k\,\Sigma_k\,\mathbf{V}_k^T. \tag{3.19}$$

Nach dem Theorem von Eckart-Young ist die Matrix $\mathbf{A}_k$ zugleich die beste Annäherung im Rang k an $\mathbf{A}$ in der Frobenius Norm (vgl. [BDO95, S. 3]): $\| \mathbf{A} - \mathbf{A}_k \|_F$ minimal.

In der Praxis funktioniert das Verfahren der LSA wie folgt:

1. Bilden der oben vorgestellten Term-Document Matrix A.
2. SVD-Zerlegung: $\mathbf{A} = \mathbf{U}\,\Sigma\,\mathbf{V}^T$.
3. Wählen einer Anzahl k von singulären Werten, mit der die Matrix $\mathbf{A}_k$ an $\mathbf{A}$ angenähert werden kann.
4. Errechnen von $\mathbf{A}_k$. Die Matrizen $\mathbf{U}$ und $\mathbf{V}^T$ können entsprechend angepasst werden (können nun Zeilen bzw. Spalten mit Nullen enthalten, oder sie werden in der Dimension angepasst). Zu beachten ist, dass die Matrix $\mathbf{A}_k$ nun nicht mehr dünn besetzt ist.
5. Berechnen der Dokument-Dokument Ähnlichkeitsmatrix mit der oben vorgestellten Skalarprodukt-Formel 3.14.

Eine schematische Darstellung der Matrizenoperationen ist in Abb. 3.5 veranschaulicht und ein einfaches, praktisches Beispiel ist in Anhang A zu finden.

[67] Das Verfahren wurde schon, abstrakt formuliert, 1988 in den USA als Patent angemeldet.

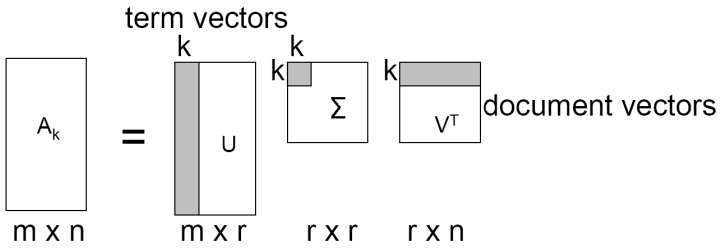


Abbildung 3.5: LSA Schematisch – Quelle: [Jan07, S. 41], angepasst.

Die Anwendung von LSA bringt zwei Schwierigkeiten mit sich: Zum einen ist es rechentechnisch sehr komplex und insbesondere bei großen Matrizen ist die Rechenzeit für die SVD nicht zu unterschätzen. Zum anderen muss der Parameter $k \in \mathbb{N}$ sinnvoll gewählt werden.
Eine gängige Methode die SVD zu berechnen basiert auf der QR-Zerlegung, für das in der mathematischen Bibliothek *Lapack*[68] hochoptimierte Verfahren bereitstehen. Die Komplexität des Verfahrens liegt aber immer noch im Bereich von $O(n^3)$. Neuere Algorithmen, wie in [Bra06] beschrieben, erreichen unter bestimmten Bedingungen (wie schwach besetzte Matrizen), eine Komplexität von $O(nmk)$, die es für Online-Verarbeitung besser verwendbar erscheinen lässt. Eigene Versuche haben ergeben, dass auch mit der in *Lapack* implementierten Version für Online-Verarbeitung akzeptable Rechenzeiten möglich sind, sofern die Dokumentenmenge im unteren zweistelligen Bereich bleibt (siehe auch Kapitel 6.3.5). Ein weiteres Problem, insbesondere bei sehr großen Dokumentenmengen ist die Größe der Matrix. Da nach der Anwendung von LSA die Matrix nicht mehr schwach besetzt ist, wird auch deutlich mehr Speicher benötigt (vgl. [BDO95]).

Leider gibt es in der Literatur nur unkonkrete Hinweise für die Wahl von k. In [DDF+90] findet sich die Angabe von $k = 100$ (für 1033 Dokumente und 5823 Termen) und in [Jan07] wird der fixe Wert von $k = 10$ vorgeschlagen. In anderen Publikationen finden sich Werte, die darüber und darunter liegen. Einige der Unterschiede sind auf die Art und Weise zurückzuführen, wie die Term-Gewichtungen ω_k in der *Term-Document Matrix* ermittelt werden, in [DDF+90] werden bspw. einfache, 'rohe' Termhäufigkeiten verwendet, während in anderen Publikationen TF-IDF Werte verwendet werden. Einen weiteren Unterschied gibt es durch die Vorauswahl der verwendeten Dokumente: Während in einigen Untersuchungen die verwendeten Dokumente durch Vorselektion einem bestimmten Oberthema zugeordnet waren, wurden in anderen Untersuchungen Dokumente ohne besondere Auswahl verwendet. In [DDF+90, S. 18] finden sich Hinweise darauf, dass sich LSA besonders dafür eignet schon vorausgewählte und zu einem Oberthema ausgesuchte Dokumente weiter zu strukturieren:

> [...] In such a circumstance our method does an excellent job of defining the isolated subdomains and separating them for retrieval.

Es existieren darüber hinaus eine Reihe von Heuristiken und Berechnungsverfahren, was die richtige Wahl von k angeht (vgl. [Bor05, S. 543ff]). Einschätzungen gehen von „allen singulären Werten größer eins“ bzw. „singuläre Werte größer als der Durchschnitt aller singulären Werte“, was der Ad-

[68] Traditionellerweise wird eine optimierte Bibliothek durch den CPU- oder Computerhersteller bereitgestellt. Eine freie Implementierung, die im Projekt auch verwendet wurde, findet sich unter http://www.netlib.org/atlas/index.html. Eine Java-Anbindung wurde durch die *Matrix Toolkits for Java* Bibliothek (siehe http://ressim.berlios.de/) realisiert.

aption des Kaiser-Guttman-Kriteriums entspricht, bis zu der Beurteilung von *Scree-Plots*[69], in denen die singulären Werte in einem Diagramm dargestellt werden (vgl. [Cat66] und [Jac93]). Dabei gibt der Abknickpunkt, an dem die Kurve vom steil Abfallenden zur nahezu Horizontalen übergeht, die Menge der singulären Werte an. Dies wird auch das „Ellenbogenkriterium“ genannt.
Allerdings ist dieses Verfahren sehr subjektiv vom „Auge des Betrachters“ abhängig und schwer automatisch auswertbar, insbesondere da viele Kurven keinen eindeutigen Knick haben oder es mehrere Knickpunkte gibt. Meist wird ein *Scree-Plot* daher nur manuell ausgewertet.
Das Kaiser-Guttman-Kriterium (auch Kaiser-Kriterium) ist ein Standardverfahren in der Statistik für die Faktorbestimmung. Dabei werden alle Faktoren ausgewählt, deren Beitrag zum ursprünglichen Wert mehr Varianz erklären als die ursprüngliche Variable selbst. Dies ist nur bei Eigenwerten größer als eins der Fall. Allerdings ist auch dieses Verfahren nicht sehr genau und arbeitet mit umstrittenen Grundannahmen, weshalb es bestenfalls als Anhaltspunkt oder Indiz dienen sollte ([Bor05, S. 544]).
Für eine maschinelle Auswertung können diese beiden Verfahren jedoch, eventuell in Kombination mit anderen externen Kriterien (z. B. eine manuelle Eingabe oder Grenzwerte von „interessanten Bereichen“) als erste Instanz hilfreich sein.

Die Bedeutung des Parameters k lässt sich darüber hinaus als eine Menge von Konzepten und / oder Themen innerhalb der betrachteten Menge von Dokumenten D verstehen. Damit würde sich durch die Wahl von k eine Obergrenze von Themen ergeben, die durch die Analyse der textuellen Ähnlichkeit ermittelt werden können. Dieser Fakt muß bei der Wahl der (optimalen) Clusterzahl (siehe unten) berücksichtigt werden.

[69] Der Name ist abgeleitet vom recht häufig anzutreffenden Aussehen der Kurve, wo die ersten Werte meist steil abfallen und die Kurve dann auf niedrigem Niveau (wie ’Geröll’ – Scree) stagniert.

3.1.2 Clusterbildung

Unter dem Begriff *Clusteranalyse*, auch *Ballungsanalyse* oder kurz *Clustern*, versteht man eine Klasse statistischer multivariater Verfahren, welche eine strukturbildende Gruppenbildung (Cluster) von Objekten ermöglichen, aufgrund von Eigenschaften, die als Ähnlichkeit oder Unähnlichkeit (Distanz) ausgedrückt werden können. Das Ziel der Clusterbildung ist, dass Objekte innerhalb eines Clusters sich ähnlicher sind, als zu Objekten in anderen Clustern. Die Clusteranalyse zählt zu der Gruppe der nicht überwachten Klassifikationsverfahren (unsupervised classification), was bedeutet, dass keine Trainingsdaten oder Lernphasen benötigt werden (Vgl. [TSK05, S. 490ff]).
Der Vorteil der Clusteranalyse ist, dass sie (im Vergleich z. B. zur LSA) mit relativ wenig numerischem Aufwand verbunden ist. Eine Clusteranalyse bspw. nach dem *K-Means* Verfahren (siehe unten) liegt im numerischen Aufwand bei $O(kn)$ (mit k zu bildenden Clustern und n Objekten). Dies erlaubt bspw. auch mehrere Szenarien „durchzurechnen“ und daraus die beste Lösung auszuwählen.[70]

Nach einer Übersicht über gängige Clustermethoden folgt ein kurzer Abschnitt über Qualitätsbewertung und die Wahl einer optimalen Clusteranzahl.

3.1.2.1 Clustermethoden

Grundsätzlich unterscheidet man die folgenden Eigenschaften von Clustermethoden (vgl. [TSK05, S. 492ff]:

- **Hierarchisch ./. Partitionierend**
  Partitionierende Clusterverfahren ordnen alle Objekte in „flache“ Cluster auf. Im Unterschied dazu können hierarchische Clusterverfahren Untergruppen bilden, bei denen ein oder mehrere Cluster zu übergeordneten Clustern zugehörig sind.
  Typische Vertreter für partitionierende Methoden sind: *K-Means*, *Repeated bisections* oder *Lingo*. Vertreter hierarchischer Methoden: *Agglomeratives hierarchisches Clustern (AHC)*.

- **Exklusiv ./. Überlappend**
  Exklusives Clustern bedeutet, dass jedes geclusterte Objekt genau einer Gruppe zugewiesen wird. Im Gegensatz dazu erlaubt überlappendes Clustern dass Objekte einer oder mehreren Gruppen zugewiesen werden. Eine Erweiterung zum überlappenden Clustern ist das sog. *Fuzzy Clustern*, bei dem Objekten eine Wahrscheinlichkeit zugewiesen wird, nach der sie zu einer Gruppe zugehörig sind.
  Ein exklusives Clustern wird meist von allen Clustermethoden unterstützt, das überlappende Clustern z. B. von *Soft K-Means* oder *Lingo*.

- **Vollständig ./. Inkomplett**
  Beim vollständigen Clustern wird jedes Objekt der Menge einer Gruppe zugewiesen. Da dies manchmal nicht gewollt ist, weil z. B. einige Objekte sehr verschieden sind (sog. „*Outlier*“), aber aus anderen Gründen in der Menge verbleiben, kann das inkomplette oder partielle Clustern nur einen Teil der Objektmenge gruppieren.
  Das inkomplette Clustern wird von manchen hierarchischen Clustermethoden unterstützt, meist ist es aber effektiver diese Elemente in einer eigenen Vorbehandlung auszusortieren.

[70] Dies soll nicht implizieren, dass das *K-Means* Verfahren immer die besten Ergebnisse liefert.

- **Clusterbasierend ./. Prototyp basierend**
  Cluster- oder datenbasierende Methoden arbeiten ausschließlich mit den Grunddaten (z. B. der Ähnlichkeitsmatrix) und ermitteln auf Basis dieser Daten die Clusterzuordnungen. Prototyp basierende Verfahren versuchen ein Cluster anhand eines typischen Vertreters dieses Clusters zu finden. z. B. wäre es bei der Clusterung von Dokumenten genauso möglich ein oder mehrere Stichworte zu geben und darauf aufbauend eine Gruppe von passenden Dokumenten zu finden.
  Allerdings kann auch ein clusterbasierender Algorithmus intern auf Prototypen (oder ähnlichen Konstrukten) aufbauen (z. B.*K-Means* durch die Wahl von Centroiden). Der ähnliche *K-Medoids* Algorithmus verwendet Datenpunkte (Medoide – Repräsentative Datenpunkte) zur Clusterzentrumsbildung).
  Typische Vertreter von Prototyp basierten Clustermethoden sind *Lingo* und *K-Medoids*.

Im folgenden ist nun eine Auswahl und Übersicht von oft verwendeten Clustermethoden für Dokumentensammlungn vorgestellt (vgl. [TSK05, S. 496ff] und [Hud07]):

- **K-Means**
  Das *K-Means* Verfahren ist mit Abstand das populärste Clusterverfahren, möglicherweise auch aufgrund seiner Einfachheit. Hier werden auf Basis von Zufallswerten Clusterzentren (Centroide) ausgewählt und die nächsten ähnlichen Objekte um diese Centroide gruppiert. *K-Means* gehört zu den klassischen partitionierenden Verfahren, da hier ausgehend von der Gesamtmenge Cluster gebildet werden.
  Allerdings geben die gebildeten Cluster nicht immer gut die Struktur der Daten wieder (sog. Cluster-Instabilität), was wesentlich an der zufälligen Wahl der Clusterzentren liegt, weshalb die meisten Implementierungen dieses Verfahren mehrfach durchführen und am Ende das beste Ergebnis auf Grundlage eines übergeordneten Bewertungsmaßes auswählen. (Vgl. [TSK05, S. 497])

- **Repeated Bisections (RB)**
  Dies ist eigentlich eine Variation von *K-Means*: Die Gesamtmenge wird bei diesem Verfahren so oft geteilt (in zwei Untercluster geteilt), bis die gewünschte Menge an Clustern erreicht ist.
  Da die Zerteilung in zwei Untercluster wie in *K-Means* durch einen Zufallswert beeinflusst wird, sind die Ergebnisse schwankend. Insgesamt erreicht dieses Verfahren ähnliche Ergebnisse wie *K-Means*.

- **Agglomerative Hierarchical Clustering (AHC)**
  Bei der Gruppe der agglomerativen Clustermethoden wird nicht die Gesamtmenge in Schritten verkleinert, sondern man beginnt mit einem Cluster pro Objekt der Menge. Diese 1er-Cluster werden dann nach Ähnlichkeitsbewertungen zusammengefügt. Dabei wird automatisch eine Hierarchie aufgebaut durch die miteinander verschmolzenen Gruppen (zusammengefügte Gruppen tragen die ursprünglichen Gruppen als Untergruppe mit). Diese Hierarchie kann als sog. Dendrogramm visualisiert werden und kann (subjektive) Aufschlüsse über die Qualität der Clusterung geben.
  AHC ist als gutes Clusterverfahren anerkannt, allerdings ist es bei kleinen Datenmengen nicht immer besser als ein partitionierendes Verfahren (vgl. [ZK02]).
  Bei der AHC unterscheidet man hauptsächlich die Art, wie Ähnlichkeiten zwischen Clustern (für das Zusammenfügen von Clustern) definiert werden: *Single Linkage* verwendet dabei den Ähnlichkeitswert der ähnlichsten Elemente, *Complete Linkage* den Ähnlichkeitswert der unähnlichsten Elemente (weitest entfernten) und *UPGMA* arbeitet mit dem Mittelwert der im Cluster enthaltenen Elemente (Centroid).

- **Suffix Tree Clustering** (STC)
  Suffix Tree Clustering arbeitet ausschließlich auf Texten und benutzt *Suffix Trees*, um häufige Wörter bzw. Teilwörter in der Dokumentenmenge besonders schnell zu finden.
  Dieser Algorithmus findet Cluster aufgrund der Verwendungshäufigkeit von Wörtern und generiert zu den Clustern aussagekräftige Beschreibungen. Allerdings ist das Verfahren kompliziert und sehr sensibel gegenüber veränderten Parametern und / oder Ausgangsdaten. Daher ist der Einsatz für automatisierte Prozesse möglicherweise schwierig zu implementieren. Ein großer Vorteil dieses Verfahrens ist allerdings die hohe Geschwindigkeit ($O(n)$) (Vgl. [ZE98]).

- **Lingo**
  Lingo ist ein rein textbasierter Ansatz zur Clusterung von Dokumenten. Dabei werden zunächst *Label* (beschreibende Wörter) aus den Texten generiert und anschließend Dokumente um diese *Label* herum gruppiert. Dabei können Dokumente zu mehreren Clustern zugeordnet werden. Die Erzeugung von aussagekräftigen *Labeln* geschieht mittels LSA.
  Die Qualität der Cluster ist sehr abhängig von den Texten, insbesondere von den im ersten Schritt ermittelten *Labeln*. *Lingo* ist, mit einigen Einschränkungen, fähig, eine Zahl von zu ermittelnden Clustern selbst zu generieren und erzeugt aussagekräftige Stichworte (die *Label*) für Cluster. (Vgl. [OSW04])

Um das Clustern zu vereinfachen, kann die Software `CLUTO` (Clustering Toolkit) verwendet werden, die eine Vielzahl von Clustermethoden bereitstellt. Die Clustermethoden *K-Means* (bzw. dem ähnliche Verfahren) und *Agglomeratives Hierarchisches Clustern* mit *Complete Linkage* sind für die Gruppierung von Dokumenten allgemein als 'gut' anerkannt (vgl. [ZK02]). Dabei wird für kleine Mengen von Dokumenten und eine kleine Anzahl von Dokumenten eher ein partitionierendes Verfahren empfohlen. In [SKK00] finden sich Hinweise, dass partitionierende Verfahren wie *K-Means* oder *RBR* in vielen Fällen bessere Ergebnisse als die der hierarchischen Verfahren liefern.
Die beiden anderen Verfahren *Suffix Tree Clustering* und *Lingo* eignen sich ausschließlich für das Clustern von Text. Da sie auch nicht über die oben vorgestellte Ähnlichkeitsmatrix, sondern direkt mit der *Document-Term Matrix* arbeiten, scheint eine Kombination mit bibliographischen Ähnlichkeitsmaßen vorerst unmöglich. Da diese Verfahren aber nachgewiesen gute Beschreibungen von Clustern liefern, lohnt sich möglicherweise eine weitere Untersuchung.

3.1.2.2 Clusteranzahl und Qualität

Die Ergebnisse einer Clusteranalyse können in ihrer Qualität mitunter stark schwanken, insbesondere wenn zufallswertgesteuerte Verfahren zum Einsatz kommen. Daher hat es sich bewährt das Ergebnis einer Clusteranalyse mit numerischen Verfahren zu bewerten (vgl. [SEW03]).
Klassischerweise werden im Information Retrieval für Klassifizierungsverfahren die Werte *Precision* und *Recall* zur Evaluation herangezogen (vgl. [TSK05, S. 549] und [SEW03, S. 217]):

- **Precision:** Gibt in einer binären Klassifizierung an, welchen Anteil die korrekt als positiv erkannten Ergebnisse in der Gesamtheit der als positiv erkannten Ergebnisse haben. In einem Dokumentenretrievalsystem kann man dies auch als Wahrscheinlichkeit ausdrücken, mit dem ein gefundenes Dokument relevant ist.
  Angepasst auf die Clusteranalyse: Die Wahrscheinlichkeit, mit der ein Cluster C_j, die Dokumente enthält, die in einer Referenzclusterung (etwa einer von Hand durchgeführten Klassifizierung) dem tatsächlichen Cluster C_i zugeordnet sind.

- **Recall:** Gibt in einer binären Klassifizierung an, welchen Anteil die korrekt als positiv erkannten Ergebnisse in der Gesamtheit der wirklich positiven Ergebnisse haben. In einem Dokumentenretrievalsystem ist dies die Wahrscheinlichkeit, mit der ein relevantes Dokument gefunden wird. Angepasst auf die Clusteranalyse: Die Wahrscheinlichkeit, mit der ein Cluster C_j alle Dokumente enthält, die in einer Referenzclusterung dem tatsächlichen Cluster C_i zugeordnet sind.

Bezogen auf die Clusteranalyse wird der Recall- und Precision-Wert immer auf das paarweise Verhältnis zweier Cluster C_j und C_i berechnet, wobei C_i das Ergebnis der „Referenzclusterung“ ist. Die Definitionen lassen es zu, dass die genannten Wahrscheinlichkeitswerte durch Mengenverhältnisse bestimmt werden können.
Formal sind Precision und Recall nun wie folgt definiert (vgl. [SEW03, S. 217] und [LA99, S. 18]):
Sei D die Menge von Dokumenten, $C = \{C_1, \ldots, C_k\}$ einer Clusterlösung von D und $C^* = \{C_1^*, \ldots, C_l^*\}$ die Lösung einer Referenzclusterung. Dann ist in Bezug auf den Cluster j und der Referenzklasse i

$$\text{precision}(i,j) = \frac{C_j \cap C_i^*}{|C_j|}, \tag{3.20}$$

und

$$\text{recall}(i,j) = \frac{C_j \cap C_i^*}{|C_i^*|}. \tag{3.21}$$

Aus dem harmonischen Mittel der Werte für *Precision* und *Recall* errechnet sich das sog. *F-Measure*, das als einzelner Wert Auskunft über die Qualität der Klassifizierung gibt:

$$\text{F}_{i,j} = \frac{2}{\frac{1}{\text{precision}(i,j)} + \frac{1}{\text{recall}(i,j)}} = \frac{2 \cdot (\text{precision}(i,j) \cdot \text{recall}(i,j))}{\text{precision}(i,j) + \text{recall}(i,j)}. \tag{3.22}$$

Für eine gesamte Clusterung kann ein gewichtetes arithmetisches Mittel über alle Cluster gebildet werden (vgl. [SEW03, S. 217]). Eine ausführlichere Darstellung von Precision und Recall findet sich außerdem in [BYRN+99, S. 75ff].

Das hier auftretende Problem mit *Precision* und *Recall* ist, dass keine Referenzclusterung vorhanden ist. Daher ist es zunächst unmöglich ein solches genaues Qualitätsmaß zu berechnen.
Für die Clusteranalyse wurden daher andere Maße entwickelt, die versprechen eine dem *F-Measure* ähnliche Aussage ohne die Existenz einer Referenzclusterung zu erlauben. Das bekannteste ist der *Dunn*-Index, bei dem die minimale Interclusterdistanz ins Verhältnis zur maximalen einzelnen Intraclusterdistanz gesetzt wird. Sei $C = \{C | C \subseteq D\} = \{C_1, \ldots, C_k\}$ eine Clusterung der Dokumente D, dann berechnet sich der Dunn-Index ([SEW03, S. 217]):

$$I(C) = \frac{\min_{i \neq j}\{\delta(C_i, C_j)\}}{\max_{1 \leq l \leq k}\{\Delta(C_l)\}} \tag{3.23}$$

mit $\delta(C_i, C_j) = \min_{x \in C_i, y \in C_j} d(x,y)$ und $\Delta(C_i) = \max_{x,y \in C_i} d(x,y)$. $d(x,y)$ sei eine Distanzfunktion, die den Abstand zwischen zwei Objekten angibt.
Zu diesem Maß sind außerdem verschiedene Varianten bekannt.
Allerdings hat sich gezeigt, dass Indizes dieser Art nur bedingt geeignet sind, eine gute Clusterung zu identifizieren (vgl. [SEW03], [KLB05]). Im Allgemeinen kann nur eine schlechte Clusteranalyse als eine solche erkannt werden, darüber hinaus ist die Aussagekraft dieser Merkmale ungenügend. Bessere Ergebnisse sind mit anderen Verfahren möglich. In [SEW03, S. 218] werden graphenbasierten Verfahren wie z. B.*Measure of expected density* vorgeschlagen. [TSK05, S. 536ff] schlägt ein ähnliches

Maß namens *Cohesion and Separation* vor. Einige Varianten von graphenbasierten Verfahren und des Dunn-Indexes sind in der Software `CLUTO` implementiert um eine Bewertung von Clusterungen bei zufallswertgesteuerten Verfahren zu ermöglichen.

Für die Qualitätsbeurteilung und insbesondere Einschätzung einer optimalen Anzahl von Clustern wird oft auf ein anderes Maß zurückgegriffen: Die Auswertung von *Silhouette Kurven*, die dem *Dunn*-Index algorithmisch ähnlich sind (vgl. [TSK05, S. 541]).
Dabei wird für jedes Objekt die Ähnlichkeit mit den Elementen im eigenen Cluster (Klasse) berechnet und mit der Ähnlichkeit zu den Elementen anderer Cluster (bzw. deren Centroide) in Relation gesetzt. Die Mittelwertsbildung über alle Elemente ergibt eine numerische Kennzahl für die Clusterqualität $\bar{s}(k)$ für eine Clusteranalyse mit k Clustern.
Sei $d_i \in C_i$ ein Objekt im Cluster $C_i \in \{C_1, \ldots, C_p\}$.

1. Berechnung der mittleren Distanz a_i für ein Objekt d_i zu Elementen der eigenen Klasse C_i:

$$a_i = \frac{1}{|C_i| - 1} \sum_{x \in C_i, x \neq i} \text{dist}(x, i) \tag{3.24}$$

2. Berechnung der minimalen mittleren Distanz b_i zu den Objekten der nächstgelegenen Klasse C_j:

$$\text{dist}(i, C_j) = \frac{1}{|C_j|} \sum_{x \in C_j} \text{dist}(i, x) \tag{3.25}$$

$$b_i = \min_{i \neq j}(i, C_j) \tag{3.26}$$

3. Berechnen des *Silhouette*-Werts für ein Objekt:

$$s_i = \frac{b_i - a_i}{\max(a_i, b_i)} \tag{3.27}$$

Der *Silhouette*-Wert $\bar{s}(k)$ kann Werte zwischen -1 und 1 annehmen, wobei höhere Werte auf eine besser strukturierte Clusterung hinweisen.
In der Literatur finden sich verschiedene Angaben über „gute“ Werte von $\bar{s}(k)$. Im Allgemeinen wird ein Wert von über $0,5$ als gute Clusterung angesehen, in der die gefundene Datengruppierung eine herausgestellte Struktur offenbart (vgl. [Han08, S. 395 Tab. 13.22]).
Eine Bestimmung der optimalen Clusterzahl unter Berücksichtigung einer minimalen und maximalen Clusterzahl lässt sich nun wie folgt vornehmen:

1. Setze $k =$ minimale Clusterzahl
2. Clusteranalyse für k Cluster
3. Bestimme $\bar{s}(k)$
4. Falls $k <$ maximale Clusterzahl setze $k + 1$ und gehe zurück zu `2`
5. Der maximale Wert $\max(\bar{s}(k))$ gibt die beste Clusteranzahl k an. Der maximale Wert sollte dabei als lokales Maximum innerhalb eines sehr begrenzten Bereiches gesucht werden, da global der maximale Wert immer erreicht ist, wenn alle Objekte in einem eigenen Cluster sitzen.

Es ist außerdem zu beachten, dass eine Clusterzahl größer als die Anzahl der LSA-Faktoren nicht sinnvoll ist, wenn man zugrundelegt, dass die Wahl der LSA-Faktoren schon eine Anzahl unabhängiger Themengebiete eingrenzt.
Die Analyse des *Silhouette* Wertes ist oft nicht eindeutig genug, insbesondere für den Einsatz als unüberwachtes Verfahren, um in jedem Fall eine optimale Anzahl von Clustern zu ermitteln. Hier müssen andere Verfahren gefunden werden oder eine Benutzereingabe muss weiterhelfen (analog zu der Ermittlung von LSA-Faktoren oben (S. 36)).
Mehr oder weniger ungeklärt ist zudem wie mit sog. *Outliers* umzugehen ist, also Objekten in der zu clusternden Menge, die nur geringe Ähnlichkeit allen anderen Dokumenten haben. Im optimalen Fall werden solche Objekte einem eigenen ein-elementigen Cluster zugeordnet werden. Da aber die Clusterbildung einiger Methoden mit zufälligen Clusterzentren arbeitet und dabei versucht wird immer alle Elemente zu clustern, können solche Elemente auch zur Verschlechterung von Ergebnissen führen. Sehr oft wird daher vor der Clusteranalyse eine Suche nach *Outliers* gestartet, um sie aus der zu clusternden Menge auszuschließen. Diesen Elementen kann dann bspw. ein „Restcluster“ zugewiesen werden um im Ergebnis den „Nichtzusammenhang“ dieser Elemente mit den anderen Objekten zu symbolisieren.

3.1.3 Kombination von Bibliographischer- und Textähnlichkeit

Mit den bisher vorgeschlagenen Verfahren konnten Ähnlichkeitsmatrizen auf bibliographischer- (Abschnitt 3.1.1.1) und auf Textähnlichkeitsbasis (Abschnitt 3.1.1.2) errechnet werden. Jedes dieser beiden Verfahren hat Schwächen und Stärken. Daher liegt der Wunsch nahe, beide Verfahren zu einem gemeinsamen Ergebnis zu kombinieren, um mehr nutzbare Informationen zu haben.
Dazu gibt es mehrere Möglichkeiten:

- **Serielle Kombination**
  Diese Methode wurde u. a. von [BMR91] und in [Jan07, S. 118] vorgeschlagen um die Ergebnisse eines bibliographischen Verfahrens zu verbessern. Dabei wird zunächst eine Clusteranalyse von bibliographischen Daten durchgeführt und diese dann durch einer Analyse von textuellen Daten optimiert.

  Die Daten der zweiten Analyse (bspw. der Textanalyse) könne dabei im einfachsten Fall als zusätzliche Lösung bzw. Visualisierung angezeigt werden. Bei [BMR91] werden „Verbesserungen“ an einer ersten Clusteranalyse von Kozitationsdaten vorgenommen, indem Dokumente oder ganze Cluster, die bspw. nach Textanalyse zu einem anderen Cluster besser passen würden, verschoben bzw. verschmolzen werden. Dazu werden die Wörter der im Cluster enthaltenen Dokumente mit denen anderer Cluster verglichen und es wird ein Textprofil für jeden Cluster erstellt. Mit diesem Textprofil kann eine Cluster – Cluster Wortähnlichkeit analysiert werden, aufgrund derer letztendlich Dokumente oder ganze Cluster miteinander kombiniert werden.
  Dieses Verfahren kann nur funktionieren, wenn die Kozitationsanalyse eine genügend stabile Basis für eine Analyse erbracht hat. Dieses ist dann gegeben, wenn es mind. ein identifizierbares Kerndokument pro Cluster gibt, dessen Textanteile sich kaum mit anderen Clustern überschneiden. In der Analyse und Evaluation von [BMR91] wurden dazu die Textdaten der Dokumente genau ausgewählt und stark gefiltert.
  Ein Spezialfall dieses Verfahrens ist gegeben, wenn nur *Outliers* oder nicht zu Clustern zugeordnete Dokumente[71] aus der ersten Clusteranalyse zu bestehenden, passenden Clustern verteilt werden.

- **Integrative Kombination**
  Diese Form der Kombination kombiniert die beiden Ähnlichkeitsmatrizen aus bibliographischer und Textähnlichkeit vor dem Clustern. Da beide Matrizen formal Dokument–Dokument Ähnlichkeitsmatrizen mit jeweils gleicher Dimension darstellen, liegt dieser Schritt nahe (und ist mathematisch möglich).
  Problematisch ist die ungleiche Verteilung von Werten in beiden Matrizen. Z. B. könnte die bibliographische Analyse zwei Dokumente X und Y als sehr ähnlich klassifizieren, was möglicherweise aber konträr zu Ergebnissen der textuellen Analyse ist. Die Entscheidung welcher Wert nun als „korrekter“ zu interpretieren ist, kann nicht universell getroffen werden (vgl. [Jan07, S. 123])
  Trotzdem erscheint diese Art der Kombination vielversprechend, da die Kombination auf Basis der Ähnlichkeitsmatrizen mehr Informationen vereinen kann und genau ein Ergebnis liefert, was dem Ziel der kompakten Darstellung für einen potentiellen Benutzer eher entspricht. Hier eine genauere Übersicht über Möglichkeiten der Matrizenkombination:

[71] Die kann z. B. passieren, wenn keine Zitationsinformationen über das Dokument vorliegen, was beim CiteSeer Datensatz gelegentlich beobachtet wurde.

– **Linearkombination der Matrizen**
Dies ist die einfachste und zugleich offensichtlichste Variante der Kombination (vgl. [Jan07]). Dabei werden die beiden Matrizen einfach mit einem Faktor α gewichtet und zusammenaddiert:

$$\bar{S} = \alpha \cdot S_1 + (1-\alpha) \cdot S_2. \tag{3.28}$$

Anders interpretiert entsteht hier ein gewichteter Mittelwert der Werte aus den beiden Ähnlichkeitsmatrizen. Mit $s_{i,j} \in S$:

$$\bar{s}_{i,j} = \alpha s_{1_{i,j}} + (1-\alpha) s_{2_{i,j}} \tag{3.29}$$

Allerdings kann diese Variante nicht auf verschiedene Verteilungen von Ähnlichkeiten, beispielsweise aus textbasierter und bibliographischer Analyse eingehen, so dass hier Informationen verdeckt oder auch ungerechtfertigt verstärkt werden können.
Eine nicht einfach zu entscheidende Frage ist die Frage nach einem optimalen Wert für α. Ein simpler Ansatz ist hier sicher die Wahl von $\alpha = 0.5$, wodurch ein arithmetisches Mittel der Matrixwerte gebildet wird. Allerdings werden dadurch auch Spezialisierungen von einem Ähnlichkeitsmaß gemittelt und gehen so möglicherweise unter. Eine andere Möglichkeit besteht darin eine Matrix stärker zu gewichten, birgt aber die Gefahr, dass damit auch z. B. „fehlerhafte“ Werte mehr an Bedeutung gewinnen. Eine vielversprechender Ansatz stellt daher der Ansatz dar, die Ergebnisse der bibliographischen Verfahren stärker zu gewichten als die der textuellen Analyse, wenn man, wie in [BMR91] davon ausgeht, dass der Zusammenhang zwischen zwei Dokumenten auf Basis von Zitationen aussagekräftiger ist, als auf Basis der Wortwahl im Text.
Außer dem hier vorgestellten arithmetisches (gewichteten) Mittel können auch andere Mittelwertverfahren angewendet werden, wie z. B. der geometrische Mittelwert:

$$\bar{s}_{i,j} = \left(\prod_{k=1}^{n} s_{k_{i,j}}^{w_k} \right)^{1/\sum_{k=1}^{n} w_k} \tag{3.30}$$

Was sich für den hier betrachteten Fall vereinfacht zu:

$$\bar{s}_{i,j} = e^{\left(\alpha \cdot \ln(s_{1_{i,j}} + (1-\alpha) s_{2_{i,j}})\right)} \tag{3.31}$$

– **Fisher's inverse chi-square method**
Diese Variante, vorgestellt in [Jan07, S. 124], versucht den Umstand der verschiedenen Verteilungen in den Ähnlichkeitsmatrizen gerecht zu werden mit einem komplexen statistischem Kombinationsverfahren.
Dabei werden die beiden Matrizen mit einer mit zufälligen Verteilungen gefüllten Matrix überlagert. Danach werden mittels einer statistischen Verteilungsfunktion die Werte aus beiden Matrizen in eine neue dritte Matrix übernommen.
Die Ergebnisse dieses Verfahrens haben den Anspruch in vielen Fällen besser zu sein, als die mit der linearen Kombination erzeugten Matrizen.

- **Ensemble von Clusterlösungen**
Diese Kombinationsvariante vereinigt verschiedene Clusteranalysen zu einer Gesamtlösung. Hier werden bspw. das Ergebnis der Clusteranalyse von bibliographischen Daten und Textdaten zunächst separat behandelt, um dann mittels einer „Entscheiderfunktion“ zu einer Gesamtlösung kombiniert zu werden. Dieses Verfahren wird von [SG03] vorgeschlagen und liefert vielversprechende Ergebnisse, ist allerdings konzeptionell von der Wahl einer sinnvollen „Entscheiderfunktion“ abhängig, welche für die Problemstellung in dieser Arbeit nicht vorliegt.

Einen Vergleich verschiedener Formen integrativer Kombinationen von Bibliographischer Kopplung und Textanalyse (teilweise mit Anwendung von LSA) zeigt [Jan07, S. 130–135]. Die Ergebnisse lassen den Schluss zu, dass die Herangehensweise einer Einzelanalyse von Bibliographischer Kopplung oder Textanalyse zu bevorzugen ist.
Die Ergebnisse der Kombination mittels „Fisher's inverse chi-square method" sind in der Evaluation zwar in einigen Fällen besser, bzw. zeigen einen höheren strukturellen Zusammenhang, als die der linearen Kombination, allerdings sind die Vorteile marginal und liegen z. T. im Bereich der statistischen Streuung.
Daher wird das Verfahren der linearen Kombination hier als gleichwertig zu der „Fisher's inverse chi-square" Methode betrachtet. Dabei hat die lineare Kombination zusätzlich den Vorteil der einfachen Implementierung.

3.1.4 Clusterlabel Generierung

Um die Navigation und Übersicht über erzeugte Cluster zu erleichtern, bietet es sich an, zu diesen Clustern treffende Bezeichnungen zu finden, die einem Anwender einen Eindruck von den enthaltenen Dokumenten liefern.
Dazu werden Algorithmen benötigt, die einem Dokument bzw. einer Gruppe von Dokumenten Bezeichner zuordnen. Grundsätzlich eignen sich für diesen Zweck eine Vielzahl von Verfahren. Hier werdem die für den vorgesehenen Anwendungszweck im vorliegenden Projekt wichtigsten kurz aufgeführt:

- **Spezielle Clustermethoden**
  Einige Clustermethoden, wie bspw. *Lingo* oder *Suffix Tree Clustering* (siehe auch Abschnitt 3.1.2) generieren automatisch passende Bezeichner für jeden erstellten Cluster.
  Allerdings arbeiten diese Clustermethoden ausschließlich auf den durch Textanalyse erstellten Daten. Ähnlichkeitswerte der bibliographischen Methoden sind nicht verwendbar.

- **Kategorisierende Verfahren**
  Diese können angewandt werden, wenn es um die Zuordnung eines Dokuments bspw. zu einer feststehenden Liste von Themen oder Einordnungen geht. Dazu stehen eine Reihe von klassischen IR Verfahren (wie Entscheidungsbäume oder Naïve Bayes Klassifizierer) zur Verfügung (vgl. [BFS03, S. 93ff]).
  Für den vorgesehenen Einsatzzweck existiert allerdings keine vorgefertigte Liste mit Themen, die angewendet werden könnte. Ein anderes Problem liegt darin, dass die meisten dieser Verfahren überwachte bzw. im Vorfeld mit Beispieldaten trainierte Methoden sind.

  Nichtsdestotrotz ist der Einsatz kategorisierender Verfahren sehr gebräuchlich, wenn bspw. ein Dokument einer Fachrichtung oder einem Forschungsgebiet zugeordnet werden soll. Oft sind zu diesem Zweck in Veröffentlichungen bestimmte Schlüsselworte (bspw. das *ACM Computing Classification System*[72], oder mit dem *Dewey Decimal System*[73] (DDC)) vorgesehen, allerdings ist die Benutzung entsprechender Schlüsselworte für Autoren von wissenschaftlicher Literatur sehr oft nur optional.

[72] Die aktuelle Liste findet sich unter: http://www.acm.org/class/, Stand 2008-02-24

[73] Dies ist das verbreiteteste Klassifikationssystem für Bibliotheksbestände. Die aktuelle Liste wird von der Organisation *Online Computer Library Center* (OCLC) verwaltet und die Benutzung ist lizenzpflichtig. Allerdings ist das Schema bei weitem nicht so feingranular wie das *ACM Computing Classification System*.
Informationen zur aktuellen Version der Klassifikationsliste gibt es unter: http://www.oclc.org/dewey/, Stand 2008-02-24

- **Wortprofile**
  Die einfachste Möglichkeit Stichwörter für jeden Cluster zu bekommen, ist die Adaption des in Abschnitt 3.1.1.2 vorgestellten *Bag-Of-Words* Formats mit TF-IDF Wortgewichten. Dabei wird ein mit TF-IDF gewichteter Termvektor für jeden Dokumentencluster unter Berücksichtigung der Worthäufigkeiten in der gesamten Dokumentenmenge (für den IDF Anteil) aufgestellt. Die Einträge mit den höchsten Wortgewichten sind aussagekräftige Stichworte für die Dokumente im Cluster (vgl. [BMR91]).
  Eine das Ergebnis verbessernde Variation kann durch das Ausfiltern von Wörtern geschehen, die nur in einem Dokument vorkommen. Dies basiert auf der Annahme, dass solche Wörter zwar einen hohen diskriminativen Charakter für ein einzelnes Dokument haben (und dadurch durch TF-IDF eine hohe Gewichtung bekommen), aber für eine Beschreibung von mehreren Dokumenten eines Clusters möglicherweise ungünstig sind.

  Aussagekräftige Beschreibungen von Dokumenten bzw. Clustern können durch die Analyse von N-gram(en) (Häufigkeitsanalyse von Wortkombinationen im Text, dabei werden meist die Kombinationen von zwei oder drei Wörtern untersucht; sog. Bi- und Trigramme) gewonnen werden. Diese Methode wird bspw. für einzelne Dokumente in der *Rexa* Literaturdatenbank (siehe S. 16) durchgeführt und angezeigt. Dieses Verfahren ist deutlich aufwendiger als die Analyse von Einzelwörtern, da es LSA ähnliche Methoden (*Latent Dirichlet Analysis*) verwendet (vgl. [WMW07]).

- **Wortprofil eines Kerndokuments**
  In [BMR91] wird vorgeschlagen, die Bezeichner eines Clusters durch die Suche nach einem Kerndokument zu vereinfachen. Ein solches Kerndokument hat im Idealfall keine oder nur sehr wenige Wörter mit Dokumenten aus anderen Clustern gemein und kann daher als „typischer Vertreter“ der Dokumentengruppe gelten. Aus diesem Kerndokument können im einfachsten Fall die Terme mit der höchsten Termfrequenz als beschreibende Stichwörter für das Cluster verwendet werden.
  Ein Kerndokument kann auch durch andere Verfahren identifiziert werden. Beispielsweise können weitere Analysen des bibliographischen Zusammenhangs innerhalb eines Clusters durchgeführt werden (z. B. mit *HITS* oder *PageRank*, siehe Seite 26), um ein Dokument mit der höchsten Bewertung zu finden. Ein weiterer Anhaltspunkt kann der Analyse des Medoids (Dokument, das dem Clustermittelpunkt (Centroid) am nächsten ist, bzw. dieses repräsentiert) eines Clusters sein (vgl. [Jan07, S. 160]).

Keines dieser Verfahren kann in allen Fällen perfekte Beschreibungen (im Sinne von: für alle Nutzergruppen) für Dokumente oder Gruppen von Dokumenten liefern. Evaluationen (bspw. in [WMW07]) legen den Schluß nahe, dass die Ergebnisse der N-gram Analyse den Einzelwörtern bzw. Stichwörtern vorzuziehen ist. Eine einfacher zu implementierende Lösung wie etwa Wortprofile, können aber dessen ungeachtet trotzdem zu aussagekräftigen Ergebnissen führen.

3.2 Untersuchung der Gebrauchstauglichkeit

Dieser Abschnitt widmet sich den Möglichkeiten der Evaluation der Verwendbarkeit der entwickelten Software, um eine entsprechende Nutzerstudie (siehe Kapitel 7) durchführen können.
Ziel ist es, eine Aussage treffen zu können, ob eine bestimmte Funktion einer Software, etwa die oben besprochene Funktion zur Gruppierung von Dokumenten, hilfreich für eine Person der Zielgruppe sein kann. Des Weiteren soll eine Beurteilung der Art der graphischen Aufarbeitung und Bedienung des Systems möglich sein.

3.2.1 Usability oder die Messbarmachung der Verwendbarkeit – Begriffserklärung

Da Gestaltungsrichtlinien und Beschreibungen für Webseiten meist nur den Schluss auf ein allgemeines Anwenderverhalten zulassen, können programmspezifische, webseitenspezifische und nutzerspezifische Ergebnisse nur mit Hilfe von Nutzerstudien gewonnen werden (vgl. [NL06, S. 10ff]).

Welche Aspekte in die Messung der Gebrauchstauglichkeit bzw. Usability einfließen sollten, lässt sich bspw. aus den Usability-Kriterien der DIN EN ISO 9241-11 ableiten. Dort sind die folgenden Leitlinien definiert:

- Effektivität zur Lösung einer Aufgabe,
- Effizienz der Handhabung des Systems,
- Zufriedenheit der Nutzer einer Software.

Die Definition der Gebrauchstauglichkeit schließt hier die Bestimmung der „Nützlichkeit“, also die Frage, ob das Produkt den inhaltlichen Zweck erfüllt, mit ein. Eine Untersuchung der Usability kann daher sowohl die Frage nach der Bedienbarkeit bzw. Benutzbarkeit, im Sinne der Oberflächengestaltung und Programmlogik, als auch die Frage nach der Nützlichkeit behandeln. Im Weiteren wird nur der Begriff „Usability“ verwendet.

3.2.2 Untersuchungsmethoden

Usability

Zur Untersuchung der Usability können verschiedene Mittel und Methoden eingesetzt werden. Es haben sich für Untersuchungen dieser Art die folgenden Methoden etabliert (vgl. [GHD02] und [Heg03]):

- **Verhaltensbasierte Techniken**
  Techniken, die auf der Auswertung des Verhaltens einer Versuchsperson (VP) beruhen. Dazu gehört beispielsweise die Beobachtung des Probanden und / oder Protokollierung von verbalen Äußerungen. Ebenso kann die Analyse von Logdateien eines Computerprogramms herangezogen werden.
  Es gelten drei Hauptvarianten als typische Vertreter dieser Technik:
  - **Beobachtungstechniken**
    Die Versuchsperson wird von einem trainierten Beobachter begleitet. Der Beobachter nimmt alle relevanten Daten (Bewegungen, Gesicht, Äußerungen, Körperhaltung etc.) auf. Um die Versuchsperson nicht zu beeinflussen und um alle relevanten Daten zuverlässig zu erfassen, werden im Allgemeinen Videotechniken und Logdatei-Analysen verwendet.

  - **Laut-Mitdenken**
    Die Versuchsperson wird gebeten alle Gedankengänge und Empfindungen während des Versuchs sprachlich zu artikulieren. Ein Beobachter ist dafür verantwortlich alle Äußerungen direkt oder indirekt über ein Aufnahmegerät aufzuzeichnen.
    Eine denkbare Variation ist, dass Versuchspersonen ihre Gedanken und Handlungen vor, während und / oder nach einer Aktion laut beschreiben.
    Ein Nachteil dieser Methode ist, dass sie von Versuchspersonen oft als ungewohnt und verwirrend empfunden wird. Unter Umständen kann sich dadurch ein Gefühl des „Erwartung erfüllen“ und des „richtige Antwort“ geben einstellen, wodurch die Evaluation verfälscht wird (vgl. [Heg03, S. 51].
  - **Video Gegenüberstellung**
    Bei dieser Technik wird der gesamte Versuch auf Video aufgezeichnet, anschließend wird die Versuchsperson mit ausgewählten Stellen des Versuchs konfrontiert. Aus den verbalen Äußerungen und Erklärungen kann ein Protokoll erstellt und insbesondere Fehlersituationen können aufgeklärt werden.

- **Meinungsbasierte Techniken**
  Meinungsbasierte Techniken können entweder in Form eines mündlichen Interviews oder eines schriftlichen Fragebogens organisiert sein. In allen Fällen ist das Ziel die Erhebung eines subjektives Meinungsbildes der Versuchspersonen über das evaluierte Programmsystem. Im Allgemeinen geben schriftliche Tests ein leichter auswertbares Bild als mündliche Befragungen, allerdings ist der Zeitaufwand für die Vorbereitung etwas höher.

  Es haben sich für die Evaluation von Computersystemen verschiedene Fragebogensysteme etabliert (vgl. [GHD02]): Dazu gehören unter anderem das „Questionnaire for User Interface Satisfaction“ (QUIS), „The Software Usability Measurement Inventory“ (SUMI), Isometrics, „Software Usability Scale“, „Nielsen's Heuristic Evaluation“ (NHE) (vgl. [Nie03]), oder „5 Dimensions of Usability“ (5Es) (vgl. [Que03]).
  Einige dieser Fragebogensysteme (SUMI, QUIS) sind Teil eines kommerziellen Softwaresystems. Nähere Beschreibungen der wichtigsten Kriterien folgen in Abschnitt 3.2.3.

Sowohl verhaltens- wie meinungsbasierte Methoden setzen die Existenz eines Programmprototypen voraus und mindestens eine Versuchsperson. Der Zeitaufwand für die Gestaltung des Versuchs und der Auswertung verhaltensbasierter Techniken ist meist deutlich höher als bei meinungsbasierten Techniken. Allerdings muss dort besondere Sorgfalt auf die Konzeptionierung und Gestaltung der Fragebögen gelegt werden um klare Ergebnisse zu erzielen.

Funktionsevaluation

Die oben genannten Methoden haben ihren Schwerpunkt in der Evaluation der Benutzerinteraktion sowie der grundsätzlichen Funktionalität der graphischen Aufbereitung. Um eine Evaluation einer einzelnen Funktion, bzw. dessen Anteil am Gesamtergebnis zu erreichen, bietet sich das Hilfsmittel des (psychologischen) *Experiment*s an (vgl. [Sta05] und [Heg03, S. 71ff]).
Bei einem *Experiment* untersucht man die Wirkung einer oder mehrere unabhängiger Variablen (auch unabhängige Faktoren) auf Versuchspersonen. Als unabhängige Variablen werden alle Faktoren bezeichnet, die in einem *Experiment* vom Versuchsleiter aktiv verändert werden können. Dazu gehören etwa Funktionen in einem Programm, die nur vom Versuchsleiter zu- und abgeschaltet werden können.

Daneben gibt es die abhängigen Variablen. Diese symbolisieren das Ergebnis, welches von den unabhängigen Variablen verändert wird. In einem Softwareprodukt kann dies beispielsweise ein Rechenergebnis oder eine vom Programms produzierte Ausgabe sein. Im Allgemeinen sind die Analyse der abhängigen Variablen das Ziel der Evaluation.
Störvariablen haben einen zusätzlichen Effekt auf die abhängigen Variablen, allerdings ist dieser Effekt im Versuchsaufbau unerwünscht und soll nicht gemessen werden. Daher versucht man den Einfluss solcher Variablen zu kontrollieren (d. h. auszuschalten) oder zu minimieren. Zu der Gruppe der Störvariablen gehören u. a. auch die Versuchspersonen selbst, da nicht sichergestellt werden kann, dass diese bspw. einen vergleichbaren Kenntnisstand mitbringen. Des Weiteren kann auch der Versuchsleiter zu Störvariablen zählen, wenn er in die Entwicklung mit einbezogen ist und eine subjektive Erwartungshaltung an das Ergebnis der Evaluation besteht (vgl. [Heg03, S. 76]).

Es gibt verschiedene Möglichkeiten Störfaktoren auszuschließen und / oder zu minimieren. In der Medizin hat sich bspw. die Methode des „Doppelblindversuchs“ bewährt, in der selbst Versuchsleiter keine Kenntnis vom Status der unabhängigen Variablen (in der Medizin bspw. von den Wirkstoffen einer Medizin) im Experiment haben (vgl. [Sta05]). Um Störeffekte durch die Auswahl der Versuchspersonen zu minimieren, wird oft auf eine zufällige Auswahl, eine größere Anzahl und Vortests gesetzt.
Wenn in einem *Experiment* keine effektive Kontrolle der Störfaktoren möglich ist, spricht man von einem *Quasi-Experiment*.

Um den Einfluss von unabhängigen Variablen auf eine oder mehrere abhängige Variablen zu analysieren, wird meist ein systematisches Manipulieren der unabhängigen Variablen vorgeschlagen (vgl. [Heg03, S. 71]). Um dabei sicherzugehen, dass nur die gewählten unabhängigen Variablen Einfluß auf die abhängigen Variablen haben, werden oft zwei verschiedene Versuchspersonengruppen eingesetzt: Experimental- und Kontrollgruppe.
Obwohl beide Gruppen formal aus Personen derselben Zielgruppe zusammengesetzt sind, werden sie mit leicht unterschiedlichen Experimenten betraut. Personen aus der Kontrollgruppe werden dabei mit einem Experiment betraut, das die unabhängigen Variablen auf einem definierten Zustand belässt, während bei Personen der Experimentalgruppe die unabhängigen Variablen variiert werden.

Es ist, gerade bei Softwareprojekten, nicht leicht die abhängigen von den unabhängigen Variablen zu trennen, da möglicherweise Teile einer Software von Versuchspersonen anders verwendet werden, wenn bestimmte Funktionalitäten (entspricht den unabhängigen Variablen) nicht vorhanden sind. Dieser Gefahr kann durch sorgfältige Planung des Experiments und einer akkuraten Durchführung entgegengewirkt werden. Für den Zweck der Softwareevaluation bietet es sich an mehrere Versionen einer Software zu erstellen, die sich um die zu evaluierende Funktionalität unterscheiden. Um interpretierbare Ergebnisse zu erlangen, sollte die Software dabei in ihrer Kernfunktionalität erhalten bleiben.

Wichtig für die Auswertung eines Versuchs ist die genaue Bestimmung der abhängigen Variablen, also bei einem Softwareprojekt die genaue Festlegung auf möglichst einzeln zu bestimmende Faktoren, die ausgewertet werden können. Dabei unterscheidet man zwischen einer quantitativen Auswertung und einer qualitativen Auswertung. Quantitative Auswertungen sind nur möglich, wenn die oben beschriebenen abhängigen Variablen eine numerische Auswertung erlauben, also mehr als zwei Ausprägungen haben. Ansonsten spricht man von einer qualitativen Auswertung. Dies ist im Allgemeinen auch bei einer Beurteilung der abhängigen Faktoren durch Experten gegeben, wenn die Auswertung bspw. auf Vergleichen beruht (besser / schlechter).

3.2.3 Untersuchungskriterien für meinungsbasierte Techniken

Im Folgenden werden einige der oben aufgeführten Fragebogensysteme hier kurz vorgestellt.[74] Die Anpassung der Kriterien, sowie die Fragebogenentwicklung für dieses Softwareprojekt ist dann in Kapitel 7 dargestellt.

Kriterien nach Nielsen

Nach Nielsen ([Nie93, S. 26]) lässt sich die Beurteilung der Usability als „Maß eines Produktpotentials um die Bedürfnisse des Benutzers zu befriedigen“ in 5 Qualitäts-Komponenten beschreiben:

- **Learnability** – Erlernbarkeit
  Wie schwer ist es für einen, mit dem System nicht vertrauten, Benutzer die Funktionen des Programms zu verwenden um einfache Aufgaben zu bewältigen?
- **Efficiency** – Effizienz / Produktivität
  Wie schnell können Benutzer mit dem System Aufgaben lösen, wenn sie die Funktionsweise des Programms verstanden haben?
- **Memorability** – Einprägsamkeit
  Wie einprägsam ist die Funktionsweise des Programms? Lässt sich für einen nach einer Weile wiederkehrenden Benutzer das Programm leicht wiedererlernen?
- **Error** – Fehlerbehandlung
  Welche und wie schwerwiegende Fehler werden von Benutzern gemacht und wie gelingt es ihnen diese zu umgehen?
- **Satisfaction** – Zufriedenheit
  Wie zufrieden sind Benutzer mit dem System?

Ein auf Grundlage dieser Konzepte entwickelter Fragebogen sollte mindestens 10 Punkte umfassen, kann aber auch deutlich detailierter sein.
In [NL06, S. 122ff] wird ein weiterer Ansatz beschrieben um insbesondere schwerwiegenden Usability-Problemen auf die Spur zu kommen. Danach wird konkret nach bestimmten funktionalen sowie Design/Layout Elementen gefragt und die Ergebnisse nach der Einschätzung der Versuchspersonenen gewichtet.
Die Anzahl der Versuchspersonen sollte mindestens 5, besser aber 10 Personen betragen (vlg. [Nie00]).

Die sog. heuristische Evaluation nach Nielsen umfasst 10 Bewertungskriterien, die von unabhängigen Benutzerinterface-Experten bewertet werden (vgl. [Nie93]). Durch die Heranziehung von Experten, im Gegensatz zu „normalen“ Evaluation mit Versuchspersonen aus der Zielgruppe, ist es möglich sehr detailiert Schwachpunkte des Oberflächendesigns zu identifizieren (vgl. [Nie04]). Durch die Einschränkung auf eine Bewertung durch Benutzerinterface-Experten, lassen sich durch diese Evaluation kaum Rückschlüsse auf die Qualität der inhaltlichen Aufarbeitung gewinnen.

Kriterien des SUMI Fragebogensystems

Im SUMI Fragebogensystem werden fünf Dimensionen der Usability identifiziert (vgl. [GHD02, S. 9]):

[74] Da alle der angegebenen Quellen englischsprachig sind, wird hier der in der Originalquelle angegebene Name eines Konzeptes und eine Übersetzung übernommen.

- **Efficiency** – Effizienz / Produktivität
  Evaluiert, wie die Software einen Benutzer bei der Arbeit unterstützt.
- **Affect** – Affekt
  Misst die emotionale Reaktion eines Benutzers zu der Software.
- **Helpfulness** – Nützlichkeit
  Bezieht sich auf das Hilfesystem und evaluiert, wie selbsterklärend die Software auf einen Benutzer wirkt.
- **Control** – Kontrolle
  Misst, ob ein Benutzer das Gefühl hat, die Software zu kontrollieren.
- **Learnability** – Erlernbarkeit / Anwendungsschwierkeitsgrad
  Misst die Zeit und Aufwand, den ein Benutzer glaubt aufwenden zu müssen um die Software effektiv verwenden zu können.

Zusätzlich kann noch eine sechste Kategorie eingeführt werden, um eine globale Einschätzung der Software zu erlauben. Insgesamt besteht ein Fragebogen nach SUMI-Kriterien aus mind. 50 einzelnen Punkten und es wird empfohlen mind. 10 Versuchspersonen zu befragen um aussagekräftige Ergebnisse zu erlangen. Eine Erweiterung des Tests integriert dem schriftlichen Teil sich anschließende Interviews, um Diskrepanzen zwischen erwarteten Ergebnissen und beobachteten Ergebnissen zu klären.

Kriterien nach Quesenbery – 5 Dimensionen der Usability

In [Que03] werden fünf Konzepte der Evaluation (5Es) vorgestellt:

- **Effective** – Wirksamkeit
  Wie vollständig und korrekt erreichen Benutzer ihr Ziel mit der Software?
- **Efficient** – Effizienz
  Wie schnell und exakt bearbeiten Benutzer Aufgaben mit dem System?
- **Engaging** – Einnehmend
  Wie angenehm ist die Benutzung und wie zufrieden sind Benutzer mit dem Programm?
- **Error tolerant** – Fehlertoleranz
  Wie effektiv berücksichtigt das Programm Fehler und hilft dem Benutzer solche zu vermeiden?
- **Easy to learn** – Erlernbarkeit
  Wie einfach ist es für einen unerfahrenen Benutzer mit dem Programm umzugehen und die Benutzung zu erlernen?

Diese fünf Punkte sollten in einer dem zu evaluierenden Programm und potentieller Nutzergruppe anpassten Gewichtung beurteilt werden.

Kriterien nach Papitsch – Vorangegangene Evaluation

In [Pap07, S. 122ff] wurde ein auf der Kombination obiger Kriterien, aufbauendes Fragebogensystem entworfen. Dieses beinhaltet die folgenden fünf Untersuchungsdimensionen:

- **Satisfaction** – Zufriedenheit

- **Efficiency** – Effizienz / Produktivität
- **Usefulness** – Zweckmäßigkeit
- **Control** – Kontrolle
- **Learnability** – Erlernbarkeit / Anwendungsschwierkeitsgrad

Auf Grundlage dieser Kriterien wurde in [Pap07] ein Fragebogen entworfen mit insgesamt 30 Fragepunkten und jeweils 5 möglichen Bewertungen im Likert-Stil von „Zustimmung“ bis „Nicht-Zustimmung“. Zusätzlich konnte jede Frage mit einem „Nicht Zutreffend“ beantwortet werden. Dieser Fragebogen wurde herangezogen um ein diesem Projekt vorangeganges System (siehe Seite 21) zu evaluieren.

Allgemeines zur Fragebogeentwicklung

Die Punkte eines Fragebogen können als offene Fragen definiert werden oder wie in [Nie93, S. 33ff] vorgeschlagen in einer mind. 5-teiligen Likert-Skala. Dabei werden auf einer Skala möglichst klare Aussagen bzw. Fragen nach dem Grad der Zustimmung bewertet.
Die Verwendung von Likert-Skalen hat meist der Vorteil der leichteren Auswertbarkeit gegenüber freien Textangaben, die meist einer speziellen Interpretation bedürfen.
Ein Problem der Likert-Skala ist oft die Interpretation der mittleren Auswahl, da diese überdurchschnittlich häufig, aus den unterschiedlichsten Gründen, von Versuchspersonen gewählt wird (vgl. [Heg03, S. 35]). Dieser Fakt ist bei der Interpretation der Ergebnisse zu beachten.

4 Anforderungen

Jede Software muss bestimmten Spezifikationen genügen. Für die hier zu entwickelnde Software wurden einige dieser Anforderungen direkt aus der Aufgabenstellung entnommen bzw. folgen direkt aus diesen, während andere Anforderungen als Entwicklungs- und Designentscheidungen zu sehen sind. Weitere Entscheidungen und Einschränkungen wurden in Bezug auf die Basis- oder Ausgangsdaten und die verfügbare Technologie getroffen (siehe Kapitel 5).

In der Aufgabenbeschreibung wurde schon die Festlegung auf sog. *Web-Technologie* getroffen als Technologie zur Darstellung für den Benutzer. Bei dieser Art eines Programmsystems wird die Software auf einem zentralen Server vom Benutzer mit einem normalen Webbrowser-Programm (z. B.*Internet Explorer*, *Mozilla Firefox*, o.ä.) angesprochen. Der Vorteil dieser Lösung ist, dass keine besonderen Programme beim Benutzer installiert werden müssen (ein Webbrowser gehört meist zur Standardausrüstung eines normalen PC) und das Programm nur an einer Stelle (auf dem Server) installiert und gewartet werden muss. Diese Art der Programmbenutzung ist auch als *Client-Server* Architektur bekannt, hier mit dem Sonderfall, dass die Client-Station mit Standardsoftware und Standardprotokollen arbeitet und mit dem Server arbeitet.

Im Allgemeinen werden für ein Softwareprodukt mindestens drei verschiedene Typen von Anforderungen unterschieden [Poh07, S. 14ff][75]:

- **Funktionale Anforderungen:**
  Anforderungen, die die Möglichkeiten eines Benutzers mit dem System beschreiben. Grundsätzlich auch die Eignung des Systems für eine bestimmte Aufgabe.
- **Qualitätsanforderungen und nicht-funktionale Anforderungen**
  Definierte Anforderungen, die nicht direkt die Funktionalität beschreiben, aber stattdessen Qualitätsmerkmale wir Zuverlässigkeit, Genauigkeit, Reaktionsgeschwindigkeit etc. umfassen. Nichtfunktionale Anforderungen beinhalten Spezifikation zu Sicherheitsmerkmalen oder auch Datenschutz. Unter Umständen können hier auch „unterspezifierte“ funktionale Anforderungen angegeben werden.
- **Rahmenbedingungen** (*Constraints*)
  Rahmenbedingungen beinhalten Teile aus der Entwicklungsplanung wie verfügbare Zeit und Aufwand, aber auch technische und technologische Voraussetzungen[76] wie verfügbare Compu-

[75] Über die Aufstellung von Anforderungen gibt es in der Fachliteratur zum *Requirements Engineering* durchaus verschiedene Ansichten. An dieser Stelle genügt die gängigen Beschreibung, die auch in [Poh07] vertreten wird. Darüberhinaus lassen sich selbstverständlich viele Anforderungen noch weiter unterteilen und unterschiedlichen Anwendungsfällen zuordnen. Die dazu nötige komplette Analyse der Nutzergruppe sowie weitere Unterteilung der Anforderungen wird hier allerdings unterlassen, da dies außerhalb des Rahmens dieser Arbeit liegt und die jetzige Analyse für diese Arbeit ausreichend ist.

[76] Der Unterschied zwischen Technik und Technologie ist in der Informatik teilweise schwer bestimmbar, da verfügbare Technik, also z. B. ein bestimmtes Computersystem, immer auch eine bestimmte Technologie beinhaltet, also z. B. eine Technologie als Anwendung einer bestimmten Programmiertechnik. Ein weiteres Problem dieser beiden Begriffe ist die unterschiedliche Verwendung in anderen Sprachen, weshalb eine direkte Übersetzung, z. B. aus dem Englischen diffizil ist und gerade auch in der Informatik zu verschiedensten Bezeichnungen geführt hat. (Vgl. hierzu Diskussion in: [Wik07d])

tersysteme für die Entwicklung und den Einsatz der Software. Im Allgemeinen schränken Rahmenbedingungen die Menge der Lösungsansätze und möglichen Umsetzungen für die Software stark ein, daher sollte die Formulierung dieser Bedingungen immer wohl überlegt sein um keine Unmöglichkeit der Erfüllung von anderen Anforderungen entstehen zu lassen.

Um nun möglichst die Anforderungen präzisieren zu können, wurden die vorangegangenen Arbeiten analysiert und die funktionalen Anforderungen, Qualitätsmerkmale sowie Rahmenbedingungen abgeleitet. Zusätzlich kamen Anforderungen hinzu, die sich aus der Problemstellung (siehe Kapitel 1.4) und Bedingungen des Umfeld sowie den Fähigkeiten des Entwicklers ergeben.

Wie in Kapitel 1.5 beschrieben, sollte zunächst eine Vorabversion (Prototyp) entwickelt, eine Evaluierung mit dieser durchgeführt und aus den Ergebnissen eine endgültige Version erstellt werden. Aus den Ergebnissen der ersten Evaluierung konnten dann weitere Anforderungen an die Endversion erhoben werden. Daher werden in Kapitel 4.2 die ursprünglich aufgestellten Anforderungen dargestellt.

4.1 Allgemeine Anforderungen

Hier nun in tabellarischer Form Listen mit formulierten allgemeinen Anforderungen an das Programmsystem.
Diese Listen sind zunächst eingeteilt, wie oben beschrieben, in Funktionale Anforderungen, Nichtfunktionale bzw. Qualitätsanforderungen und Rahmenbedingungen. Es ist zu beachten, dass die Anforderungen zunächst für das Gesamtsystem gelten und unabhängig von der Ausführung als Websystem. Eine detaillierte Übersicht, welche Anforderungen mit welcher Technologie erfüllt werden können findet sich in Kapitel 5. Hier soll es erstmal nur um Anforderungen gehen, die im, Rahmen der Möglichkeiten, unabhängig von einem bestimmten Programmsystem oder Technologie sind.

Zunächst die **funktionalen Anforderungen** mit den, den Benutzern als Fähigkeit des Programms präsentierten, Funktionen.

Funktionale Anforderung	**Beschreibung**
Suche nach relevanten Dokumenten mit Suchbegriff(en)	Passende Dokumente zu einem Suchbegriff aus der Datenbasis heraussuchen und anzeigen. Dies ist eine der Grundfunktionen des Systems, mit dem aus einer großen Menge von Dokumenten potenziell Passende herausgesucht werden für eine weitere Bearbeitung.
Dokumente nach „simplen“ Kriterien sortieren	Sortieren/Rangfolge der Dokumente nach objektiven Kriterien wie: Veröffentlichungsjahr, Autor(en), Titel, usw. Diese Funktion hilft Benutzern sich mit 'ihren' Dokumenten zurechtzufinden und Überblick zu bekommen.
Dokumente in Gruppen aufteilen	Die Dokumente auf Gruppen verteilen nach festgelegten Kriterien, wie Zitationen oder Textähnlichkeit. Die Anzahl der zu bildenden Gruppen soll vom Benutzer bestimmbar sein oder ein Optimumwert automatisch ermittelt werden. Die Verfahren zur Bestimmung von Dokumentengruppen sind in Kapitel 3.1.1 beschrieben.
Stichworte zum Inhalt von Dokumentengruppen generieren	Zu jeder Gruppe eine Beschreibung in Stichworten generieren, die möglichst genau die Dokumente in der Gruppe beschreibt. Wenn die Stichworte gut die Eigenschaften der Dokumentengruppe beschreiben, hilft dies Benutzern bei der Auswahl von relevanten Dokumenten.
Beschreibungen und Namen von Gruppen ändern	Die Beschreibungen/Stichworte von Gruppen sollten vom Benutzer änderbar sein. Dies erlaubt es Benutzern eigene Einordnungen von Gruppen zu machen.
Dokumentengruppen erzeugen, löschen und verschieben	Benutzer sollen die Möglichkeit haben, Dokumentengruppen selbst anzulegen, zu löschen oder in der Reihenfolge zu verschieben. Dies lässt die Möglichkeit zu, eigene Ordnungskriterien auf Dokumentenmengen anzuwenden.
Aktionen auf ein Dokument:	
– Löschen	Dokument löschen, aus Ergebnismenge entfernen.
– Verschieben	Dokument in eine andere Gruppe verschieben.
– Detailansicht	Alle Details aus den bibliographischen Angaben zu einem Dokument anzeigen, sofern möglich auch das *Abstract*.

Funktionale Anforderung	Beschreibung
Tabellenansicht für Dokumente und Gruppen	Die Dokumente sowie umschließenden Gruppen sollen in einer tabellarischen Form dargestellt werden können. Dies soll eine der Grundansichten des Programms werden.
Baumansicht oder hierarchische Ansicht für Dokumente und Gruppen	Die Dokumente sowie umschließenden Gruppen in einer hierarchischen Form, wie z. B. einer Baumdarstellung visualisieren. Dabei sollen, im Rahmen der Darstellung möglich, die selben Funktionen pro Dokument vorhanden sein wie in der Listendarstellung.
Abspeichern des Such- und Bearbeitungsergebnisses	Um die Ergebnisse zu einem späteren Zeitpunkt anzusehen oder weiterzubearbeiten soll der Benutzer seine Daten in eine Datei auf seinem Rechner abspeichern und auch wieder ins Programm hochladen können.
Erstellen einer Literaturliste	Eine Literaturliste mit relevanter Literatur kann erstellt werden. Dazu sollen einzelne Dokumentinformationen (Metainformationen über ein Dokument) in anderen Formaten zur Verwendung/Weiterbearbeitung zur Verfügung gestellt werden können, z. B. in BIBTEX. Ein Download dieser Liste sollte in einem allgemein verbreiteten Format, wie z. B. *Portable Document Format* (PDF), möglich sein.
Mehrsprachige Oberfläche	Die Oberfläche soll nicht sprachabhängig sein, bzw. ein Benutzer soll sich seine präferierte Sprache aussuchen können. Dazu sollen mindestens zwei vollständig unterstützte Sprachen (z. B. Deutsch und Englisch) angeboten werden. Eine Konsequenz daraus ist, bei der Entwicklung besonders auf eine strikte Trennung von Code und Darstellung zu achten (nicht-funktionale Anforderung).

Tabelle 4.1: Anwenderfunktionen und Interaktionen im System als funktionale Anforderungen

Nicht-funktionale Anforderungen / Qualitätsanforderungen	Beschreibung
Plattformunabhängig für den Benutzer	Der Benutzer soll unabhängig (z. B. von einem bestimmten Betriebssystem oder Installation eines speziellen Programms) mit dem System arbeiten können, um nicht unnötige Schranken oder Hinderungsgründe zu schaffen.
Einfach und ohne viel Vorwissen benutzbar	Das System soll möglichst einfach benutzbar sein. Vorwissen zu Datenformaten von Metadaten und Ähnlichkeitsmaßen ist nicht Voraussetzung.
Datensparsamkeit	Es sollen keine für den Betrieb unnötigen Daten erhoben und gespeichert werden. Insbesondere, sofern nicht durch andere Funktionen erfordert, sollen keine Benutzernamen und/oder Passwörter gespeichert bzw. verwendet werden.

Nicht-funktionale Anforderungen / Qualitätsanforderungen	Beschreibung
Schnelle Antwort- und Reaktionszeiten des Systems	Das System soll zu anderen im Internet benutzbaren Systemen vergleichbare Antwort- und Reaktionszeiten aufweisen, um eine ausreichende Akzeptanz beim Benutzer zu erreichen. Auch sollen Wartepausen (z. B. bei der Kalkulation von Werten) dem Benutzer ausreichend kenntlich gemacht werden.
Serverplattform anpassbar	Die Serverplattform (Betriebssystem, Hardware) soll anpassbar bzw. nicht festgelegt auf eine bestimmte sein. Damit kann das System schneller auf einer anderen Plattform eingesetzt werden. Ein weiterer Effekt ist, dass eine bessere Trennung zwischen Betriebssystem und Anwendung erzwungen wird, da keine spezialisierten Funktionen verwendet werden können, die das Programm auf eine bestimmt Basis festlegen.
Grunddatenbasis ausbaubar	Die Grunddatenbasis, bzw. das Datenmodell zur Speicherung, soll es erlauben weitere Datenquellen zu integrieren. Dies ist für einen späteren Ausbaustand hilfreich.

Tabelle 4.2: Nicht-funktionale Entwicklungsanforderungen und Qualitätsanforderungen

Zum Schluss noch die **Rahmenbedingungen**, inklusive der Entwicklungsanforderungen (technisch und organisatorisch):

Rahmenbedingung	Beschreibung
Implementierung als Webanwendung	Das System soll als eine sog. *Webapplikation* implementiert werden. Hierbei wird das Programms ausschließlich mit einem Webbrowser gesteuert, was das Programm potenziell für einen großen Anwenderkreis erschließt. (Diese Bedingung ist außerdem Teil der Aufgabenstellung.)
3-Tier-Architektur	Unabhängig von der Client-Server Architektur, mit grafischer Darstellung ausschließlich auf dem Client-Rechner, soll die Datenhaltung in eine Datenbank ausgegliedert werden. Dies erlaubt eine höhere Modularisierung und Anpassbarkeit auf veränderte Gegebenheiten.
Grunddatenbasis lokal vorhanden	Die Grunddatenbasis für Auswertungen und Suchergebnisse sollte aus Geschwindigkeitsgründen lokal auf dem Server vorhanden sein.
Vorkalkulation von Werten	Falls möglich und nötig, sollen Werte (wie Ähnlichkeitsmaße) im Voraus berechnet werden und so möglichst schnell bereitstehen.

Rahmenbedingung	Beschreibung
Programm aus wenigen oder einer Komponente	Das Serverprogramm sollte möglichst nur aus einer oder wenigen Programmkomponenten bestehen (im Gegensatz z. B. zu einer Sammlung von vielen kleinen Programmen, die jeweils einen kleinen Teil berechnen oder CGI-Programmen). Das erlaubt es ein einheitliches „Look and Feel“ über alle Programmfunktionen zu implementieren und erhöht die Wartbarkeit.
Webentwicklung mit MVC-Techniken	Die GUI-Entwicklung soll mit Hilfe einer das `MVC` (*Model-View-Controller*) Architekturmuster unterstützenden Bibliothek erfolgen aus Gründen der Modularisierung, Wartbarkeit, schnelleren Entwicklung und grundsätzlichen Trennung von Programmcode/-logik sowie Präsentation und Darstellungskomponenten. Dies bedingt die Einbeziehung einer zusätzlichen Bibliothek, die später ausgewählt wird. Die 'normalen' Web-Entwicklungsplattformen (wie Java Servlet-API oder PHP), bringen eine solche Fähigkeit noch nicht mit.
Weboberfläche potenziell auf vielen Plattformen benutzbar	Es sollte eine Technologie ausgewählt werden, die zumindest potenziell auf vielen Plattformen darstellbar und verwendbar ist. Auch wenn eine Webapplikation vom Grundgedanken her auf vielen Plattformen und mit verschiedenen Browsern funktioniert, kann man durch Wahl einer inkompatiblen oder sehr spezialisierten Technologie diese Unabhängigkeit verlieren oder dadurch Barrieren aufbauen.
Objektorientierte Programmiersprache	Programmentwicklung mit einer objektorientierten Programmiersprache, um Modularisierung zu unterstützen und die Wiederverwendbarkeit von vorhandenem Code zu ermöglichen.

Tabelle 4.3: Rahmenbedingungen und technische Anforderungen

Die technischen Anforderungen / Rahmenbedingungen werden detailliert in Kapitel 5 besprochen (und dadurch potenziell in technischer Hinsicht erweitert) mit der Auswahl konkreter Bibliotheken und Anwendungsmöglichkeiten.

4.2 Anforderungen an einen Prototypen zu Evaluationszwecken

Nachdem die grundsätzlichen Eigenschaften der Software oben formuliert wurden, sollen hier kurz die Erfordernisse für eine erste Evaluation beschrieben werden. Bei dieser Evaluation sollen die Grundfunktionen der Software überprüft und das generelle Konzept der Gruppierung von Dokumenten auf Akzeptanz bei einer potenziellen Nutzerschaft getestet werden (siehe auch Kapitel 7.1). Für diese Evaluation müssen also nicht zwangsweise alle Oberflächen-, Auswerte- und Ein / Ausgabefunktionen vorhanden sein.

Die Vereinfachungen für eine Version zu Evaluationszwecken sind wie folgt:

Anforderung	**Beschreibung**
Einfachere Oberfläche	Verzicht auf Funktionen wie das Sortieren von Dokumenten oder manuelles Anlegen von Gruppen. Geringere Qualitätsanforderungen, was z. B. Reaktionsgeschwindigkeit angeht.
Manuelle Angabe von Gruppenanzahl	Für das Bilden von Dokumentengruppen eine vom Benutzer vorgegebene Zahl verwenden und keine automatische Erkennung eines Optimums.
Gruppierung von Dokumenten nach Bibliographischer Kopplung	Bei der Gruppierung von Dokumenten wird ausschließlich das Kriterium der Bibliographischen Kopplung benutzt. Dies erlaubt einem potenziellen Benutzer einen genügenden Eindruck von den Gruppierungsmöglichkeiten, die das Programm bietet.
Kein Im- & Export von Ergebnissen	Das Modul zum Im- und Export von Ergebnissen ist für die erste Evaluation nicht erforderlich. Außerdem kein Export bzw. Umwandlung von Dokumenteninformationen in andere Formate (z. B. BIBTEX).
Eine Sprachversion	Auch um gleiche Voraussetzungen für eine Bewertung zu machen, sollte nur eine Sprachversion verwendet werden. Die Versuchspersonen sollten aber über die potenziell weiteren Sprachvarianten Kenntnis haben, um die Verwendbarkeit des Systems einzuschätzen.

Tabelle 4.4: Anforderungen an Evaluationsversion

5 Auswahl einer technischen Basis

5.1 Analyse der Anforderungen und Übersicht zu möglichen Lösungen

Wie in Kapitel 4.1 als Rahmenbedingung beschrieben, soll ein *web*basiertes System für die interaktive Arbeit mit einer potentiell großen Datenbasis entstehen. Daher sind in diesem System mehrere Teile genauer zu spezifizieren: Client (Webbrowser) und Server.

Auch wenn die Beschreibung: *Webbrowser* als standardisiert erscheint, gibt es diverse Varianten, die es zu berücksichtigen gilt. Insbesondere, ist dies wichtig, um den Anforderungen zu einem möglichst reaktionsschnellen und graphisch anspruchsvollem interaktiven System zu genügen. Eine Übersicht über Möglichkeiten und einsetzbarer Technologie bietet Kapitel 5.2.

Auf der Serverseite sind die Vorgaben nicht so eng gefasst, so dass verschiedenste Implementierungen und Basistechniken in Frage kommen. Die Auswahl einer Programmiersprache und Programmumgebung findet sich in Kapitel 5.3, eine Übersicht über möglichen Einsatz von Toolkits / Bibliotheken findet sich in Kapitel 5.3.3.

Eine Betrachtung und Auswahl der von verwendbaren Daten (siehe auch Kapitel 2) ist am Ende diese Kapitels im Abschnitt 5.5 dargestellt.

5.2 Client

Ein sog. *web*basiertes System besteht allgemein aus dem Server sowie Clients, die mind. aus einem Webbrowser bestehen. Ein Webbrowser kann im allgemeinen mindestens *HTML* als Seitenbeschreibungssprache interpretieren. Darüberhinaus können auf dem Clientsystem verschiedene Technologien und Programmierschnittstellen, teilweise als PlugIn realisiert, verwendet werden. Webseiten, die eine solche Kombination verwenden, werden häufig auch als „Rich Internet Applications“ (RIA) bezeichnet, da sie das Aussehen und die Funktionalität von herkömmlichen Desktop-Applikationen haben, aber im Webbrowser dargestellt werden. Das hier zu entwickelnde System fällt unter diese Kategorie, da die Anforderungen, insbesondere die Qualitätsanforderungen zu Reaktionsgeschwindigkeiten, nicht mit einfacher HTML-Programmierung zu erreichen sind. Darüberhinaus sind meist graphisch anspruchsvolle Web-Oberflächen problemloser und benutzerfreundlicher durch zusätzliche Technologien realisierbar (vgl. [O'R04]).
Einige dieser Techniken sind als zusätzliche Software zu installieren, während einige auch schon bei einer Standardinstallation des Betriebssystems bzw. Browsers dabei sind.

Eine Auswahl sollte sich an den Kriterien: Verfügbarkeit, freie Zugänglichkeit (also keine rein kommerzielle Systeme), und mit der Technik zu erlangenden Möglichkeiten orientieren.
Die z. Z. gängigsten PlugIns bzw. Technologien sind:

- **Flash**[77]
  Dieses ursprünglich von *Macromedia* und heute von *Adobe* weiterentwickeltes Format ist wohl das am weitesten verbreitetste Format für Multimediainhalte im Internet. Das PlugIn existiert für die meisten gängigen Betriebssysteme (Windows, MacOS, Linux und verschiedene andere) und erlaubt es Medieninhalte verschiedenster Art (Bilder, Videos, Audio) in eine Webseite einzubetten. Mit Flash realisierte Anwendungen reichen von einfachen 'MouseOver'-Effekten, Lauftexten, Videosequenzen bis hin zu interaktiven Spielen.
  Anwendungen für dieses Format können nur zum Teil mit Standardsoftware erstellt werden, für komplexere Applikationen wird eine spezielle kommerzielle Entwicklungsumgebung benötigt. Die Laufzeitumgebung (das PlugIn) bringt darüberhinaus eine Programmiersprache *ActionScript* mit, die *JavaScript* sehr ähnlich ist und von *Adobe* auch aktiv zur Integration in die nächste Version von *Mozilla Firefox* beworben wird. Ein Webdesigner hat die Möglichkeit Flashelemente als Einzelelemente in die Seite zu integrieren, oder auch die ganze Seite damit zu gestalten. Die Interaktion mit anderen Elementen der Seite wird über *ActionScript* realisiert was es teilweise recht komplex zu handhaben macht. Das Sicherheitskonzept sieht vor, dass *Flashelemente* nur auf sich selbst und die sie umgebende Seite Zugriff haben, also insbesondere nicht auf andere Dateien oder Resourcen des Computers.
  Eine gute Übersicht über die Möglichkeiten von *Flash* bietet auch der Übersichtsartikel in Wikipedia [Wik07a].

- **Silverlight**[78]
  Dieses Format ist noch recht neu und bislang nur wenig verbreitet. Es wurde von *Microsoft* als Konkurrenz zu *Adobe Flash* entwickelt und basiert auf *.NET* Framework bzw. der Programmiersprache *C#*. Es bietet in etwa die gleichen Features wie *Flash*, aufgrund der deutlich komplexeren und umfangreicheren Programmiersprache aber potentiell mehr Möglichkeiten und Verbindungen zu anderen Techniken.
  Bisher existiert diese Technologie aber erst in Version 1 und auch nur für neuere *Microsoft*-Betriebssysteme. Ein entsprechendes PlugIn für MacOS ist bisher erst angekündigt und von *Novell* wird eine Linux-Portierung entwickelt, insgesamt gesehen ist aber die Verbreitung noch sehr gering. Dies könnte sich aber in wenigen Jahren ändern, z. B. wenn Microsoft Betriebssysteme bei Auslieferung oder Update automatisch mit dieser Technologie versorgt werden und / oder große Internetdienstleister diese Technologie adaptieren.

- **Curl**[79]
  Diese Technologie ist ähnlich der von Java oder auch Flash, zielt aber hauptsächlich auf „Enterprise Web Applications“ und sowohl eine Client- als auch Serverplattform.
  Es existieren PlugIns für die verbreiteten Browser und Betriebssysteme wie Windows, Linux und MacOS, es wird aber für den produktiven Einsatz notwendigerweise die zugehörige Serverkomponente benötigt, die nur kommerziell verfügbar ist. Für Privatanwender existiert zwar eine nicht-kommerzielle Version, allerdings ist diese in der Verwendung (Lizenz) sehr eingeschränkt und bietet nicht alle Funktionen.
  An Sicherheitsfunktionen bietet Curl auf der Clientseite ein Sandbox-Konzept ähnlich der Java-Technologie: Es werden grundsätzlich alle Zugriffe auf den Clientrechner unterbunden, es sei denn, der Benutzer hat ausdrücklich zugestimmt. Curl-Applets können darüber hinaus signiert

[77] Herstellerinformationen: http://www.adobe.com/de/products/flashplayer/, Stand 2007-10-04
[78] Herstellerinformationen: http://silverlight.net/, Stand 2008-03-01
[79] Herstellerinformationen: http://www.curl.com/products_platform.php, Stand 2007-11-05

werden um dauerhaft zu höheren Zugriffsrechten zu gelangen.
Die Verbreitung dieser Technologie ist, insbesondere im Vergleich zu anderen hier Aufgeführten, eher als sehr gering einzustufen und aufgrund der rigiden Lizenzpolitik und der beabsichtigten Zielgruppe auch kaum wachsend. Damit ist Curl eher als ein „Exotensystem“ zu sehen.

- **SVG**[80]
  Scalable Vector Graphics (SVG) ist zunächst nur ein Grafikformat, mit dem vektorbasierte Daten, Rasterdaten und Text dargestellt werden können. Das Format ist offen vom *(W3C)* spezifiziert und wurde zeitweise als direkter Konkurrent zu Flash gesehen. Ziel war es, ein Grafikformat für eine Vielzahl von Anzeigegeräten von mobilen Endgeräten (Mobiltelefone beispielsweise) über Webbrowser, Set-Top-Boxen für Fernseher bis hin zu Druckformaten zu schaffen.
  Mittlerweile wird SVG von einigen Webbrowsern wie *Opera* und *Mozilla Firefox* zumindest mit den Grundfunktionen direkt unterstützt, die Verbreitung von SVG-Grafiken im Internet ist aber trotzdem als gering einzustufen. Grund dafür ist auch eine direkte Konkurrenz zu etablierten und noch zu etablierenden Formaten wie *Flash* oder *Silverlight* von großen Softwareproduzenten, die mutmaßlich nur wenig Interesse an dem Format haben. So hat die Firma *Adobe* ihre Unterstützung des Formats weitgehend eingestellt, nachdem sie durch Aufkauf einer Firma selbst zum Hersteller der *Flash*-Technologie wurde.
  SVG unterstützt skalierbare Vektorgrafiken, Rastergrafiken und Text. Daneben besteht z. B. die Möglichkeit mittels *JavaScript* die Darstellung zu beeinflussen, zu animieren oder auch weitere Elemente nachzuladen. Die Unterstützung hierfür ist aber stark vom Anzeigesystem abhängig.
  Die Verbreitung des Formats ist sehr uneinheitlich. Relativ viele mobile Endgeräte, insbesondere sog. „Smartphones“, unterstützen das Format direkt, einige Webbrowser sind, auch ohne zusätzliche Software, fähig, SVG-Grafiken anzuzeigen und die Verbreitung von Werkzeugen, wie Grafikprogrammen, die SVG erzeugen können wächst stetig. Darüberhinaus gibt es Bestrebungen SVG als Standardgrafikformat für Illustrationen in der *Wikipedia* zu machen. Trotz allem ist dieses Format z. Z. aber eher selten im Internet gesehen.

- **Java**[81]
  Dies ist im eigentlichen Sinne keine Browser-Technologie, kann aber im Webbrowser als PlugIn installiert werden. Java umfasst als sog. *Middleware*-Technologie sowohl eine Ausführungsumgebung, die sog. *Virtual Machine* (VM), eine dazugehörende Programmbibliothek als auch eine Programmiersprache. Es ist möglich die VM als ein PlugIn in gängigen Webbrowsern zu installieren, womit sich sog. *Applets* auf eine Webseite integrieren kann.
  Die Programmiersprache Java und die dazugehörende Ausführungsumgebung sind sehr mächtig, was es ermöglicht komplexe Programme auf diese Art und Weise zu implementieren. Allerdings ist es nicht möglich, dass ein Applet auf Ereignisse außerhalb der Applet-Anzeigefläche reagiert; so ist es z. B. nicht möglich, mit im Browser dargestellten HTML-Elementen zu agieren. Daher macht dieses Art der Anwendung nur Sinn bei unabhängigen Gestaltungselementen (bspw. Visualisierungselementen) oder wenn die gesamte Seite aus dem Applet besteht.
  Java als Webbrowser-PlugIn ist im Allgemeinen selten von Hause aus installiert. In den meisten Fällen ist der Benutzer gezwungen die Software selbst zu installieren, was vom Hersteller *Sun Microsystems* zwar sehr komfortabel gemacht wurde, aber trotzdem einige Benutzer abschrecken dürfte. Ein weiterer Negativpunkt ist die oft merkliche Verzögerung, die beim Start von Applets

[80] Webseite zur Standardisierung: http://www.w3.org/Graphics/SVG/, Stand 2007-11-05
Viele Informationen zum Einsatz finden sich auch unter: http://www.selfsvg.info/, Stand 2007-11-05
[81] Herstellerinformationen: http://java.sun.com/, Stand 2007-11-04

im Browser auftritt, wenn die VM geladen wird.
Aus Sicherheitsgründen darf ein Java-Applet nur Daten von dem Server empfangen, von dem auch die Seite geladen wurde. Diese Beschränkung kann nur mit einer Software-Signatur und entsprechendem Zertifikat umgangen werden. In diesem Fall muss ein Benutzer explizit zustimmen. Darüber hinaus läuft ein Java Applet immer in einer sog. „Sandbox“, deren Sicherheitsbestimmungen einstellbar sind und in der viele Operationen zunächst deaktiviert sind.

- **ActivX**[82]
  Dies ist mehr eine Schnittstellentechnologie als ein PlugIn und ausschließlich für Internetbrowser vom Hersteller *Microsoft* (*Internet Explorer*) verfügbar. Mit ActivX können, ähnlich wie bei *Flash* und *Java* Multimediaelemente oder auch Steuerungselemente in die Webseite eingebettet werden. Allerdings bestehen diese Elemente aus komplexen, plattformspezifischen Programmen, die durch den enthaltenen Maschinencode alle Möglichkeiten haben, die auch der aktuell am System angemeldete Benutzer hat.
  Aktuell wird diese Schnittstelle von wenigen Systemen direkt genutzt, ein Beispiel für eine direkte Nutzung ist das Update-System für *Microsoft-Windows*. Aus technischen Gründen wird diese Schnittstelle von einigen PlugIn-Herstellen für ihr System genutzt.
  Ein Sicherheitskonzept kennt diese Schnittstelle nur zum Installationszeitpunkt. Sobald ein Seite ein sog. *ActivX Control* installiert hat, kann diese Seite auf alle Ressourcen des Rechners zugreifen (sofern der Benutzer die nötigen Zugriffsrechte besitzt).
  Die Verfügbarkeit dieser Schnittstelle ist beschränkt auf den Browser *Internet Explorer* unter dem Betriebssystem Windows und kann systembedingt auch nicht einfach auf andere Browser und Plattformen angepasst werden.

- **PDF**[83]
  Dies ist zunächst weniger eine Schnittstelle, denn ein Dateiformat: *Portable Document Format*. Es existiert aber ein Browser-PlugIn, dass solche Dateien in eine Webseite einbetten kann. Da PDF-Dateien meist Dokumente darstellen, die einfach nur angezeigt werden sollen, wird das PlugIn meist seitenfüllend verwendet.
  Innerhalb PDF-Dateien ist es möglich mittels *JavaScript* aktive Inhalte zu übermitteln, diese Möglichkeit wird aber bisher außerhalb einiger Formularanwendungen nur selten benutzt. Der Zugriff auf Elemente der einbettenden Seite oder des Browsers ist innerhalb des PDF-PlugIns für dort angezeigte Inhalte nicht möglich.
  Auch wenn der *Adobe PDF Reader* selten bei der Installation eines Rechners von Hause aus dabei ist, haben ihn viele Nutzer installiert; die Verbreitung ist als recht hoch einzuschätzen.
  An Sicherheitskonzepten bringt der *Adobe PDF Reader* nur mit, dass bei Zugriff auf das Internet oder anderer Ressourcen mittels aktiver Inhalte eine Rückfrage beim Benutzer erfolgt. Darüberhinaus sind die Möglichkeiten aktiver Inhalte auf das aktuelle Dokument beschränkt.

- **JavaScript / ECMAscript**[84]
  Diese Technologie und Programmiersprache ist mittlerweile seit einigen Jahren sehr verbreitet und erfolgreich, auch, weil sie in die meisten Browser direkt eingebaut und nicht erst installiert werden muss. Auch wenn der Name dies nahelegt, hat diese Technologie nichts mit *Java*

[82] Herstellerinformationen: http://activex.microsoft.com/activex/activex/, Stand 2007-11-06
[83] Herstellerinformationen: http://www.adobe.com/devnet/pdf/, Stand 2008-03-02
[84] Mozilla Entwicklungsseite mit Informationen zur Sprache und Integration in Browser: http://developer.mozilla.org/en/docs/About_JavaScript, Stand 2008-03-02

zu tun; bestenfalls ist die Syntax einiger Programmelemente dieser ähnlich. Der Name *ECMAscript* ist aus dem Standardisierungsprozess der *European Computer Manufacturers Association* (ECMA) entstanden und vereinigt verschiedene Ansätze, die aus den Entwicklungen der beiden Firmen *Netscape* und *Microsoft* enstanden sind.
JavaScript ermöglicht es u. a. innerhalb einer Webseite auf Elemente einer Seite zuzugreifen, diese zu ändern und/oder auszutauschen. Desweiteren sind viele Standardelemente gängiger Programmiersprachen vorhanden und in Browsern neuerer Generation (ab z. B.*Internet Explorer 5.5*, *Mozilla Firefox 1.0*) ist es möglich per JavaScript eigenständig Anfragen an den Server zu schicken, ohne dass die Seite neu geladen werden muss. Diese Technik, meist unter dem Namen Asynchronous JavaScript and XML (AJAX) bekannt, wird bei vielen modernen Seiten extensiv eingesetzt und wird als Grundtechnik des sog. *Web 2.0* angesehen. Eine weitergehende Übersicht und Erklärung des Begriffs ist in [O'R05] von Tim O'Reilly nachzulesen.
Ein weiteres interessantes Feature bieten JavaScript-Implementierungen in aktuellen Browsern mit dem schon zu HTML 5 gehörenden `Canvas`-Element, das es erlaubt mittels 2D-Zeichenfunktionen beliebige Grafiken darzustellen. Diese Funktion ist allerdings noch als experimentell einzustufen, da in gängigen Browsern das Zeichnen noch stark durch Geschwindigkeit- und Kompatibilitätsprobleme behindert wird.
An Sicherheitskonzepten hat JavaScript im Wesentlichen zu bieten, dass Programmelemente in einer Art Sandbox (wie auch schon *Java*) laufen und außer zu den Elementen Webseite und dem Server, von dem die Seite geladen wurde, keinerlei Verbindung aufnehmen können (*Same-Origin Policy*).

Auswahl einer Variante

Viele dieser Technologien werden alleinstehend verwendet (z. B. für zusätzliche Funktionen, wie Visualisierung, Werbung, etc.). Eine weiterer Schwachpunkt einiger Techniken (bspw. *Java*, *Flash* usw.) ist die mangelhafte Barrierefreiheit, die nach heutigen Kenntnissen auch nicht einfach nachgerüstet werden kann.[85] Die *JavaScript* Technik wird zumindest bei Verwendung bestimmter Techniken als zugänglich für sehbehinderte Menschen gesehen.[86]

Da *JavaScript* die einzige auf allen Browsern und Betriebssystemen verbreitete Technik ist, kann man zunächst also nur diese voraussetzen. Leider variieren die Implementationen in verschiedenen Browsern und Versionen zum Teil erheblich, so dass zur Vereinfachung der Entwicklung zunächst das System auf einen „Referenz"-Browser angepasst wird. In einer späteren Version sollten dann Anpassungen für andere Browser möglich sein. Dabei ist darauf zu achten, dass bei der Entwicklung keine proprietären Erweiterungen eines Browsers verwendet werden, die in anderen Systemen nicht zur Verfügung stehen.
Für den „Referenz"-Browsers fiel die Wahl auf den *Mozilla Firefox*[87], insbesondere da dies ein Browser ist, der auf allen gängigen Betriebssystemplattformen zur Verfügung steht und eine moderne Implementation der *JavaScript* Sprache bietet.

Der Einsatz einer weiteren Technik, z. B. zur Visualisierung von Zusammenhängen im Programm, darf konsequenterweise nur optional sein.

[85] *Flash* bietet wenigstens eine Schnittstelle für Screenreader, zumindest für textuelle Inhalte.
[86] Eine Standardisierung dieser Techniken ist in Vorbereitung: `http://www.w3.org/TR/wai-aria-primer/`, Stand 2008-03-02; Ein Leitfaden zur Verwendung gibt: `http://www.w3.org/TR/wai-aria-practices/`, Stand 2008-03-02.
[87] Download möglich unter: `http://www.mozilla.com/`, Stand 2008-03-02

5.3 Server

In diesem Abschnitt geht es um die Auswahl von Techniken für die Serverkomponente des Systems. Zu Beginn einer Programmentwicklung (bzw. der Programmierung) muss eine Programmiersprache gewählt werden, die den Erfordernissen entspricht und deren Einsatz im Rahmen der vorhandenen Mittel möglich ist. Hier (Abschnitt 5.3.1) sind daher nur Programmiersprachen untersucht, die auch ohne Zahlung von Lizenzgebühren oder den Kauf von Entwicklungsumgebungen einsetzbar sind.

Neben der Wahl einer Programmiersprache wird die Frage nach der Einbindung der Webapplikation untersucht bzw. die Frage, wie die Anfrage eines über das Internet mit dem Server verbundenen Client an ein Softwareprogramm weitergeleitet werden kann. Im Abschnitt zu Basistechnik (siehe Abschnitt 5.3.2) sind verschiedenen Möglichkeiten dazu vorgestellt. Dazu soll hauptsächlich zusätzliche Serversoftware wie z. B. Webserver oder sog. „Standardsoftware“ untersucht werden, die aber nicht unmittelbar zum hier entwickelten Programm gehört.

Nach der Auswahl einer Programmiersprache wird eine Laufzeitumgebung bzw. sog. „Rahmensoftware“ (*Framework*) ausgewählt und festgelegt. Dies ist nötig, da bestimmte Rahmenbedingungen an die Softwareentwicklung geknüpft sind, die keine der vorhandenen Programmiersprachen aus dem Stand erfüllt. Daher ist zusätzliche Software vonnöten. Eine Übersicht und Auswahl ist in Abschnitt 5.3.3 zu finden.

5.3.1 Programmiersprache

Die Auswahl der Programmiersprache, und damit auch ein großer Teil der Programmumgebung, ist ein essenzieller Teil des Systementwurfs.
Es gibt viele Möglichkeiten und Kriterien, welche Programmierumgebung gewählt werden sollte. Entscheidenden Einfluss haben die aufgestellten Anforderungen (Kapitel 4.1), Anforderungen, die sich aus anderen Anforderungen ergeben und natürlich auch Erfahrung und Fähigkeiten der Entwickler sowie besondere Vorlieben.[88]
Eine Auswahl und kurze Erklärung zu in Frage kommenden Programmiersprachen, die insbesondere bei der Entwicklung von Web-Applikationen oft verwendet werden, ist hier dargestellt (vgl. auch [HV07]):

- **C++**
  Eine in den 1980er Jahren aus der Programmiersprache **C** weiterentwickelte Programmiersprache, die sowohl objektorientierte Programmierung als auch klassisches prozedurales Design unterstützt.
  Diese Sprache ist sehr weit verbreitet und bietet gute Unterstützung durch zusätzliche Werkzeuge (Compiler und Entwicklungsumgebungen), ist aber auch sehr komplex und in einigen Merkmalen kompliziert zu handhaben.
  Trotz der sehr großen Verbreitung von Werkzeugen wird C++ eher selten für Webapplikationen eingesetzt, was möglicherweise an Sicherheitsproblemen früherer Programme oder an fehlenden (bzw. nur wenigen) Bibliotheken liegt, die das Programmieren für das Web erleichtern.

[88] Anmerkung: Dies geht davon aus, dass ein Programm bedeutend schneller und besser entwickelt werden kann, wenn die Entwickler mit der Sprache vertraut sind. Dies schließt natürlich nicht aus, dass Entwickler eine Sprache umfassend erlernen können, was aber potenziell bedeutend länger dauert.

- **PHP**[89]
  Steht als rekursives Akronym für „PHP: Hypertext Preprocessor“ und ist eine Skriptsprache, die serverseitig (meist direkt vom Webserver oder als CGI-Applikation) interpretiert wird und sehr weit verbreitet ist für die Gestaltung von dynamischen Webseiten oder Webanwendungen.
  Die Struktur von PHP ist zunächst strikt prozedural, seit Version 5 wird aber auch objektorientierte Programmierung unterstützt.[90]
  PHP ist von Hause aus auf Programme für die Anwendung auf Webseiten ausgelegt, es ist bspw. möglich PHP-Anweisungen und HTML-Code zu mischen. Dabei wird der PHP-Code mit speziellen Tags vom HTML abgetrennt. Mit diesem Modell sind zwar relativ schnell einfache dynamische Seiten zu realisieren, aber für größere Projekte wird es schnell unübersichtlich. Daher gibt es mittlerweile auch eine Vielzahl von Projekten und auf PHP aufbauenden Rahmenwerken, die eine strukturierte Programmierung erlauben (vgl. [DP07]).

- **Ruby**[91]
  Ruby ist eine lange Zeit nur in Japan bekannte Skriptsprache, die vollständig objektorientiert arbeitet und seit ein paar Jahren in der westlichen Welt Bekanntheit erlangt hat. Die Syntax der Sprache ist recht eingängig, klar und fehlertolerant. Die Erweiterung „Ruby On Rails“, die sich zunehmender Beliebtheit erfreut, ist ein Framework, das dem *Model-View-Controller* Konzept bei der Entwicklung von komplexen Applikationen folgt und als sog. „Rapid Developing Tool“ gilt.
  Allgemein kann man aber sagen, dass die Entwicklung von Web-Applikationen mit Ruby nicht sehr verbreitet ist, die Sprache aber aufgrund der eingängigen Struktur und der sehr lebhaften Anwenderschaft noch viel Potenzial hat.

- **C#**[92]
  Diese Programmiersprache wurde ursprünglich von *Microsoft* erfunden und bietet sehr ähnliche Konzepte wie Java an. Dabei läuft die Ausführung des Programms in einer virtuellen Maschine *Common Language Runtime* (CLR) ab und bietet daher ähnliche Sicherheitsmerkmale wie die Java-VM.
  C# ist eine der Grundtechnologien der Firma *Microsoft* bei ihrer .NET Strategie und wird sowohl für Webapplikationsprogrammierung (*Active Server Pages*), als auch für aktive Multimediainhalte (Silverlight, siehe 5.2) verwendet. In der langfristigen Planung soll C# die bisher oft verwendete Sprache *Visual Basic* in *Microsoft* Produkten ersetzen.

  Zurzeit ist diese Sprache überwiegend auf die Windows-Plattform begrenzt, eine Portierung auf andere Plattformen wie Linux und MacOS macht aber Fortschritte z. B. durch das Mono-Projekt. Grundsätzlich gelten für C# die gleichen Einschränkungen wie bei Java. Viele Bibliotheken und Programme sind mit der neuen Sprache nicht direkt weiterverwendbar, allerdings deckt die von *Microsoft* mitgelieferte Klassenbibliothek viele Grundbedürfnisse ab und ähnlich Java's JNI erlaubt die Sprache in anderen Programmiersprachen geschriebene Programmteile als sog. „unmanaged Code“ weiterzuverwenden.
  Von einer echten Plattformunabhängigkeit, wie bei Java, ist es allerdings noch weit entfernt, da zum Beispiel die Klassenbibliotheken von *Microsoft* unter anderen Betriebssystemen nicht

[89] Infos: http://www.php.net/, Stand 2008-03-02
[90] Genauer gesagt, war objektorientierte Programmierung schon seit Version 4 möglich, allerdings mit gravierenden Einschränkungen, die ein objektorientiertes Programmdesign weitgehend unmöglich machten.
[91] Infos: http://www.ruby-lang.org/, Stand 2008-03-02
[92] Herstellerinformationen: http://msdn.microsoft.com/vcsharp/, Stand 2008-03-02

verwendet werden können. Daher haben Portierungen der Laufzeitumgebung noch große Unterschiede zur Originalimplementation. Nichtsdestotrotz ist abzusehen, dass die Verbreitung von C# in den nächsten Jahren deutlich hinzugewinnen wird, da *Microsoft* die `.NET`-Umgebung bei neueren Windows-Versionen automatisch mit ausliefert.

- **Java**[93]
  Diese Sprache, von *Sun Microsystems* entwickelt als Interpretersystem, bringt moderne objektorientierte Konzepte mit, deren zugrundeliegenden Datentypen einzig Primitive (also keine Objekte) darstellen. *Java* bezeichnet aber nicht nur eine Programmiersprache, sondern eine ganze Technologie, die eine Virtuelle Maschine [VM] als Code-Interpreter, Klassenbibliotheken und Programmiersprache umfasst. Die Verbreitung ist mittlerweile enorm groß, auch weil diese Technik für verschiedenste Betriebssysteme und Plattformen von mobilen Handgeräten (PDA, Handy) bis zu Großrechenanlagen verfügbar ist. Eine besondere Stellung ist durch den Einsatz als Multimedia-Technik im Internet (siehe 5.2) gegeben.
  Die weiteren Vorteile von *Java* sind:
  - Gute Runtime- und Framework-Bibliotheken, sowohl mit freier als auch mit kommerzieller Software. Die mitgelieferte Klassenbibliothek deckt alle Grundbedürfnisse ab.
  - Java-Software kann ohne Neukompilierung auf verschiedensten Plattformen laufen, wenn es eine kompatible Java-VM für diese Plattform gibt. Für alle gängige Betriebssysteme ist diese Bedingung erfüllt.
  - Große Auswahl von Interfaces und Konnektoren zu anderen Systemen und Technologien. Das betrifft zum einen Schnittstellen zu Datenbanksystemen, Internet Services (z. B. Webservices auf SOAP-Basis) oder auch Benutzeroberflächen in 2D- und 3D-Grafik.
  - Kommerzieller Support möglich und Sprache sowie Laufzeitumgebung werden ständig weiterentwickelt von *Sun Microsystems*.
  - Leicht zu erlernen, da viele Konzepte von älteren Sprachen wie C/C++ übernommen wurden. Allerdings wurden einige komplexe Möglichkeiten von anderen Sprachen z. B. C++ nicht übernommen, wie dynamische Operatorüberladung, Funktoren, etc. Dies macht einige Anwendungen in Java etwas „sperriger“, als in anderen Sprachen, allerdings oft auch deutlich klarer verständlich.
  - Für spezielle Einsatzzwecke existieren besondere Versionen der Laufzeitumgebung und Virtual Machine, wie die *Micro-Edition* für Geräte, bei denen es auf geringen Speicherbedarf und kleine Programme ankommt (z. B. Mobiltelefone) oder die *Enterprise Edition J2EE*, die über den normalen Standardsatz von Bibliotheken zusätzliche Fähigkeiten für den Einsatz im Unternehmen mitbringt.
  - Mit den *Java Server Pages* (JSP) existiert ein standardisiertes Format für die Generierung von dynamischen Webseiten mit eingebetteten Javaprogrammfragmenten ähnlich *PHP*.

  Darüber hinaus sollen natürlich auch ein paar Nachteile von Java nicht verschwiegen werden: Es ist oft langsamer, als vergleichbare Implementationen in Programmiersprachen, die z. B. direkt Maschinencode produzieren (z. B. `C/C++`, `D`). Dies ist allerdings sehr stark abhängig von Problem, Programmierung und Einsatz der Software. So kann eine in Java geschriebenes Software durchaus ebenbürtig oder sogar schneller als sein in C++ geschriebenes Gegenstück sein. Ein weiterer potentieller Nachteil von Java ist die (bedingte) Inkompatibilität zu anderen Programmiersprachen und Bibliotheken. Dies kann insbesondere ein Problem darstellen, wenn vorhandene Programme und Bibliotheken weiterbenutzt werden sollen. Ein Java-Programm kann zunächst

[93] Herstellerinformationen: `http://java.sun.com/`, Stand 2008-03-02

nur mit anderen in Java entwickelten Komponenten interagieren. Ein Ausweg stellt hier nur das etwas kompliziert zu benutzende *Java Native Interface* (JNI) dar.

- **Groovy**[94]
  Hier nur der Vollständigkeit halber erwähnt: Mit Groovy besteht eine weitere Möglichkeit die Java-VM zu nutzen, allerdings durch eine andere Programmiersprache als Java selbst. Groovy selbst verwendet Konzepte aus Java und Ruby und die Syntax stellt eine Mischung aus beiden Sprachen dar. Vorteil dieser Sprache ist zweifellos die Möglichkeit bestehenden Java-Code und Klassen weiterverwenden zu können, zudem ist Groovy mittlerweile in den Java-Standardisierungsprozess einbezogen.
  Interessant an Groovy für Webprogrammierung ist die direkte Möglichkeit über eine eingebaute Templateengine HTML- und SQL-Code zu erzeugen.

- **Perl, Pike, Python, etc.**
  Es gibt einige weitere Sprachen, die relativ häufig für die Programmierung dynamischer Webseiten verwendet werden. Trotz das in einigen dieser Sprachen sind auch OOP Konzepte verwirklicht sind (Pike, Python, Perl teilweise), ist die Programmierung von größeren Systemen mit dem Großteil dieser Sprachen eher kompliziert, da z. B. keine Namespaces bzw. Modularisierung oder die Web-Programmierung unterstützenden Bibliotheken vorhanden sind. Des weiteren fehlt es an unterstützenden Werkzeugen, wie eine spezialisierte Integrierte Entwicklungsumgebung (IDE).
  Es ist selbstverständlich trotzdem möglich mit einer dieser Sprachen auch komplexe Systeme zu entwickeln. Allerdings ist anzunehmen, dass ein mit der Sprache nicht vertrauter Entwickler, bedeutend länger mit der Programmierung verbringen würde, als mit einer der oben aufgeführten anderen Sprachen.

Aus technischen und auch praktischen Überlegungen wird die Programmiersprache *Java* zum Einsatz kommen, da es für diese Programmiersprache eine Vielzahl von Techniken und Bibliotheken gibt, die sie für einen Einsatz als Web-Applikationsserver prädestinieren. Ein weiterer Vorteil in Bezug auf die oben angesprochenen Kriterien ist, dass *Java*-Programme mit einem normalen Texteditor entwickelt werden können bzw. es sehr leistungsfähige Entwicklungsumgebungen (IDE) auch als freie Software (z. B.*NetBeans*, *Eclipse*) gibt.

5.3.2 Basistechnik

Mit der Entscheidung zu einer Programmiersprache sind einige weitergehenden Entscheidungen zur Betriebssystemumgebung z. B. verbunden. Da hier ein Websystem entwickelt wird, muss entschieden werden, welche weiteren Programme wie bspw. Webserversoftware eingesetzt werden sollen.
Es gibt zwei wesentliche Techniken dynamische Webseiten auf Serverseite zu erzeugen, wenn ein Webserver für statische Inhalte gegeben ist. Die erste Variante `CGI` oder auch `FastCGI` lässt die Seiten durch ein separates Programm außerhalb des Webservers erzeugen während bei der zweiten Variante die Seiten vom Webserver direkt bzw. einer eingebundenen Komponente generiert werden.
Da beide Varianten sehr gebräuchlich sind, hier eine kurze Gegenüberstellung:

- **`CGI` bzw. `FastCGI`**
  Hierbei wird die Webseite durch ein externes Programm generiert. Beim klassischen `CGI` (*Common-Gateway-Interface*) wird dabei für jede Anfrage an der Webserver eine neue Instanz des externen Programms erzeugt und über Standardein- und Ausgabe mit Daten versorgt bzw. die erzeugte

[94] Infos: `http://groovy.codehaus.org/`, Stand 2008-03-02

Webseite weitergeleitet. Dieses Verfahren ist zwar schön einfach (es ist keine weitere Software, außer dem Webserver nötig) aber auch in höchstem Maße ineffizient. Insbesondere die Notwendigkeit, dass für jede eingehende Anfrage an den Webserver eine Instanz des verarbeitenden Programms erzeugt wird, macht dieses Verfahren in größeren Konfigurationen unökonomisch. Insbesondere bei interpretierten Sprachen, wie Java, ist dieses Verfahren sehr unbefriedigend, da für jede ausgelieferte Seite eine Instanz des Interpreters gestartet wird, was unter Umständen zu Verzögerungen führt und Resourcen belegt. Es kommt noch hinzu, dass bestimmte Programmiertechniken wie sie für persistente Daten im Programm (bspw. statische Variablen) nötig sind, hierdurch nicht möglich sind, da ja für jede Seitenabfrage eine neue Instanz des Programms erzeugt wird.
Einen Ausweg bietet `FastCGI`, wo das externe, verarbeitende Programm dauerhaft vorhanden ist und die notwendigen Daten vom Webserver (z. B. Formulareingabewerte, Parameter etc.) über einen Schnittstelle (bspw. Datei oder Netzwerkverbindung) übermittelt werden. Dieses Verfahren wird aufgrund der Komplexität der Anordnung (Konfiguration von Netzwerk, Ein- & Ausgabeparameter, usw.) selten für kleine Projekte verwendet, ist aber bei vielen größeren Internetseiten im Einsatz. Die komplexe Anbindung an den Webserver wird meist durch eine zusätzliche Software übernommen.[95]

- **Webserver-Modul**
  Bei dieser Variante wird auf ein externes Programm verzichtet und die Verarbeitung direkt in den Webserver selbst gelegt. Sehr oft ist die Webserver-Software mit sog. *PlugIns* erweiterbar, um z. B. bestimmte Programmiersprachen (bspw. *PHP* oder *Visual Basic* mit *Microsoft Active Server Pages* (ASP)) zu unterstützen. Dabei wird im Webserverprogramm dann die Seite an die Pluginsoftware übergeben, wenn bestimmte Zeichen oder Codewörter im Quellcode der Webseite vorhanden sind.
  Durch diesen Mechanismus kann die Konfiguration relativ einfach gehalten werden mit allen Vorteilen, die auch bei `FastCGI` gelten. Allerdings soll auch nicht verschwiegen werden, dass diese Variante, je nach Einsatz, möglicherweise Sicherheitsrisiken birgt, da nun ein komplexes Programm im Kontext des Webservers verarbeitet wird, was nicht immer gewünscht ist. Aus diesem Grund wird bei vielen Web-Providern z. B.*PHP* nur als `CGI` Applikation angeboten, obwohl es als PlugIn für die Webserversoftware deutlich effizienter einsetzbar wäre.

Ein Java-Programm lässt nun sowohl durch die klassische `CGI`-Variante mit dem Webserver verbinden, als auch über eine Variante der `FastCGI` Lösung. Da gerade bei Java die `CGI`-Variante nicht empfehlenswert ist, alleine schon wegen der merklichen Verzögerung beim Starten, und eine Lösung als Webserver-Modul nicht existiert (bzw. nicht weiterentwickelt wird), wird hier die `FastCGI` Lösung bevorzugt.
Wie schon oben angedeutet geht dies ohne zusätzliche Software nur mit hohem Aufwand. Daher wird hier eine Software eingesetzt, die sich *Java-Servlet-Container* nennt. Diese Software kümmert sich um die Initialisierung einer Java-Software (Servlet)[96] und die Kommunikation mit dem Webserver. Darüberhinaus können populäre Implementationen (z. B.*Apache-Tomcat*) dieser Software selbst als Webserver dienen.

[95] Einen Überblick zum Einstieg über einsetzbare Software und auch Hintergründe liefert die Webseite `http://www.fastcgi.com/` Einen eher technischen Einblick zu Fähigkeiten, Einsatzmöglichkeiten und Programmierung und Vergleich mit anderen Lösungen bietet [Pet02].

[96] Das Kunstwort *Servlet*, zusammengesetzt aus den Worten *Server* und *Applet*, hat sich eingebürgert für diese Art von Programm. (Vgl. [Wik07c])

Der Einsatz einer solcher Software ist mittlerweile Stand der Technik; es gibt verschiedene Implementation solcher Servlet-Container, sowohl als freie Software, als auch kommerzielle Software. Die Kommunikation mit dem Webserver ist standardisiert als „Apache JServ Protocol“ (AJP) Protokoll mit einem PlugIn für den Apache Webserver verfügbar (sofern man nicht die Container-Software direkt als Webserver verwenden möchte).
Ein Diagramm des technischen Systemaufbaus stellt sich so dar:

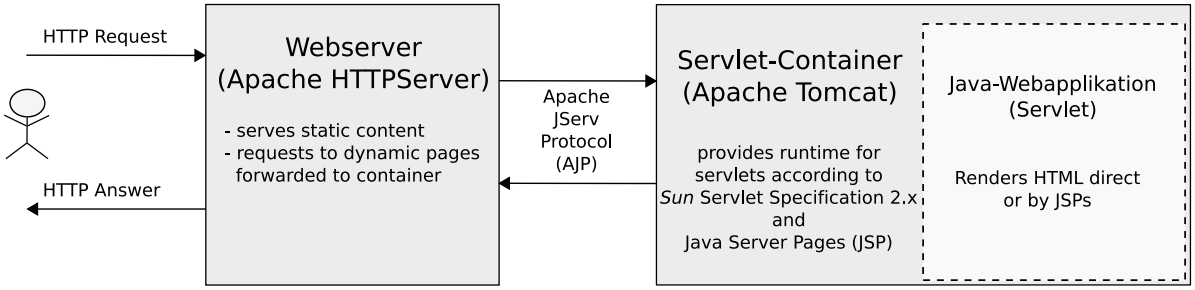


Abbildung 5.1: Einbindung Servlet-Container in System schematisch

In der Grafik sind als Komponenten die entsprechenden Softwareprodukte der *Apache Software Foundation* angegeben, die als freie Software unter vielen Betriebssystemen zur Verfügung stehen. Es sind aber mit dieser oder sehr ähnlicher Konstellation auch Produkte anderer Hersteller einsetzbar und kombinierbar, wie z. B.*IBM Websphere* oder *BEA Weblogic*.
Im weiteren Aufbau sollen aber die frei verfügbaren Softwareprodukte *Apache Tomcat* als Servlet-Container und *Apache HTTPServer* als Webserver verwendet werden.

5.3.3 Auswahl von Programmkomponenten und Rahmenwerk

In den vorherigen Abschnitten wurden Grundtechnologien ausgewählt sowohl für Client als auch für Server. Nun fehlt noch eine Technologie, die diese beiden Teile so verbindet, dass erstens die Rahmenbedingungen aus Kapitel 4.1 erfüllt, als auch die dort aufgestellten Anforderungen erreicht werden.
Eine einfache Entwicklung ist schon mit den bisher ausgewählten Komponenten möglich: Mit einem Java-Servlet und dem Servlet-Container ist es problemlos möglich HTML Seiten dynamisch zu erzeugen und in einem beliebigen Internet Browser darzustellen. Dabei können HTML-Seiten mittels JSP oder mit direkten Ausgaben (vergleichbar `System.out.println`) im Servlet gerendert werden. Diese Art der Softwareentwicklung ist zwar möglich, aber aufwendig und tendiert zu Unübersichtlichkeit. Eine Zusammenfassung dieser und weiterer Punkte fasst [Tho03] gut zusammen.
Zur Verbesserung soll nun, wie auch schon in den Rahmenbedingungen gefordert, ein *Model-View-Controller* (MVC) basiertes Architekturmuster verwendet werden. Dazu ist ein zusätzliches Framework nötig, das diese Entwicklung unterstützt, da mit herkömmlicher Servlet und JSP Programmierung, keine Entwicklung nach diesem Entwurfsmuster möglich ist.[97]
In Abb. 5.2 ist das konzeptionelle Schema dargestellt, wie mit Servlets und JSP-Technologie eine MVC-Architektur aufgebaut werden kann:

[97] Mittels JSP-Technologie ist eine vereinfachte Form des MVC-Modells möglich, von Sun als „Model 1“ bezeichnet. Im engeren Sinne entspricht diese aber nicht der Definition von MVC, daher wird von Sun empfohlen eine erweiterte Variante „Model 2“ zu nutzen, das z. B. mit einem zusätzlichen Framework (bspw. *Struts*) zu realisieren ist (vgl. [Qus03]).

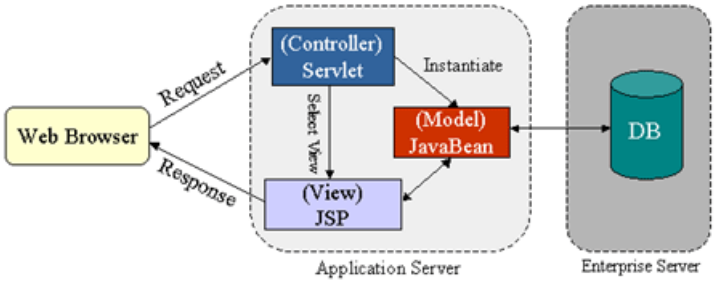


Abbildung 5.2: Model-View-Controller Architektur für Webanwendungen. Quelle: [Qus03]

Um nun die Auswahl einer geeigneten Software zu treffen, bietet es sich an, gängige Frameworks zu untersuchen und ein Geeignetes auszuwählen.
Nun gibt es zur Zeit in diesem Bereich eine geradezu stürmische Entwicklung; Es werden „fast täglich“ neue Toolkits, Frameworks und Bibliotheken entworfen oder weiterentwickelt. Ich werde mich daher hier auf die repräsentative Auswahl von weniger, schon etablierter Software beschränken. Eine Aufstellung dieser und vieler weiterer solcher Frameworks ist unter [AXP07] dargestellt.

Aufstellung möglichen Varianten

- **Java Server Faces – JSF**
  Das *Java Server Faces*-Framework ein von der Firma *Sun* geschaffener Standard für die Realisierung von grafischen Oberflächen mit der Java Enterprise Edition (`J2EE`).
  Mit diesem Framework wird eine abstrakte Sicht der Oberfläche entworfen, die aus wiederverwendbaren Komponenten besteht. Im Prinzip ist das Konzept der Java Server Faces nicht auf Weboberflächen beschränkt. Es können im Prinzip auch andere Darstellungsmethoden gewählt werden, ein charakteristischer Anwendungsfall liegt aber bei der Präsentation von Weboberflächen.

  JSF sieht eine strikte Trennung zwischen Darstellung und Code vor: Die Darstellung kann dabei über JSP (*Java Server Pages*) oder sog. *Facelets* gehen. Bei beiden Varianten werden JSF-Komponenten, wie gekapselte Objekte, in eine JSP Seite eingebettet. Bei Facelets kann dies sogar mit einem Standard-Designprogramm für HTML-Seiten geschehen.
  Der Vorteil dieser Herangehensweise ist die völlige Trennung zwischen Darstellung und Programmlogik. Eine JSF-Komponente kann dabei als unabhängig von anderen Komponenten betrachtet werden, die Zusammenführung mit der Programmlogik geschieht durch die dahinterliegende Kontrolllogikschicht. Hierfür bietet JSF eine API, die es der Kontrolllogik ermöglicht, Ereignisse (z. B. das Drücken eines Buttons) zu verarbeiten und an die Modellklassen in Form von sog. „Beans“ weiterzuleiten.

  Die Java Server Faces bieten einerseits eine sehr strukturierte, leistungsfähige und vielfach verwendbare, aber auf der anderen Seite auch eine recht formalisierte, festgelegte Herangehensweise an grafische Oberflächen für Serversysteme an. Dies kann zum einen sehr positiv sein, wenn z. B. Komponenten in einem Programmsystem tatsächlich vielfach verwendet werden können, bedeutet aber gleichzeitig auch einen erhöhten Aufwand zur Entwicklung solcher Komponenten. Diesem Fakt entsprechend bieten mittlerweile mehrere Firmen und Organisationen solche Komponenten als fertige Software an. Dabei geht die Auswahl von einfachen Formularfeldern bis hin zu komplexen AJAX-Komponenten mit optisch aufwendigen JavaScript Funktionen oder auch

Komponenten, die weitere Programmlogik gleich mitbringen (z. B. Oracle ADF Faces).
Alles in allem ist die Verwendung von JSF sehr komplex und bedingt außerdem die Verwendung der Java Enterprise Edition, deren Einbindung (und durchgehende Verwendung) weitere Konsequenzen für die Programmentwicklung und weitere Softwarekomponenten (einen J2EE-Applicationserver wie z. B. *JBoss* (Open Source) oder *Glassfish* (Referenzimplementation von Sun) benötigt.

- **Apache Tapestry**[98]
  Apache Tapestry ist den Java Server Faces ähnlich, konzentriert sich allerdings ausschließlich auf Weboberflächen. Wie bei den Java Server Faces bestehen die Webseiten aus Komponenten, die eingebunden werden. Hier geschieht dies über HTML-Seiten, die als Template verwendet werden. Dabei können Standard HTML-Tags verwendet werden, wobei zusätzliche Attribute vergeben werden, um die Verbindung zum Framework herzustellen. Vorteilhaft an dieser Stelle ist, dass diese HTML-Templates mit Standard-Webeditoren erstellt und bearbeitet werden können.
  Für das Framework ist nun jede Webseite zunächst erst mal auch eine Java-Klasse mit Get-/Set-Methoden für den Zugriff auf Felder sowie Eventlistener-Methoden. Von der MVC-Klassifikation aus gesehen, stellen die HTML-Seiten / Templates einen *View* dar und die zugehörigen Java-Klassen dienen als *Controller*. Durch diese Trennung kann nun sehr einfach auf Daten (Formulardaten, Metadaten) der Seite über Instanzmethoden zugegriffen werden und die Verarbeitung angestoßen werden, ohne im Java-Code mit dem HTML selbst in Berührung zu kommen.

  Apache Tapestry hat sehr viele gute Ansätze, die zum Teil mit „klassischer“ Herangehensweise an Webprogrammierung nicht mehr viel gemein haben. Die Sicht einer Webseite als eine Java-Klasse ist im Vergleich zu anderen Frameworks sehr innovativ und wegweisend. Allerdings kann die Entwicklung mit Tapestry auch sehr komplex werden, wenn z. B. neue *View*-Komponenten entwickelt werden müssen. Von Hause aus bringt Tapestry nur einfache Standardelemente mit, was sich aber bald ändern könnte.[99]

- **Apache Struts**[100]
  Struts ist zusammen mit Java Server Faces das wohl bekannteste Framework für Webprogrammierung und eins der am meisten eingesetzten Frameworks.
  Die Philosophie hinter Struts ist eng an das MVC-Modell angelehnt, bei *Sun* für Webprogrammierung als „MVC-Model 2“ bezeichnet. Für die Darstellung *View*) wird auf JSP gesetzt, jeder Zugriff auf einen solchen *View* wird über eine Java 'Action'-Klasse abgewickelt, die als *Controller* fungiert. Daten und weitere Programmlogik sind über Java-Beans zugänglich, die durch das Framework z. B. automatisch erzeugt bzw. einzelnen Actions zugeordnet werden können. Actions, Beans und JSPs werden über eine zentrale Konfigurationsdatei im XML-Format miteinander verknüpft.
  Weitere wichtige Fähigkeiten sind die Unterstützung von Internationalisierung und eine Technik, 'Tiling' genannt, die es erlaubt eine Webseite aus mehreren Unterseiten, sog. 'Tiles' zusammenzusetzen.

  Struts ist sehr gut dokumentiert, es sind diverse Bücher verfügbar: von „Struts for Dummies“ über die üblichen Referenzsammlungen bis zu kompletten Falldokumentationen. Daneben exis-

[98] http://tapestry.apache.org/
[99] Dies soll sich, laut Ankündigungen auf der Homepage, in der nächsten Version sehr verbessern, insbesondere, was AJAX-Support angeht. Allerdings ist zum Zeitpunkt dieses Schreibens dieses erst als sog. Preview-Release erhältlich.
[100] http://struts.apache.org/

tiert eine, wohl auch wegen der großen Verbreitung, aktive Gemeinschaft mit Diskussionsforen, Webseiten und Mailinglisten. Dies kann als ein großer Vorteil dieses Frameworks gesehen werden. Daneben besticht die relative Einfachheit, im Vergleich zu den anderen Frameworks, mit der die ersten Seiten erstellt werden können.

Struts ist zurzeit in zwei Varianten zu haben: Version 1 und Version 2. Die Version 1 hat eine recht lange Entwicklung hinter sich, wird aber nur noch begrenzt weiterentwickelt und ist vielfach eingesetzt.
Version 2 ist noch sehr neu und ist aus der Migration von Struts 1 und einem von Struts 1 im frühen Stadium abgeleiteten Framework *WebWork* entstanden, das zu einiger Verbreitung gelangt ist. Daher ist Struts 2 auch nicht API-kompatibel zu Version 1, die Gemeinsamkeiten liegen eher im Konzeptionellen.
Mit der Version 2 kamen erweiterte *JavaScript* Funktionen hinzu, um AJAX-Techniken zu realisieren oder für optische Effekte zu ermöglichen. Außerdem wurden einige strukturelle Mängel der Version 1 beseitigt und die Entwicklung mit dem Framework vereinfacht. Dazu gehört auch die Fähigkeit andere Technologien als `JSP` für die Darstellung einsetzen zu können.

Neben den aufgezählten Frameworks gibt es natürlich noch viele weitere und auch einige, die sich nur um einen Teil der MVC-Komponenten kümmern. Erwähnenswert ist hier z. B. das Google Web Toolkit (`GWT`)[101], als Oberflächenkomponente, dass Oberflächen mit einem Java-zu-JavaScript Compiler aufbaut. Dieses Konzept ist sehr vielversprechend, allerdings zunächst auch beschränkt auf die mitgelieferten GUI-Elemente. Nichtsdestotrotz ist GWT mittlerweile zu einiger Verbreitung gelangt.
Eine andere Variante ist *Spring*[102] mit einem umfassenden Konzept für Business- und Programmlogik sowie des *Spring MVC*-Moduls für Webanwendungen. Dabei greift *Spring* sehr tief in die Entwicklung ein um von bekannten APIs zu abstrahieren und zielt eher auf J2EE-Anwendungen (vgl. [Hö04]).

Auswahl einer Variante

Für das hier zu entwickelnde System kommen im Prinzip alle aufgezeigten Frameworks in Frage. Mit *Apache Struts* oder *Apache Tapestry* kann man die sehr vielfältigen Angebote der Apache Software Foundation nutzen. *JSF* ist als Referenzimplementation von *Sun* erhältlich und passt sich in die Java-Klassenstruktur ein. Ein Framework wie *JSF* ist zwar sehr leistungsfähig, benötigt allerdings auch entsprechende Tools und Umgebung (wie z. B. die *Java Enterprise Edition*). Grundsätzlich ist der Ansatz von *JSF* und *Apache Tapestry* für Webprogrammierung ein eher Abstrakter in Bezug auf eine einzelne Webseite als *Struts*. Einen Vergleich zwischen *Apache Struts* und *JSF* bietet [Sch05], anhand eines Beispiels und detaillierter Erklärung der verschiedenen Konzepte.
Eine weitere Möglichkeit würde die Verwendung des *Google Web Toolkits* bieten, dessen innovativer Ansatz viele Möglichkeiten eröffnet. Allerdings ist das Google Web Toolkit mit Erweiterungen nicht einfach zu bedienen (aufgrund des Konzeptes eines Java-zu-JavaScript Compilers) und bedenkenswert sind auch Datenschutz- und Lizenzfragen (vgl. [LMS07]).

Auch um auf eine größere Auswahl an Dokumentation und Hilfestellung zurückgreifen zu können, fiel die Wahl auf Struts Version 2. Da mit Struts eine komplexe MVC-Umgebung inklusive Unterstützung für Internationalisierung und AJAX zur Verfügung steht, ist für das Kerngebiet der Entwicklung keine weitere Bibliothek notwendig.

[101] http://code.google.com/webtoolkit/
[102] http://www.springframework.org/

5.3.4 Evaluation und Auswahl von weiteren Komponenten

Für das Gesamtsystem wurden noch weitere Komponenten ausgewählt:

- **Datenbank**
  Theoretisch kann fast jede „handelsübliche“ relationale Datenbank verwendet werden. Aufgrund guter Leistung, freier Verfügbarkeit und vorhandener Einbindung in das Java Datenbanktreibermodell (JDBC) wurde eine MySQL-Datenbank[103] zum Einsatz ausgewählt.
  Durch Einsatz der weitgehend standardisierten *Structured Query Language* (SQL) sind Varianten verschiedener Hersteller einsetzbar; für die hier entwickelte Applikation waren die Fähigkeiten des ausgewählten Datenbanksystems mehr als ausreichend.

- **Suchmaschine**
  Die initiale Auswahl von relevanter Literatur geschieht durch eine Suchfunktion im Programm. Diese Funktion könnte durch eine interne Suchfunktion implementiert werden, was aber aufgrund der wenigen verfügbaren Daten (es sind außer dem Titel meist nur Ausschnitte aus dem Vorwort bzw. das Abstract eines Dokumentes in der lokalen Datenbasis vorhanden) möglicherweise nicht sehr effizient wäre.
  Daher wurde diese Funktion durch die Anbindung einer externen Suchmaschine gelöst. Diese Lösung ist zwar nicht ideal, da sie erhebliche Geschwindigkeitseinbußen bzw. Verzögerungen bei der Benutzung des Programms einbringt, allerdings konnte dadurch der Aufwand für die Suche nach Dokumenten minimiert werden. Bei einer möglichen Weiterentwicklung des Systems, sollte hier evtl. doch auf eine interne Volltextsuchmaschine[104] zurückgegriffen werden.

  Hier wurde nun die Suche nach Dokumenten mittels der Suchmaschine *Yahoo*[105] implementiert, da diese eine relativ einfache und unkomplizierte Programmschnittstelle bereithält, mit der sich Suchergebnisse ermitteln lassen. Diese Auswahl eines Suchergebnis-Anbieters ist relativ problemlos austauschbar.[106]

- **Clusterengine**
  Wie schon in Kapitel 3.1.2 beschrieben, kam die Software *CLUTO*[107] in Form einer dynamisch ladbaren Bibliothek zum Einsatz. Die Integration in das Java-System wurde mit Hilfe von *JCluto*[108] realisiert, was allerdings diverse Anpassungen erforderte.

- **PDF-Export**
  Eine PDF-Exportfunktion lässt sich unter Java mit vielen verschiedenen Mitteln realisieren. Eine recht weit verbreitete Standardbibliothek dafür stellt *iText*[109] dar, mit der sich auch Rich Text Format (RTF) Dokumente erstellen lassen. Daher wurde diese Bibliothek eingebunden.

- **Lineare Algebra**
  Für die LSA Operation (siehe Kapitel 3.1.1.2) werden optimierte mathematische Operationen aus der linearen Algebra benötigt. Der „de facto“ Standard in der numerischen Mathematik stellt

[103] http://www.mysql.com, Stand 2008-02-08
[104] Der Aufbau eines solchen Systems ist z. B. mit dem *Apache Lucene* Toolkit relativ problemlos möglich.
[105] http://developer.yahoo.com/, Stand 2007-11-30
[106] Ideal wäre es sicher gewesen, direkt über *CiteSeer* zu suchen, da mit der dortigen Suche subjektiv bessere Ergebnisse ermittelt werden konnten. Leider bietet *CiteSeer* keine adäquate Schnittstelle und aufgrund teilweise extrem langer Ladezeiten schien eine Umwandlung der Suchergebnisse direkt von der Webseite nicht praktikabel.
[107] http://glaros.dtc.umn.edu/gkhome/cluto/cluto/overview, Stand 2008-02-08
[108] Teil des *TableView* Programms unter: http://www.ccgb.umn.edu/software/java/, Stand 2008-02-08
[109] http://www.lowagie.com/iText/, Stand 2008-02-08

dafür die Bibliothek *Lapack* dar. Eine in Java implementierte Variante davon ist das *Matrix Toolkits for Java*[110]. Der Vorteil dieser Implementierung ist, dass sowohl eine in „purem“ Java geschriebene Version verwendet werden kann, als auch automatisch eine für die konkrete Maschine optimierte Variante benutzt wird.

5.4 Zusammenfassung

Zusammenfassung der auswählten Technologien und Techniken in Kurzform:

Technologie	Verwendung
Programmiersprache: **Java**	Als verwendete Programmiersprache für das Serversystem wurde Java gewählt.
Implementation als **Servlet**	Das Serversystem wurde als *Java Servlet* implementiert. Dazu wurde ein Java Servlet Container verwendet und installiert.
Darstellung auf Client mit Standard-**HTML** und **JavaScript**	Zur Darstellung der Inhalte auf dem Client wurde bestimmt, das, soweit möglich, Standardtechniken verwendet werden, die ohne zusätzliche Software auskommen. Dazu kann i.a. 'normales' HTML und JavaScript verwendet werden.
Servlet-Entwicklung mit **Struts 2**	Als MVC-Bibliothek (*Model-View-Controller*) kam die Software *Struts 2* zum Einsatz.
Datenbankserver **MySQL**	Als Datenbankserver kam eine *MySQL*-Instanz zum Einsatz. Im Prinzip ist aber jeder SQL-Datenbankserver einsetzbar, für den ein Java-Datenbanktreiber existiert.
Externe Suchmaschine: **Yahoo**	Die initiale Suche nach relevanten Dokumenten basiert auf einer (Voll-) Textsuchmaschine. Diese wurde durch die Anbindung an den Webservice von *Yahoo* integriert.
Clusterengine: **CLUTO**	Eine schnelle Clusterbildung ist für eine gute Benutzbarkeit des Programms wichtig. Daher wurde hier auf eine optimierte Bibliothek zurückgegriffen.
Lineare Algebra: **Matrix Tookits for Java**	Eine optimierte Variante für lineare Algebra (siehe Kap. 3.1.1.2)

Tabelle 5.1: Zusammenfassung: Verwendete Technologie und Techniken

[110] http://ressim.berlios.de/, Stand 2008-02-08

5.5 Daten

Aus den in Kapitel 2 vorgestellten Daten wird eine Datenbasis für das Projekt ausgewählt. Dabei werden im Abschnitt 5.5.1 nochmal die üblichen Probleme dargelegt, die bei Zitationsdatenbanken (und ihrer Analyse) auftreten.
Im Anschluss folgt die konkrete Auswahl und kurze Beschreibung der für den Nutzbarmachung notwendigen Schritte. Eine Evaluation eines Testdatensatzes mit den in Kapitel 3 vorgestellten Methoden folgt am Ende des Kapitels.

5.5.1 Ausgangsdaten und Vorbereitung

Für die effektive Anwendung der in Kapitel 3 vorgestellten Verfahren sind Ausgangsdaten nötig, die sowohl allgemeine Metadaten (Autorennamen, Titel, Publikationsjahr, Verlag und weitere Angaben je nach Dokumententyp: Seitenzahlen, Zeitschriftenname, Zeitschriftenjahrgang und Nummer etc.) enthalten, als auch den textuellen Inhalt. Anstatt des kompletten textuellen Inhalts kann auch nur das Abstract bzw. die Zusammenfassung enthalten sein. Weiterhin sollten zu Dokumenten auch die Quelle bekannt sein (also eine Adresse (Verlag, URL), bei der das komplette Dokument verfügbar ist). Um Medienbrüche zu vermeiden sind Internetadressen zu bevorzugen.
Für die Anwendung von Zitationsanalysen, ist eine verarbeitete und zu anderen Dokumenten eindeutig zugeordnete Form der Referenzen im Dokument nötig.

Probleme in Literaturdatenbanken

In Kapitel 2 bzw. Abschnitt 2.2.2 wurden verschiedene Literaturdatenbanken vorgestellt, die potentiell die für dieses System interessante Daten bereithalten.
Allerdings sind in diesen Daten verschiedenste Fehler enthalten, so dass die tatsächliche anzutreffenden Qualität der Daten, insbesondere von frei verfügbaren Sammlungen, stark schwankend ist.
Übliche Mängel sind zum Beispiel:

- **Fehler in Metadaten:**
  Fehler bei der Metadatenerkennung wie falsche Autorennamen, Adressen, Organisationen, Titel oder Schlagwörtern.

- **Duplikate:**
  Oft sind Dateien wissenschaftlicher Publikationen nicht nur an einer Stelle im Internet abgelegt, sondern an verschiedenen Stellen, teilweise in unterschiedlichen Versionen.
  Daher muss ein Datenbankdienst mit wissenschaftlicher Literatur, der auch selbstständig im Internet sucht, zuverlässig Duplikate erkennen und aussortieren. In der Praxis sind Duplikate in den meisten wissenschaftlichen Datenbanken anzutreffen, allerdings mit unterschiedlicher Häufigkeit.
  Duplikate haben auch noch einen anderen negativen Effekt, es wird durch sie der Zitationsgraph (siehe Kapitel 3.1.1.1) verändert, da Zitationen möglicherweise nicht richtig zugeordnet werden können oder gar verdoppelt werden.

- **Unvollständige Referenzen:**
  Nicht alle Zitationen eines Dokumentes werden extrahiert. Dies kann z. B. aufgrund Fehlern in der Erkennung von Referenzen im Quelldokument vorkommen. Dies hat Auswirkung auf den Zitationsgraphen, da dadurch Informationen fehlen und die Ähnlichkeit zweier Dokumente möglicherweise niedriger berechnet wird.

- **Falsch zugeordnete Referenzen:**
  Einem Dokument **a** wird fälschlicherweise ein Dokument **c** als Referenz / Zitation zugeordnet, obwohl Dokument **b** richtig gewesen wäre.
  Dieser Fehler ist anhand der „fertigen“ Datenbank nur schwer nachzuweisen und wird daher hier als klein bzw. nicht relevant angenommen.

Es gibt im Weiteren noch systembedingte Fehler, die hier der Vollständigkeit halber aufgeführt werden:

- **Datenerfassung:**
  Grundlage der erfassten Daten ist in den meisten Fällen die Verfügbarkeit im Internet oder die Publikation bei ausgewählten Verlagen. D. h. nur ein potentiell kleiner Teil von verfügbaren Dokumenten konnte in einer Datenbank aufgenommen werden.
  Daher fehlen möglicherweise wichtige Quellen wie Periodika und insbesondere Monographien.

- **Motivation einer Zitation:**
  Die Zitation einer Quelle muss, im Gegensatz zu den Annahmen in Abschnitt 3.1.1.1, nicht unbedingt auf einer wissenschaftlichen oder inhaltlichen Grundlage beruhen. Manchmal werden Quellen auch aufgrund einer anderweitigen Zusammenarbeit oder aus strategischen Gründen, zum Beispiel um die Zitationshäufigkeit einer bestimmten Quelle zu erhöhen, angegeben. Eine andere Möglichkeit ist, dass besonders populäre Quellen oder Arbeiten prominenter Personen eher angegeben werden. Dieses Verhalten ist als Matthäus-Effekt[111] der Bibliometrie bekannt.

- **Selbstzitationen:**
  Die Möglichkeit eines Wissenschaftlers eine eigene, früher veröffentlichte, Arbeit zu zitieren wird traditionellerweise gerne genutzt, ist nach Meinung einiger Wissenschaftler aber als „unetisch“, teilweise im Randbereich des Plagiatismus, anzusehen.
  Dies gilt insbesondere, wenn Wissenschaftler eine Vielzahl von einander zitierenden Arbeiten veröffentlichen, möglicherweise motiviert durch das Ziel die Gesamtanzahl der Veröffentlichungen und Zitationen zu erhöhen (vgl. [EG08]).[112]

Die meisten dieser Fehler und Probleme sind in allen Datenbanken in unterschiedlicher Ausprägung vorhanden. Es sollten aber bei der Bewertung einer Datenbank und insbesondere der durch bibliometrische Verfahren gewonnenen Daten diese Fakten bekannt sein.

5.5.2 Wahl der Datenquelle

Um eine der oben vorgestellten Literaturdatenbanken zu nutzen, ist entweder eine nicht-interaktive, elektronisch nutzbare Schnittstelle nötig (zum Suchen von Dokumenten und Analysieren von Zitationen) oder ein Datenbankauszug kann lokal im zu entwickelnden System genutzt werden. Letzteres erlaubt im Allgemeinen bessere Reaktionszeiten bei der Verwendung in interaktiven Systemen.
Da keines der oben vorgestellten Systeme eine solche Schnittstelle im ausreichenden Umfang bereitstellt, wurde auf die öffentlich verfügbare Datenquelle der *CiteSeer*-Datenbank zurückgegriffen. Der Hauptgrund für diese Wahl war die einfache Verfügbarkeit und die Tatsache, dass die Daten aus einem „echten“ funktionierenden und evaluierten System stammen.
Allerdings weist diese Datenbank alle oben aufgeführten Fehler und Problem auf, insbesondere sind

[111] Benannt nach einem biblischen Gleichnis im Evangelium des Matthäus und kann umgangssprachlich als „Wer hat, dem wird gegeben.“ beschrieben werden. (Vgl. [Wik08a])

[112] Eine Onlinedatenbank mit betreffenden Artikeln aus der MEDLINE Datenbank findet sich unter: `http://discovery.swmed.edu/dejavu/`, Stand 2008-03-02

viele Duplikate und fehlerhafte Metainformationen enthalten. An Daten zu einem Dokument stehen ansonsten die üblichen Metadaten bereit und das Abstract des Dokuments (sehr oft mit dem Einleitungstext und/oder Inhaltsverzeichnis kombiniert) zur Verfügung. Pro Dokument sind nur die unmittelbar in der Datenbank vorhandenen Dokumente referenziert, außerhalb dieser Menge liegende Dokumente sind nicht verzeichnet.[113]

Aus der *CiteSeer*-Datenbasis wurden für die Nutzung in diesem System 716 772 Dokumenteneinträge mit Referenzierungen (aus XML-Dateien) übernommen, sowie zusätzliche 10 773 Dokumente ohne Referenzierungen (aus BIBTEX Dateien). Es waren keine weiteren expliziten Informationen zu den Dokumententypen enthalten (also ob Artikel, Proceeding, Buch etc.), die Sprache war immer mit `en` (Englisch) angegeben, was teilweise inkorrekt war und für gut 4% aller Dokumente war kein gültiges Publikationsdatum enthalten.
Im Zuge des Imports wurden die textuellen Anteile der Daten in die Zeichenkodierung `UTF-8` übertragen, um verschiedene Sonderzeichen zu ermöglichen, die in den Originaldateien z. T. als HTML-Sonderzeichen gespeichert waren. Eine Übersicht über den Prozess gibt Abb. 5.4, die Datenhaltung innerhalb der Applikation wird in Kapitel 6.1.1 beschrieben.

Dokumente	727 545
Referenzen zwischen Dokumenten	1 751 492
Ø Referenzen / Dokument	2,4[114]
Sprache der Dokumente	Englisch[115]
Zeitlicher Umfang	Schwerpunkt 1994–2004 (auch siehe Abb. 5.3)

Tabelle 5.2: Kenndaten des *CiteSeer*-Datensatzes

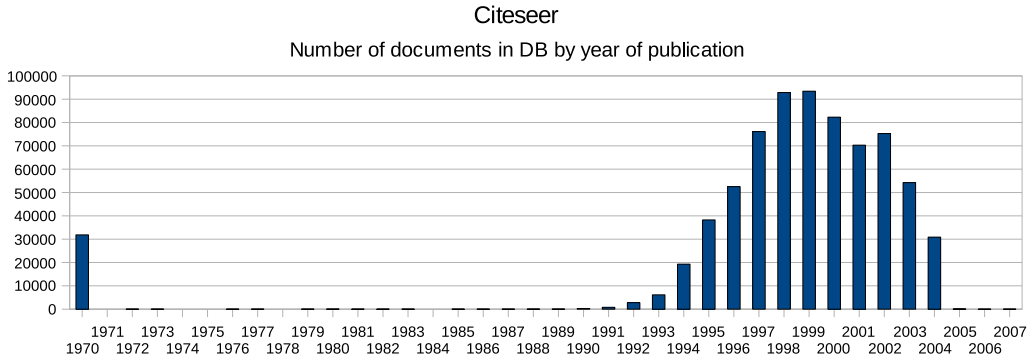


Abbildung 5.3: Verteilung der Dokumente nach Publikationsjahr in importierter *CiteSeer*-Datenbank
Fehlende bzw. nicht extrahierbare Angaben wurden als 1970 eingeordnet.

[113] Allerdings sind solche Dokumente teilweise über das Webinterface sichtbar, was zu dem Schluss führt, dass neben der veröffentlichten Datenbank noch eine weitere Datenbank existieren muss.
[114] An diesem Wert lässt sich schon absehen, dass die CiteSeer-Daten relativ unvollständig sind.
[115] Einige, sehr wenige, Dokumente sind (nicht gekennzeichnet) in anderen Sprachen abgefasst.

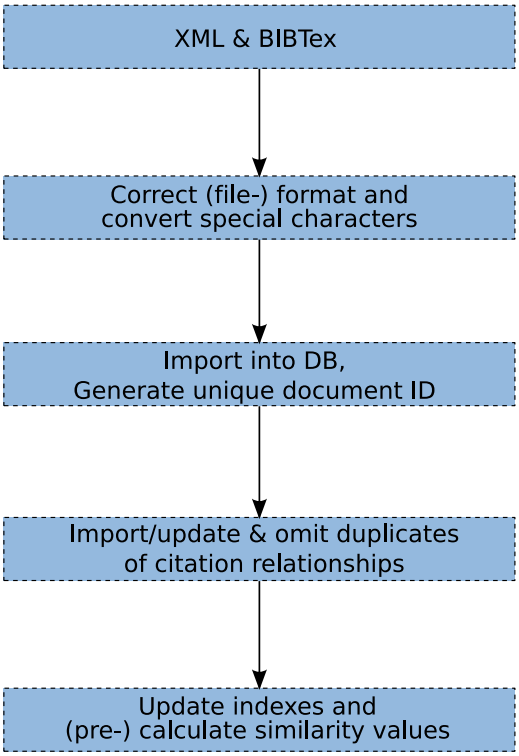


Abbildung 5.4: Import der Daten aus *CiteSeer*-Datensatz in Datenbank und Preprocessing.

6 Struktur und Implementation

Hier wird das fertige Programm in einigen Details sowie der Benutzung vorgestellt, dabei wird zunächst auf grundlegende Datenstrukturen eingegangen. Eine Beschreibung der Programmstruktur folgt und anschließend die Erläuterung der Programmhauptfunktionen.

Die in Kapitel 3.1.1 und 5.5.1 vorgestellten Daten und Verfahren wurden in der Java-Webapplikation *CiteseerCluster* implementiert. Dabei wurden die in den Kapitel 4 erläuterten Anforderungen umgesetzt und mit den in Kapitel 5 vorgestellten Techniken implementiert.

Eine detaillierte Darstellung der Arbeitsweise der internen Algorithmen anhand eines Beispieldatensatzes ist in Abschnitt 6.3 aufgeführt.

6.1 Datenstrukturen

In *CiteseerCluster* wurden die Datenstrukturen für Dokumente, Dokumentengruppen und Dokumentensammlungen nach dem in Abbildung 6.2 dargestellten *Unified Modelling Language* (UML) Schema implementiert. Zu beachten ist, dass abgebildete Schema nur eine grobe Struktur darstellt. Einige Attribute der Klasse `Dokument` (zusätzliche Informationen zu einem Dokument, wie z. B. Seitenangaben, Journaltitel, usw.) wurden ausgelassen. Die Struktur lässt sich dabei wie folgt beschreiben:

- Ein Dokument (`Document`) hat die üblichen Attribute, wie Autor(en), Titel, Abstract, Seitenangaben, URL, Typ, Thema, ISBN, ISSN, etc.
  Zusätzlich hat ein Dokument ein nutzerspezifisches „Label“ und eine ebenfalls nutzerspezifische numerische Bewertung.

- Dokumente können als das Ergebnis einer Suche (`SearchResultEntry`) erzeugt werden, dort sind dann noch ein weitere Informationen verfügbar wie ein Kurztext, eine weitere URL (Dokument auf der Suchseite) sowie Titel. Suchergebnisse haben außerdem einen sog. *Namespace* zugeordnet, der die Suchdomain (z. B. `CiteSeer`) angibt.
  Grundsätzlich ist es möglich, dass Dokumente auch aus anderen Quellen stammen (z. B. von Nutzern manuell eingegeben), was hier aber noch nicht verwendet wurde.

- Ein Autor hat außer einem Vor- und Zunamen auch eine Organisation und möglicherweise ist eine Webadresse (URL) für diese Person bekannt. Dazu ist zu bemerken, dass insbesondere Daten über Organisationen im *CiteSeer* Datensatz nur sehr begrenzt und oft fehlerhaft vorhanden sind.

- Dokumente können in Gruppen (`SubCollection`) einsortiert sein. Eine solche Gruppe hat dann auch einen Namen und Beschreibung. Zusätzlich können Stichwörter für eine Gruppe festgelegt werden.

- Alle Gruppen von Dokumenten sind in einer Dokumentensammlung (`DocumentCollection`) zusammengefasst. Eine solche `DocumentCollection` repräsentiert damit den, möglicherweise bearbeiteten, Stand einer Suche nach Dokumenten. In einer solcher Sammlung werden

erfasst: Name (Identifier), Beschreibung, Stichworte sowie Zeitstempel der Erzeugung und letzten Änderung.
Für die interne Zuordnung wird außerdem eine eindeutige ID-Nummer generiert.

- Ein Benutzer kann mehrere solcher Dokumentensammlungen verwalten (`DocumentRepositoryBean`). Dort werden Informationen über das letzte Speichern festgehalten und Ein-/Ausgaberoutinen sind für das Speichern und Laden aller untergeordneten Objekte zuständig. Die `DocumentCollection` sind dabei unabhängig voneinander und repräsentieren ausschließlich verschiedene Such- und Bearbeitungsvorgänge im System.
- Jeder der Klassen hat für Ein- und Ausgabe entsprechende Methoden, die ein XML-basiertes Dateiformat (bzw. die für diese Klasse relevanten Teile) lesen bzw. schreiben können.
  Eine Serialisierung der Klassen wird durch Implementierung des Interfaces `Serializable` unterstützt. (Dies erlaubt das Beenden der Servlet-Engine ohne Datenverlust.)
- Das zusätzliche Interface `TreeViewNodeInterface` erlaubt eine vereinfachte, Baum-orientiert Sichtweise auf die Dokumentenmenge. Diese wird durch die Weboberfläche visualisiert.

Die Ein- und Ausgabe der Daten (zum Speichern und Laden von Ergebnissen) geschieht durch XML-basierte Dateien. Diese können in der Webanwendung durch einen Benutzer generiert (und dann lokal gespeichert) werden bzw. wieder eingelesen werden. In der entsprechenden XML-Datei sind alle Daten, die in der Klassenstruktur gespeichert sind enthalten. Ein `XSL`-basiertes Stylesheet (*eXtensible Stylesheet Language*) erlaubt außerdem das komfortable Ansehen (Abb. 6.1) des gespeicherten Ergebnisses außerhalb der Applikation. Um die Datenintegrität bzw. die Unversehrtheit des Dateiformats zu überprüfen, wird beim Einlesen die vom Nutzer übergebene XML-Datei gegen eine hinterlegte Dokumententypdefinition (DTD) geprüft. (Siehe Anhang B)

6.1.1 Datenbank

Alle Daten der Dokumente sind in einer MySQL-Datenbank gespeichert und werden durch eine eindeutige ID-Nummer ebenso wie Autoren und auch Organisationen identifiziert. Dabei sind Verknüpfungen zwischen diesen Daten in weiteren Tabellen, wie auch die Zitationsbeziehungen abgelegt. Das Datenbankschema ist in Abbildung 6.3 abgebildet.
Diese Tabellen sind nicht auf eine bestimmte Datenquelle ausgelegt, bspw. wird durch die Tabelle `citeseer_t` die Verknüpfung der gespeicherten Daten mit den ID-Nummer bzw. Dokumentennamen aus der *CiteSeer*-Datenbank hergestellt. Mit diesem Verfahren ist die implementierte Datenbank nicht abhängig von einer bestimmten Datenquelle.

Die Daten einer anderen Literaturdatenbank könnten ohne weiteres importiert werden mit dem Zusatz einer neuen Tabelle zur Übersetzung der hier generierten Dokumenten-IDs zu den ursprünglichen (in der Ausgangsdatenbank) verwendeten Dokumentenbezeichnern.
Um eine weitere Literaturdatenbank zu importieren ist allerdings ein Verfahren zur Duplikaterkennung nötig, damit Dokumente nicht mehrfach gespeichert werden.

Innerhalb der Applikation wurde die Anbindung an die Datenbank durch eine zentrale Klasse `DatabaseStatements` geregelt, die vorbereitete SQL-Befehle und Abfragen aus einer Datei liest und daraus `PreparedStatement`-Objekte generiert. Eine weitere Klasse `DatabaseManager`, die als Singleton implementiert ist, hält einen Pool von Verbindungen zur Datenbank aufrecht, die bei Bedarf verwendet werden.

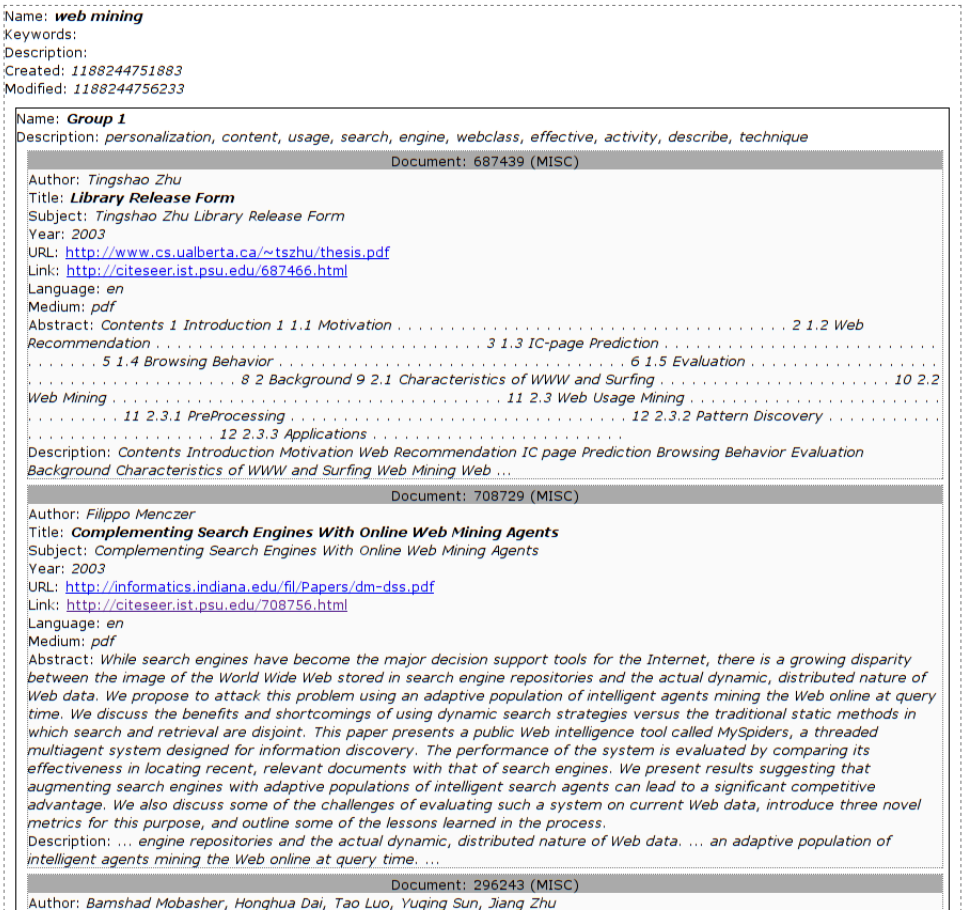


Abbildung 6.1: Offline-Ansicht der XML-Exportdatei mittels XSL-Stylesheet in HTML gewandelt im Webbrowser.

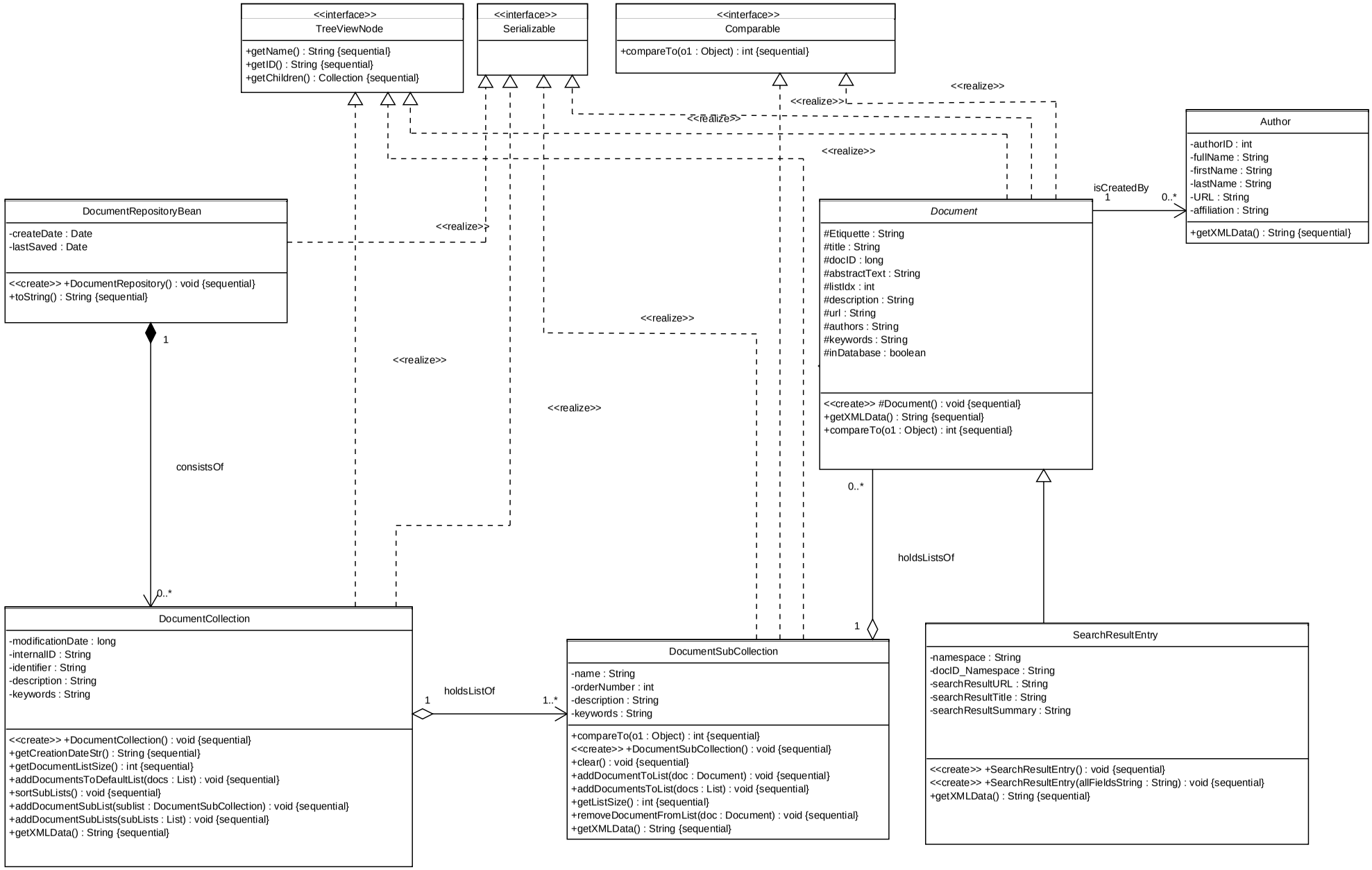


Abbildung 6.2: Datenstruktur des Programms als UML-Klassendiagramm

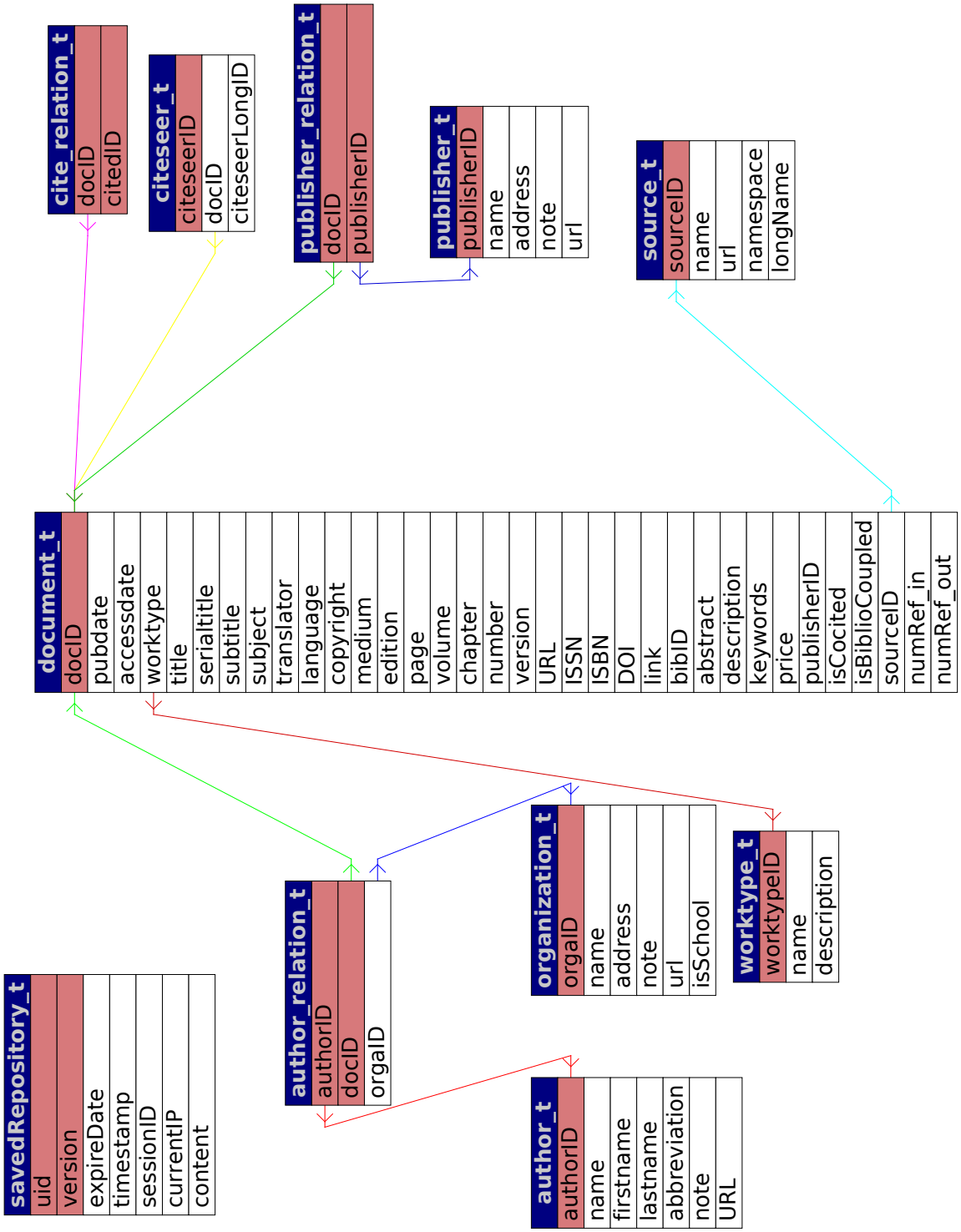


Abbildung 6.3: Datenbankschema

6.1.2 Klassenstruktur

Das Programm selbst ist strukturell in mehrere Pakete unterteilt unter der Hierarchie `de.defaultdomain.citeseer`:

- *Grundebene*: Action-Klassen des *Struts*-Frameworks.
- `base`: Datenbankanbindung, Yahoo-Suche, allg. Konstanten und Einstellungen, Hilfsklassen und -methoden.
- `document`: Alle Klassen (wie oben beschrieben), die zum Dokumenten-Datenmodell gehören.
- `algorithm`: Alle konzeptionellen Algorithmen und Verfahren, sind hier in mehreren Klassen implementiert. Dabei sind die meisten Algorithmen in statischen Methoden implementiert, um eine einfachere Austauschbarkeit zu ermöglichen.
- `gui`: Initialisieren des Servlets, Datenklasse (Bean) für Einstellungen.
- `evaluation`: Test- und Debuggingfunktionen.

6.2 Benutzerdialoge

Das Programm stellt sich organisatorisch in zwei Teilen dar: Die Suche nach Dokumenten und die Organisation von Dokumenten.
Der Grund für die im Programm so deutliche Teilung war, dass ein Benutzer zunächst die Gelegenheit haben sollte eine für ihn/sie passende Dokumentensammlung „zusammenzusuchen“ und dann erst weitere Bearbeitungsschritte mit dieser Dokumentensammlung zu machen. Der Einstieg (bzw. die Startseite) (Abb. 6.4) des Programms ist so gestaltet, dass eine Suche nach Dokumenten „nahegelegt“ wird und sich damit ein Wiedererkennungseffekt zu klassischen Suchmaschinen ergibt.

6.2.1 Suche nach Dokumenten

Die Suche nach Dokumenten ist in den Grundzügen so gestaltet, wie es von bekannten Internetsuchmaschinen bekannt ist. Ein Unterschied besteht in der Art, dass die folgenden Suchergebnisse (äquivalent in etwas zu der nächsten Ergebnisseite bei einer herkömmlichen Suchmaschine) auf Karteireitern (Abb. 6.5) dargestellt werden. Eine zusätzliche Möglichkeit besteht darin, Suchergebnisse aus verschiedenen Suchanfragen zu kombinieren. Die Anordnung auf Karteireitern soll dabei die Übersichtlichkeit bewahren.
Nachdem ein Benutzer erfolgreich eine Liste von Dokumenten zu einem oder mehreren Suchbegriffen erstellt hat, lässt sich diese Liste mit einem Name, Stichworten und Beschreibung im System speichern (Abb. 6.6).

6.2.2 Organisation von Dokumenten

Die in der Suche gespeicherten Listen von Dokumenten sind als sog. Dokumenten-Kollektion in einem Repository organisiert. Auf der Übersichtsseite (Abb. 6.7) sind alle diese Dokumenten-Kollektion aufgeführt mit Namen, Zeitstempel, vergebenen Stichworten und Beschreibung. Gegebenenfalls können von hier auch Kollektionen gelöscht oder mit anderen Kollektionen fusioniert werden.

Von der Übersichtsseite kann das ganze Repository als XML-Datei auf den Rechner des Benutzers gespeichert oder auch wieder hochgeladen werden.

Zusätzlich besteht die Möglichkeit temporär die Daten in der Datenbank zu speichern. Um hier eine Anonymität zu gewährleisten, es sollen ja keine Benutzernamen und Kennwörter verwaltet werden, wird ein Zufallskennwort erzeugt und dem Benutzer angezeigt. Nach Eingabe des in einer früheren Sitzung erzeugten Kennwortes werden die Daten wieder aus der Datenbank geladen. Diese Verfahren hat den Vorteil, dass keine direkten personenbezogenen Daten benötigt werden. Allerdings besteht natürlich die Gefahr, dass ein anderer Benutzer durch 'raten' des Kennworts an die Daten gerät.
Um die Größe der Datenbank klein zu halten, müssen alte Einträge nach gegebener Zeit wieder aus der Datenbank entfernt werden, diese Funktion ist zwar schon vorgesehen wird in der jetzigen Version aber noch nicht automatisch durchgeführt.

6.2.3 Bearbeiten von Dokumenten-Kollektionen

Diese Funktion stellt das Kernstück der Anwendung dar. Hier können die Dokumente in zwei verschiedenen Sichten: Tabellarisch (Abb. 6.8) oder als Baumdarstellung (Abb. 6.9), betrachtet werden. Zu einem Dokument können Anmerkungen, in Form einer kurzen Notiz oder als farbliche Markierung, gesetzt werden. In der Übersicht sind für ein Dokument die „üblichen" Informationen zu sehen wie Autor(en), Titel, Publikationsjahr und eine zwei- oder dreizeilige Kurzfassung. Auf Wunsch kann das Abstract des Dokumentes eingeblendet werden und es sind im Normalfall zwei Internetlinks verfügbar: Zur ursprünglichen Fundstelle im Internet und ein zweiter Link zur Beschreibung in der *CiteSeer*-Datenbank (bzw. die Literaturdatenbank, von der dieses Dokument stammt) mit der Möglichkeit dort den Volltext zu sehen. Darüberhinaus ist auch noch eine Druckansicht der tabellarischen Ansicht möglich.
Ohne weitere Bearbeitung bestehen zumeist zwei Gruppen von Dokumenten, eine „Default-Gruppe", mit allen Dokumenten, die in der Datenbank referenziert werden konnten und eine „Rest-Gruppe" mit Dokumenten, die durch die Suchmaschine gefunden wurden, aber nicht in der Datenbank gefunden wurden[116].

Grundsätzliche Optionen eines Benutzers umfassen außerdem die Möglichkeit Dokumente zu sortieren (durch Klick auf die Spaltenüberschrift), ein Dokument aus der Menge zu entfernen oder auf einer gesonderten Seite alle Daten eines Dokuments zu betrachten (Abb. 6.10). Hier besteht auch die Möglichkeit eine BIBTEX Angabe für dieses Dokument zu sehen und alle bisher (auf dieser gesonderten Seite) betrachteten Dokumente als Angaben auf einem PDF-Dokument zu speichern.

Die in Kapitel 3.1.1 beschriebene Funktionalität des Clustern bzw. Gruppierens von Dokumente wird durch Klick auf einen entsprechenden Button in der oben angeordneten Toolleiste ausgelöst. Die konkret verwendete Clustermethode und das Ähnlichkeitmaß wird wird vorher in einem Einstellungsdialog ausgewählt (Abb. 6.11).
Die gebildeten Gruppen sind anschließend zunächst einmal nur durchnummeriert. Manchmal kann eine Gruppierungsverfahren nicht alle Dokumente korrekt in Gruppen verteilen, diese „Restdokumente" verbleiben dann in der „Default-Gruppe". Bei jeder Gruppe hat ein Benutzer die Möglichkeit Beschreibungen anzugeben, die Gruppe zu löschen, eine Gruppe manuell anzulegen und Dokumente von einer Gruppe in eine andere zu verschieben.
Implementiert sind die meisten dieser Funktionalitäten auf der Oberfläche durch AJAX Aufrufe, um eine möglichst hohe Reaktionsgeschwindigkeit der graphischen Oberfläche zu ermöglichen und um den Server weniger zu belasten (bspw. durch eine geringere Menge erstellter HTML-Seiten und weniger Datenübertragung).

[116] Dies kann z. B. dadurch passieren, dass die Suchmaschine neuere Dokumente zurückgeliefert hat, als in der Datenbank gelistet sind.

Abbildung 6.4: Bildschirm *CiteseerCluster*: Startseite

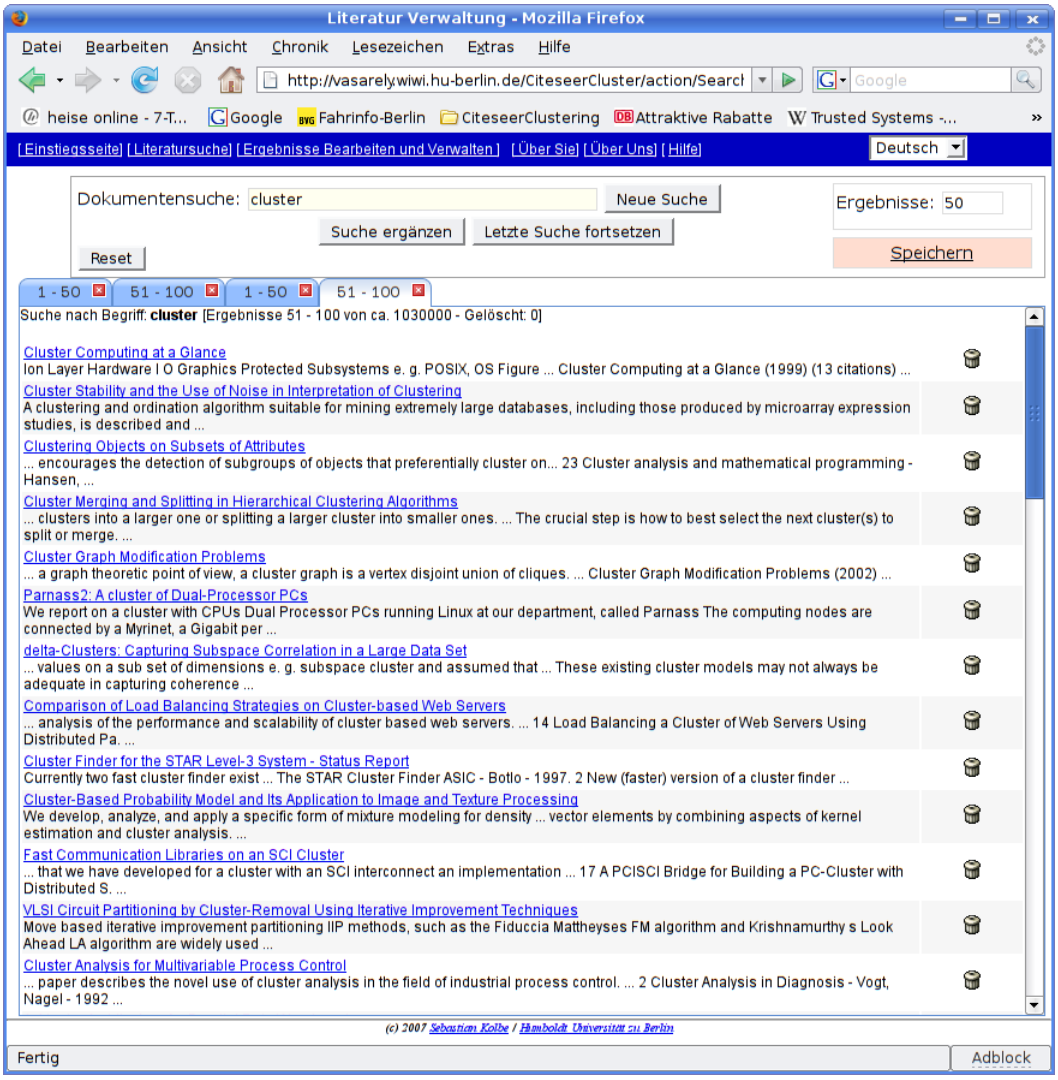


Abbildung 6.5: Bildschirmansicht *CiteseerCluster*:
Suchergebnisse zu zwei verschiedenen Suchbegriffen

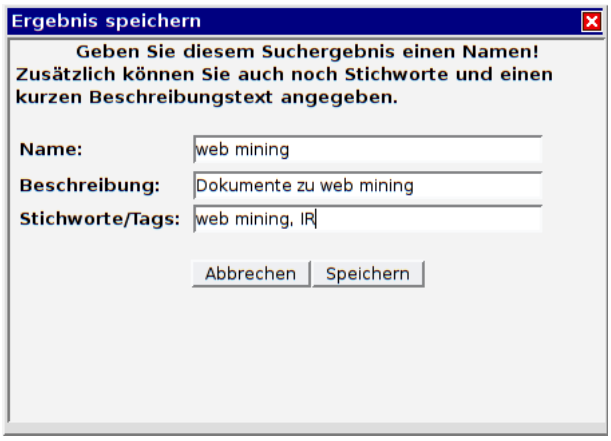


Abbildung 6.6: Bildschirmansicht *CiteseerCluster*: Abspeichern von Suchergebnissen.

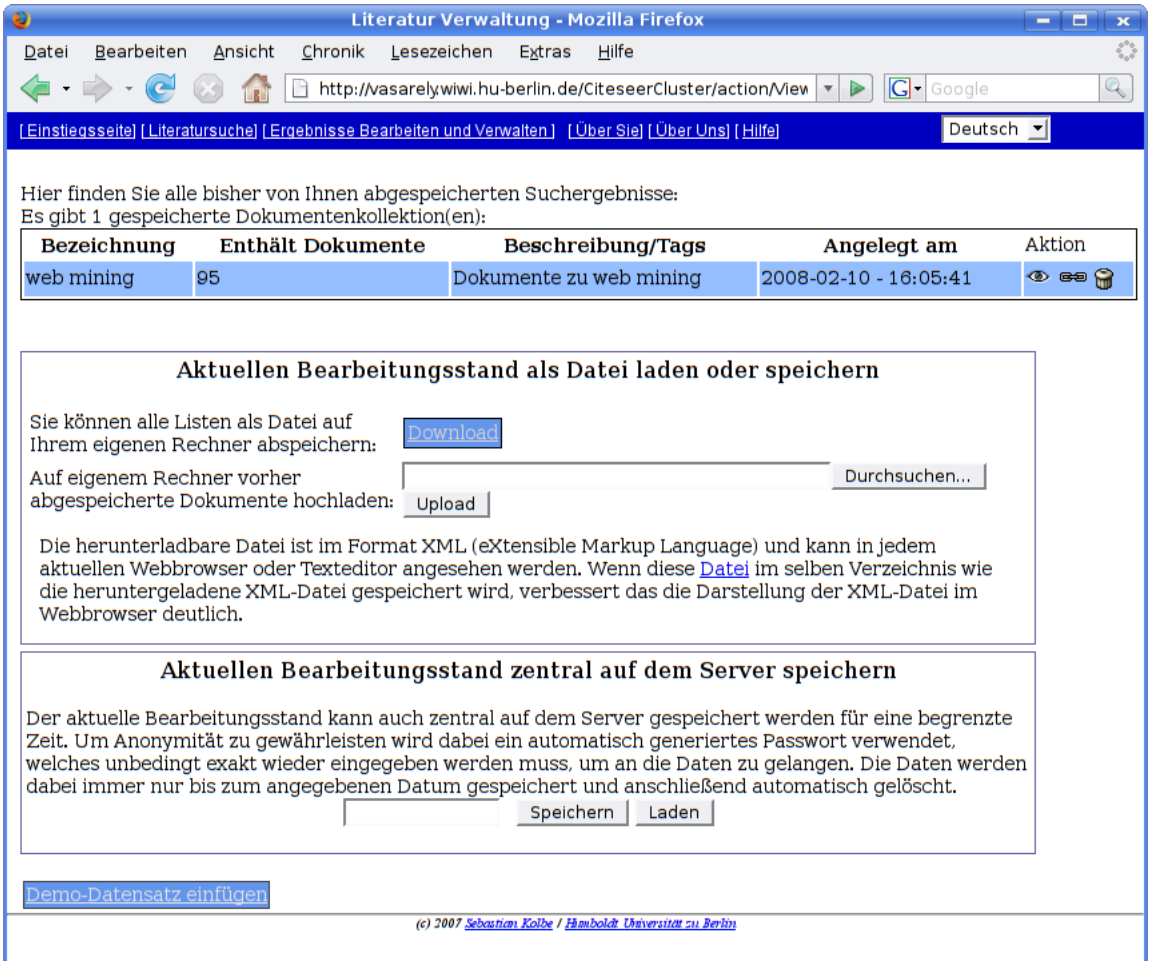


Abbildung 6.7: Bildschirmansicht*CiteseerCluster*: Verwaltung von Dokumenten-Kollektionen.

Abbildung 6.8: Bildschirmansicht *CiteseerCluster*: Tabellarische Ansicht der Dokumenten-Kollektion
Die Toolbuttons oben rechts erlauben die Organisation der Dokumente in Gruppen

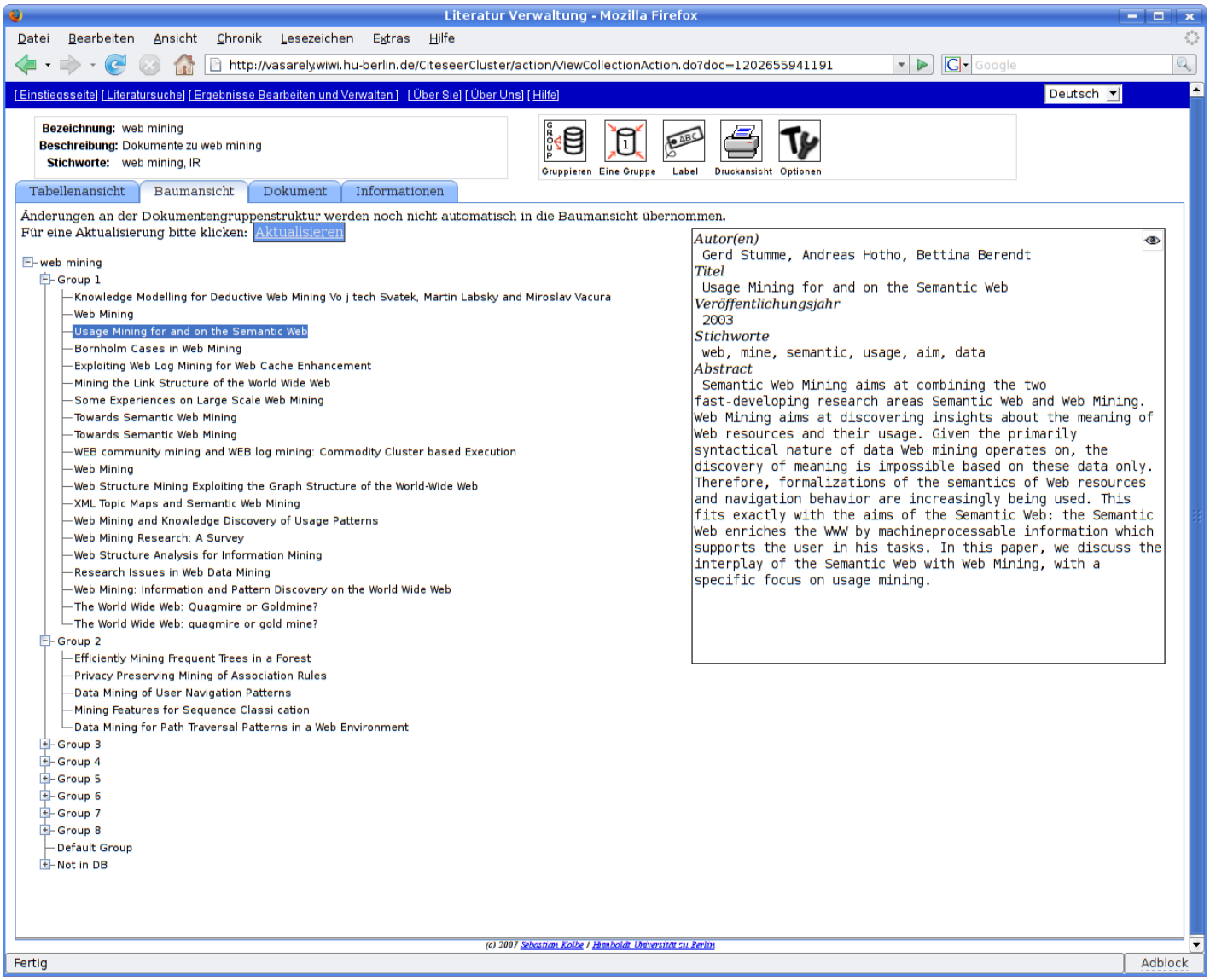


Abbildung 6.9: Bildschirmansicht *CiteseerCluster*: Baumansicht der Dokumentengruppen
Durch Klick auf ein Element des Baumes, wird die Anzeige in der rechten Hälfte des Bildschirms aktualisiert

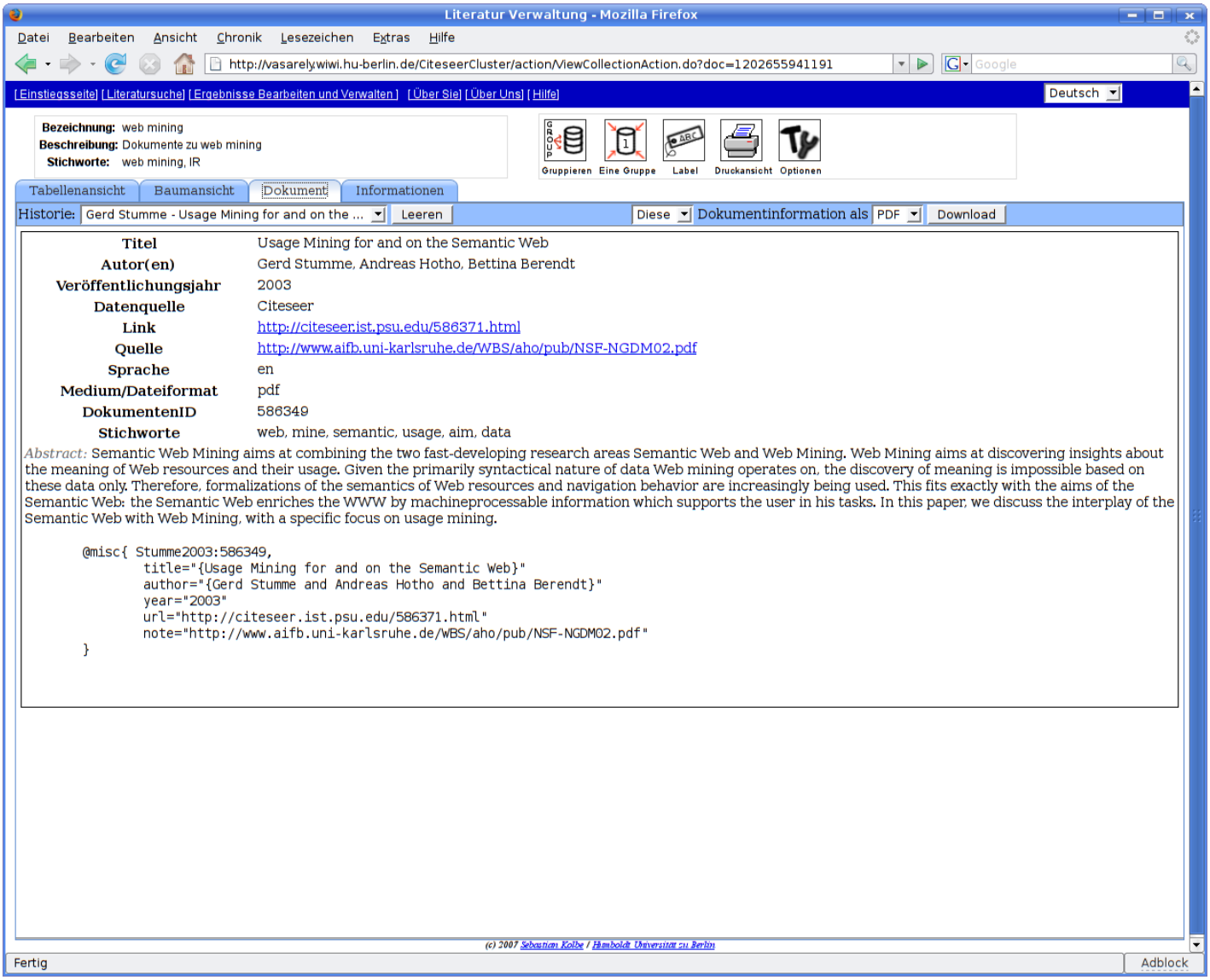


Abbildung 6.10: Bildschirmansicht *CiteseerCluster*: Nach Auswahl eines Dokuments in einer der anderen Ansichten wird hier eine detailierte Ansicht des Dokuments inkl. einer BIBTEX Darstellung angezeigt

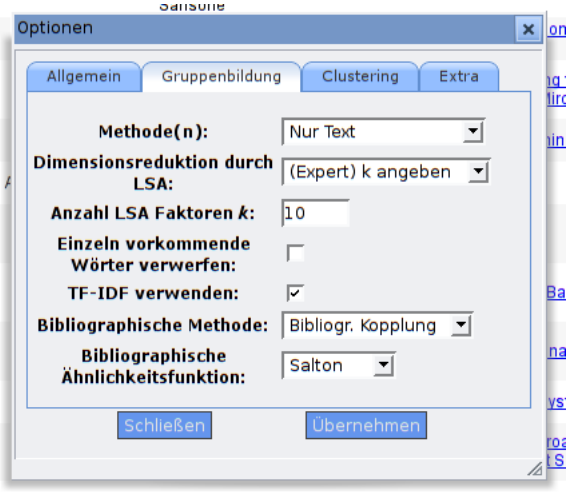


Abbildung 6.11: Bildschirmansicht *CiteseerCluster*: Dialog zum Einstellen von Optionen zur Änlichkeitsanalyse und Clusterung

6.3 Interne Arbeitsweise

Aus den in Kapitel 5.5.1 wird nachfolgend (Abschnitt 6.3.3) ein exemplarischer Datensatz ausgewählt und anhand dieses Datensatzes werden einige der implementierten Algorithmen, die in Kapitel 3 vorgestellt wurden, ausgewertet.

Zuvor ist in Abschnitt 6.3.1 das grundsätzliche Verfahren der Dokumentensuche und in Abschnitt 6.3.2 die Dokumentengruppierung dargestellt. Abschließend sind in Abschnitt 6.3.5 die Möglichkeiten der Optimierung zusammengetragen. Damit sollen die, im Kapitel 1.3 formulierten technischen Fragen, beantwortet werden.

6.3.1 Dokumentensuche

Relevante Dokumente zu einer Stichwortsuche werden, wie schon oben angedeutet, mit einer externen Suchmaschine (Yahoo) ermittelt. Das Ergebnis einer solchen Suche ist eine Anzahl *CiteSeer* spezifischer Dokumenten-IDs, die zunächst geordnet (es kommen oft ID-Nummern mehrfach), teilweise umgewandelt in numerische (manche Dokumenten-IDs sind „symbolisch“ zusammengesetzt aus Name, Jahreszahl und Anfang des Titels) durch Abfrage der *CiteSeer*-Webseite und anschließend gegen die lokale Datenbank referenziert werden. Dokumenten-IDs, die nicht in der Datenbank gespeichert sind, werden in eine gesonderte Gruppe verschoben. Diese Dokumente sind von der weiteren Verarbeitung ausgenommen, werden aber auf der Oberfläche als gesonderte Resultate dem Benutzer angezeigt.

Der enthaltene Text (Titel und Abstract) aus der Datenbank gelesener Dokumente wird zunächst verarbeitet, um zusätzliche Informationen wie Stichwörter dem Benutzer anzeigen zu können und eine Vorverarbeitung (Filtern von Wörtern) für die Analyse der Textähnlichkeit zu realisieren. In Abbildung 6.12 ist die Abfolge der Schritte dargestellt.

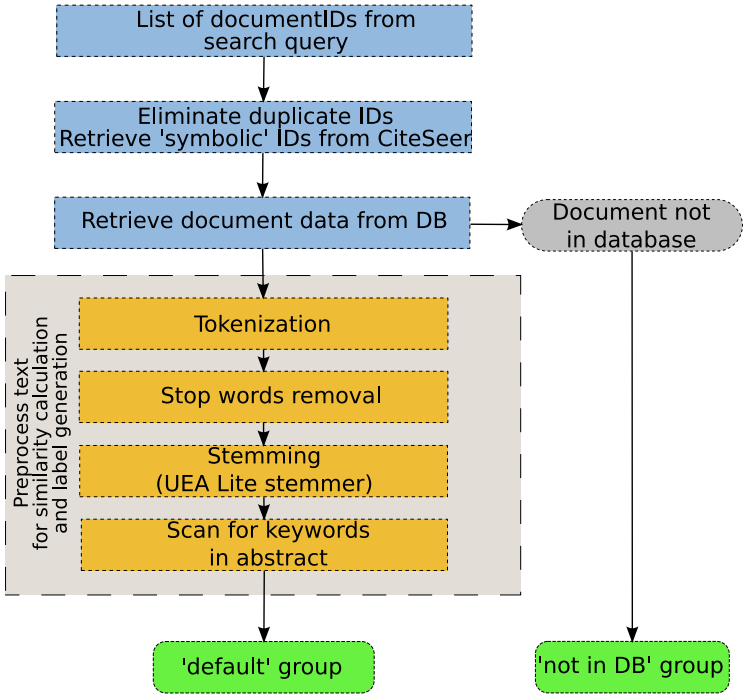


Abbildung 6.12: Schematische Darstellung der Dokumentensuche und Vorverarbeitung.

6.3.2 Dokumentengruppierung

In der Software sind bibliometrische sowie auch textuelle Verfahren der Ähnlichkeitsanalyse implementiert. Dabei stehen einem Benutzer bei den bibliometrischen Methoden die Möglichkeiten der Analyse der Bibliographischen Kopplung sowie der Koziationsanalyse zur Verfügung. Die Ähnlichkeitsberechnung kann hier mit dem *Jaccard*, *Dice*, *Amsler* oder mit dem *Salton* Maß erfolgen.
Die Analyse der Textähnlichkeit ist mit und ohne den Einsatz von LSA möglich. Auf Wunsch ist die automatische Ermittlung einer (quasi-) optimalen Anzahl von Faktoren möglich.[117]

Eine, sofern anwendbare, anschließende Kombination der beiden Verfahren beruht auf der linearen Kombination der Matrizen. Für die folgende Clusterung sind mehrere Verfahren durch die Software *CLUTO* implementiert, unter anderem hierarchische agglomerative Verfahren mit „Complete Link“ und „UPGMA“, sowie ein *K-Means* ähnliches Verfahren „direct clustering“ und *RBR*.
Eine, in einem Suchbereich, optimale Clusterzahl wird ermittelt durch die Analyse der *Silhouette*-Werte. Dabei wird der Suchbereich definiert als zwischen 2% und 15% der Dokumentenmenge. Eine, sofern vorhanden, Anzahl von LSA Faktoren wird entsprechend berücksichtigt als obere Schranke.
In Abbildung 6.13 ist das Verfahren schematisch dargestellt.

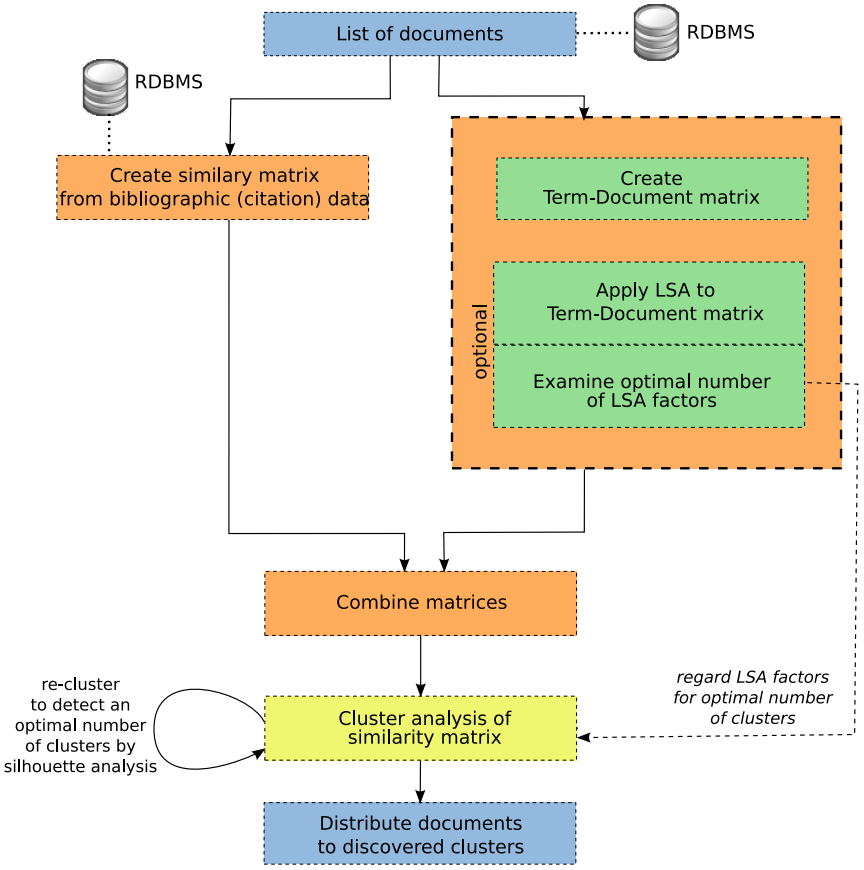


Abbildung 6.13: Schematische Darstellung der Clusterbildung, bei Kombination von textueller und bibliographischer Analyse.

[117] Dies ist, wie in Kapitel 3.1.1.2 dargestellt, nur eingeschränkt möglich und fehleranfällig.

6.3.3 Testdatensatz für Evaluation

Aus dem in Kapitel 5.5 vorgestellten Korpus wurde zu Evaluations- und Testzwecken ein Datensatz zum Suchbegriff `web mining` extrahiert. Die Größe des Datensatzes ist dabei so gewählt, wie sie ein potentieller Benutzer möglicherweise auswählen würde, um die Dokumentenmenge am Bildschirm nachvollziehen zu können. Die Ergebnisse der Auswertung dieser Daten ist im Anschluss (Abschnitt 6.3.4) zu finden.
Eine Zusammenstellung der Daten dieses Testdatensatzes liefert folgende Tabelle:

Zusammenstellung Testdatensatz	
Suchbegriff	`web mining`
Dokumente	132
Alle Terme	10 072
Davon unterschiedliche Terme	1 923
Dokumente verbunden durch BC	97
Dokumente verbunden durch CC	53
Dokumente verbunden durch BC & CC	110

Tabelle 6.1: Kennzahlen des Testdatensatzes

6.3.4 Evaluation Testdaten

Wie in Kapitel 3.1.2.2 vorgestellt, wird der *Silhouette*-Wert für eine Clusterbegutachtung eingesetzt.

Für diesen Testdatensatz mit 132 Dokumenten kann angenommen werden, dass eine „interessante“ Clusterung im Bereich von 5 – 15 Gruppen, d. h. verschiedenen Themen, liegt. Dieser Wert wurde aufgrund praktischer Erfahrung und Beobachtung in der Evaluation gewonnen.

Einen Vergleich verschiedener Clustermethoden auf dem Testdatensatz mit der Bibliographischen Kopplung gibt Abb. 6.14. Man beachte, dass insbesondere die Werte für das Agglomerative Hierarchische Clustern mit „Complete Linkage“ besonders schlecht abschneiden. Dies liegt im wesentlichen darin begründet, dass dieses Verfahren eher für größere Datensätze mit einer höheren Clusteranzahl gedacht ist. Möglicherweise sind auch die Daten dieses Testdatensatzes ungünstig. Die anderen Clustermethoden (K-Means[118], RBR[119], AHC – UPGMA) gruppieren im „interessanten“ Bereich den Datensatz sehr ähnlich.

[118] Ein „echtes“ K-Means ist in *CLUTO* nicht implementiert, es wurde die „direct-clustering“ Methode stattdessen verwendet, die zusätzlich ein Clustergütemaß in Form einer Optimierungsfunktion verwendet.
[119] Repeated-Bisections mit einer globale Optimierungsfunktion (wie bei „direct-clustering“).

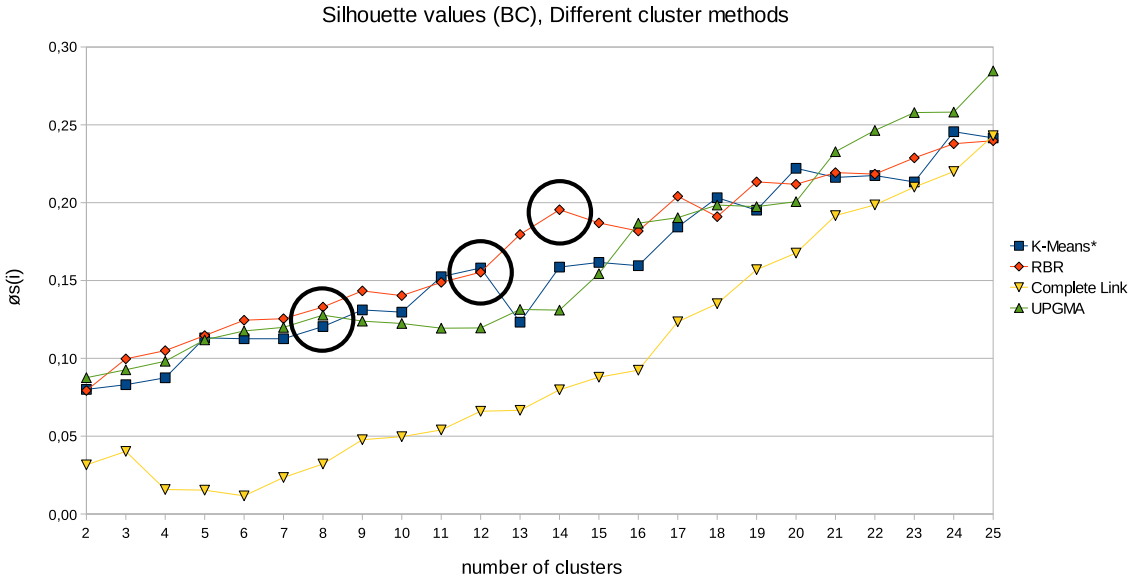


Abbildung 6.14: Vergleich des *Silhouette*-Wertes mit verschiedenen Clustermethoden und Bibliographischer Kopplung.

Punkte lokalen Maximums (im Bild markiert) der Silhouettekurve finden sich bei diesen Cluster bei $k = 8$, $k = 12$ und $k = 14$. Darüberhinaus finden sich unterschiedliche weitere Maxima, z. B. für den *K-Means* Algorithmus etwa bei $k = 20$.

Die Analyse der Textähnlichkeit birgt zunächst einmal das Problem der Menge der verwendeten LSA-Faktoren. Abb. 6.15 zeigt einen *Scree*-Plot der LSA-Faktoren für den Datensatz:
In dieser Grafik ist deutlich der charakteristisch steile Abfall der singulären Werte am linken Rand zu sehen. Nach dem Kaiser-Guttman-Kriterium sollten keine Faktoren kleiner 1.0 verwendet werden, was die Menge der Faktoren auf 28 begrenzt.
Die Analyse des Abknickpunktes (Ellenbogen-Kriterium) lässt sich am besten an einem Ausschnitt betrachten (siehe Abb. 6.16):
Die Abknickpunkte lassen sich anhand des Gradienten identifizieren, allerdings zeigt nicht immer der größte Gradient den „besten“ Abknickpunkt an. Nach der Grafik wären bspw. $k = 3$, $k = 7$ oder $k = 12$ als Kandidaten für die LSA zu sehen. Im Grundsatz sollte aber immer der letzte, bzw. am weitesten rechts liegende, Abknickpunkt verwendet werden, sofern die Menge der Faktoren noch im Rahmen des Gewünschten steht. Es ist wichtig einen oberen Grenzwert für die Anzahl der Faktoren zu definieren, da diese Zahl großen Einfluß auf das nachfolgende Clustern hat. Aufgrund der Interpretation der LSA als eingeschränkte Themenauswahl, sollte die Anzahl der Cluster nicht höher liegen als die Anzahl der LSA-Faktoren.
Im Programm wird daher die Faktorenanzahl nach dem Kaiser-Guttman-Kriterium bestimmt und auf Grundlage des Gradienten ein Abknickpunkt im Scree-Plot bestimmt.

Ein Vergleich der Clusterlösungen verschiedener Clustermethoden bei der Analyse der Textähnlichkeit (Abb. 6.17) zeigt, dass das Verfahren des Agglomerativen Hierarchischen Clusters hier eindeutig die besten Ergebnisse liefert. Die beiden Varianten „Complete Linkage“ und „UPGMA“ liegen sehr

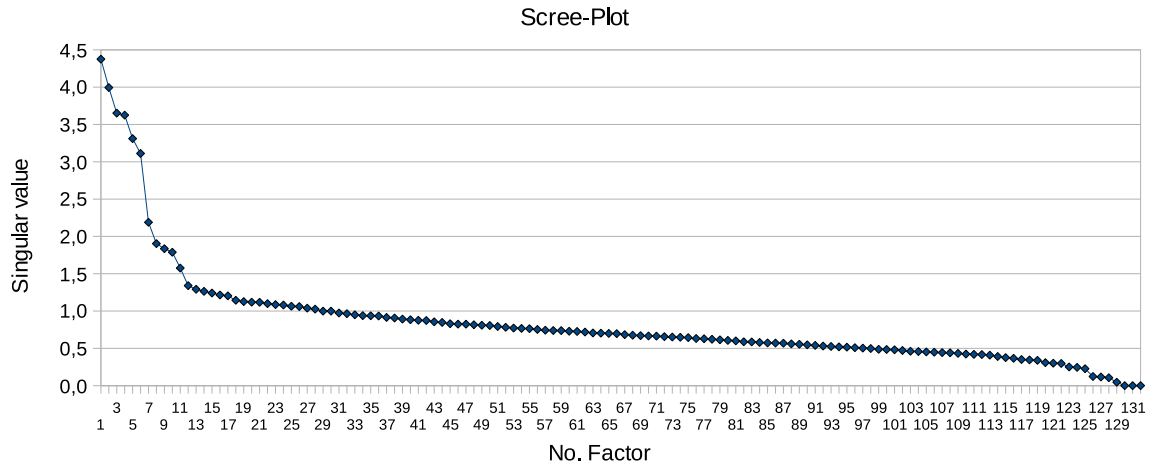
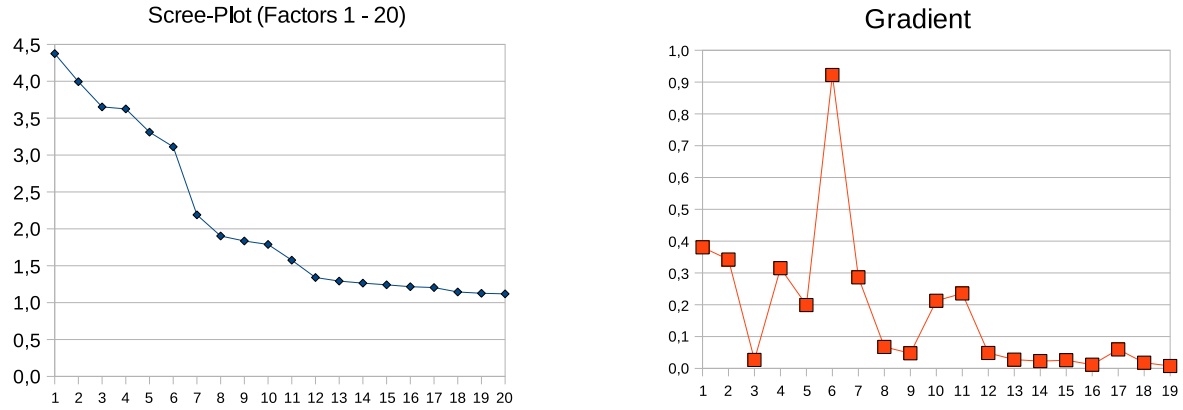


Abbildung 6.15: Scree-Plot der Singular Values aus der SVD Analyse



(a) Scree-Plot der ersten 20 SVD-Faktoren

(b) Gradienten der SVD-Faktoren

Abbildung 6.16: Scree-Plot der ersten 20 Singulären Werte aus der SVD Analyse.

dicht beieinander, das „UPGMA“-Clustern kommt aber auch hier zu einer ausgeprägerten (mit definierten Minima und Maximal) Kurve.
Eine Analyse des Kurvenverlaufs zeigt deutliche Maxima bei $k = 7$, $k = 8$ und $k = 10$, wobei das hierarchische Clustern hier den höchsten Wert aller Clustermethoden liefert. Allerdings tendieren die hierarchischen Verfahren (zumindest mit *CLUTO*) sehr dazu, die Dokumente sehr ungleich zu verteilen, was meist zu mind. einem sehr großen und mehreren sehr kleinen Clustern führt, was möglicherweise nicht zielführend ist.
Auch zu beobachten ist, dass die Clusteranalyse der textuellen Informationen deutlich mehr Struktur (also höhere Werte im Silhouette-Maß) zeigen, als die Analyse der bibliographischen Informationen.

Es wurde, das im Kapitel 3.1.3 beschriebene Verfahren der Linearkombination im Programm implementiert. Die Kombination von Textanalyse und Bibliographischer Kopplung zeigt Abb. 6.18 mit verschiedenen Werten für den Gewichtungsfaktor α.

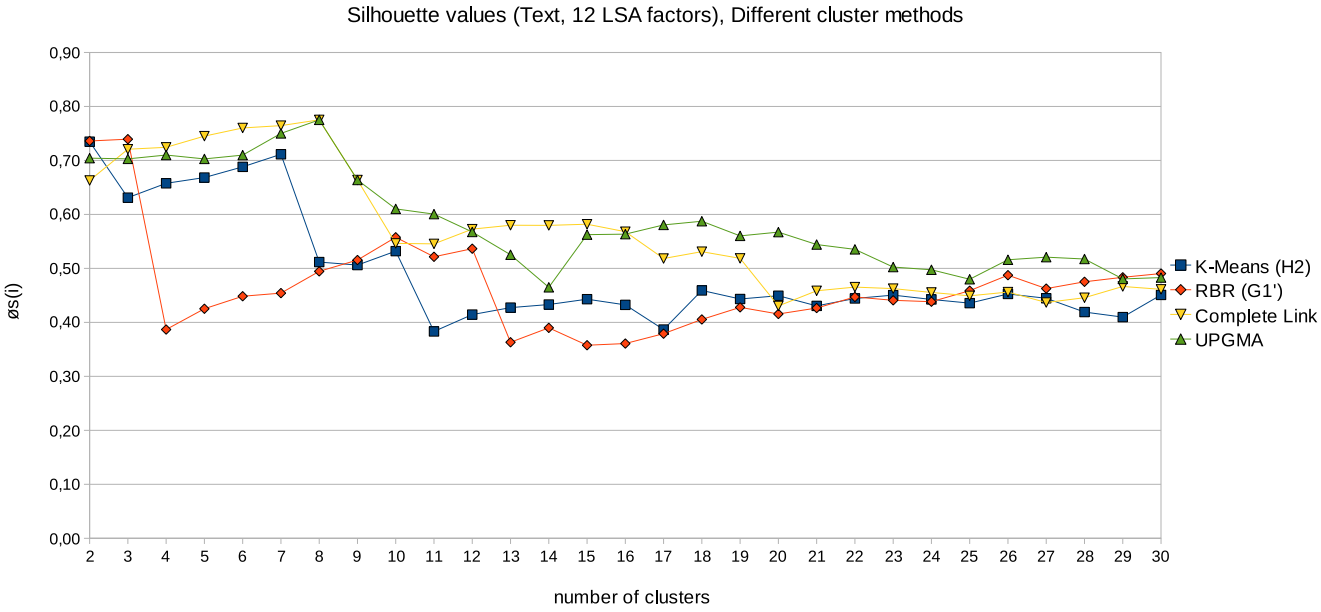


Abbildung 6.17: Vergleich der *Silhouette*-Werte verschiedener Clustermethoden bei Textähnlichkeitsanalyse mit 12 LSA Faktoren.

Wie in der Grafik zu sehen, steigt der Silhouette-Wert global nicht durch die Kombination der Matrizen. Es gibt eine stark ausgeprägten Abfall der Kurve zwischen $k = 5$ und $k = 10$. Das lässt darauf schließen, dass dort ein lokales Minimum des Clusterzusammenhangs erreicht ist. Eine genaue Betrachtung der in diesem „interessanten“ Bereich sich befindenen Maximumpunkte kann einen guten Aufschluß über die Clusteranzahl geben, bei der sich die beiden implementierten Maße ergänzen.
Eine schwer zu entscheidende Frage ist die nach einem optimalen Wert für α, trotz der schwer zu interpretierenden Ergebnisse Ergebnisse sollte die bibliographische Methode bei der Kombination bevorzugt werden (also ein Wert von $\alpha < 0.5$) (siehe auch Kapitel 3.1.3).

Die Abbildung 6.19 zeigt die gleiche Kombination wie Abb. 6.18, unter der Verwendung des gewichteten geometrischen Mittels bei der Kombination der Matrizenwerte.
Hier ist zu beobachten, dass sich die *Silhouette*-Werte weit unter dem Niveau der bei der Linearkombination beobachteten Werte bewegen. Diese Methode gibt damit in diesem Testdatensatz ist schlechteren Werte ab und lässt auf keine gute Eignung für eine Kombination schließen.

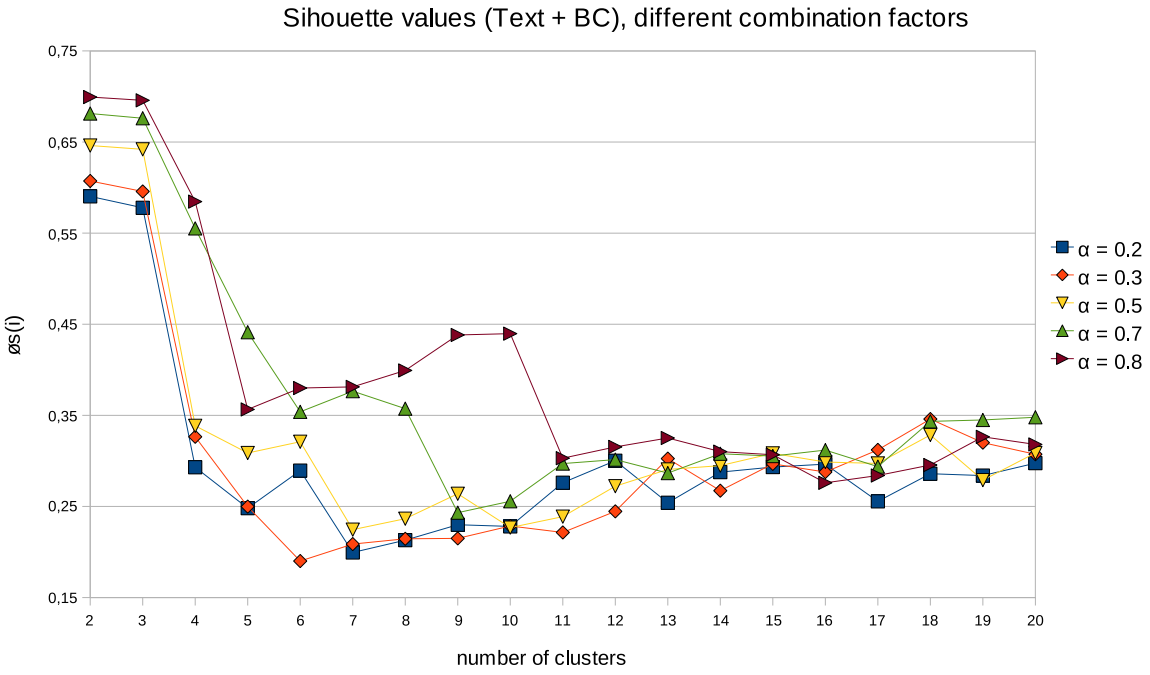


Abbildung 6.18: Vergleich der *Silhouette*-Werte bei linearer Kombination von Text- und bibliographischer Ähnlichkeit und verschiedenen Werten für Gewichtungsfaktor α.

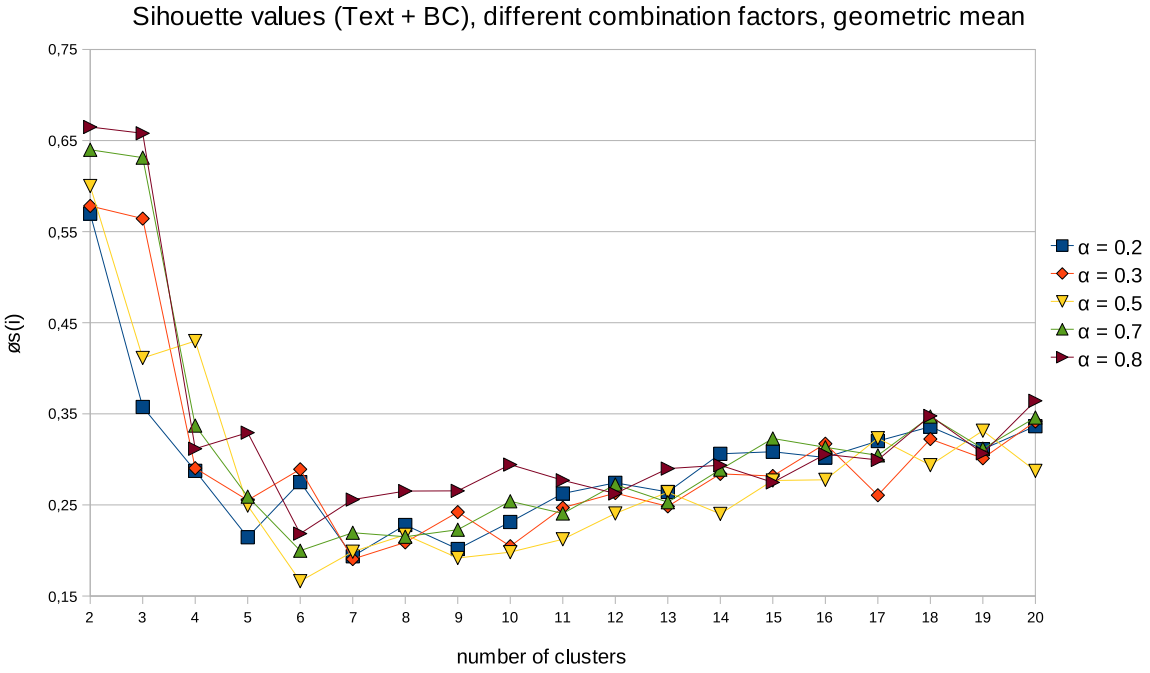


Abbildung 6.19: Vergleich der *Silhouette*-Werte bei Kombination von Text- und bibliographischer Ähnlichkeit mit geometrischem Mittel und verschiedenen Werten für Gewichtungsfaktor α.

6.3.5 Leistungsbewertung und Optimierung

Mit der hier vorgestellten Software ist es grundsätzlich möglich, Dokumente in Gruppen ähnlicher Objekte aufzuteilen. Das Ähnlichkeitskriterium ist dabei einstellbar und mit der Technik der Kombination von Ähnlichkeitsmatrizen sind auch verschiedene Methoden der Ähnlichkeitsermittlung zusammenführbar.

Für den Einsatz in einem Online-System sind bestimmte weitere Anforderungen zu beachten, wie bspw. Reaktionszeiten und Verarbeitungszeiten. Die Reaktionszeiten sind dabei im Allgemeinen direkt von den Verarbeitungszeiten abhängig. Eine Zeitmessung verschiedener Teile der Verarbeitung mit dem Testdatensatz ist in Abbildung 6.20 zu sehen. Wobei die drei algorithmischen Teile (zwei Ähnlichkeitsmatrizen erstellen und Clustern) den Hauptteil der Zeit einnehmen.[120] Die vorgeschaltete Dokumentensuche über die Suchmaschine und Vorverarbeitung ist hier nicht gezeigt, da die ermittelten Werte zwar sehr schwankend waren, aber trotzdem sich im Vergleich kleinen Bereichen bewegten (kleiner 2 Sekunden).

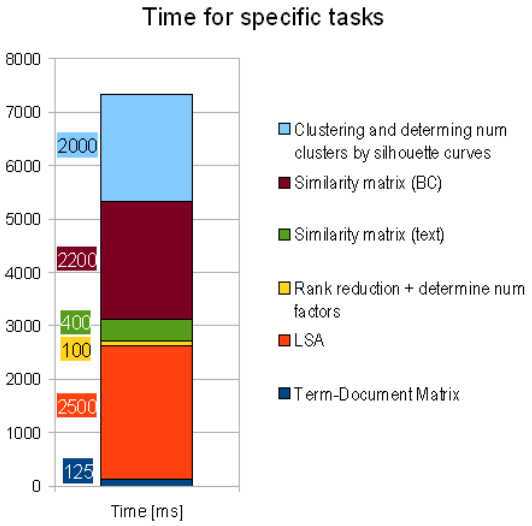


Abbildung 6.20: Verarbeitungszeit Testdatensatz.

Optimierung und Diskussion

Um die Verarbeitungszeit zu optimieren, sind verschiedene Ansätze denkbar. Zum einen könnten alle Ähnlichkeitswerte von Dokumenten in einer Datenbanktabelle vorberechnet werden. Diese Lösung wurde hier aus Kapazitätsgründen unterlassen, da diese Datenbanktabelle extrem groß wird. In diesem System wurde daher nur die absolute Anzahl der Zitationen von und auf ein Dokument für jedes Dokument vorberechnet. Dies entspricht den beiden Termen, im Nenner des *Jaccard* Indexes, die so vorberechnet werden konnten.

Für ein vielgenutzes System könnten als Alternativlösung die bibliographischen Ähnlichkeitswerte komplett vorberechnet werden und die textuellen Ähnlichkeiten weiterhin im Onlineverfahren. Die bei den obigen Messwerten Dies könnte ein akzeptabler Kompromiss bzgl. Vorberechnungszeit und Speicherplatz darstellen, da es nicht zwischen allen Dokumenten bibliographische Ähnlichkeiten gibt (bzw. diese Null sind), während eine textuelle Ähnlichkeit praktisch immer vorhanden ist.

[120] Gemessen mit Hardware / Software: AMD-Athlon64 3200+, 1Gb Speicher, MySQL Server 5.0.45, Java JDK 1.6.01, Ubuntu Linux 7.10.

7 Evaluation des Systems

In den vorangegangen Kapiteln wurde beschrieben, welche Techniken und Algorithmen in der entwickelten Software implementiert wurden. Um nun aber eine Aussage machen zu können, ob sich das Programm grundsätzlich eignet für den Zweck der Literaturrecherche, bedarf es weiterer Untersuchungen. Diese müssen den angestrebten Nutzerkreis und die zur Verfügung stehenden Daten berücksichtigen.

Die für dieses System verfügbaren Möglichkeiten zur Evaluation sind in Kapitel 3.2 beschrieben und sollen hier angewendet werden.

Zunächst werden nun die Ziele der Evaluation dargelegt, darauf aufbauend die verschiedenen Teile der Evaluation und die Durchführung beschrieben sowie die Auswertung vorgenommen.

7.1 Ziele der Evaluationen

Ein Evaluation des Software-Prototypen verfolgt folgende zwei Ziele:

- **Bewertung und Verbesserung der Benutzerfreundlichkeit**
  Der konzeptionelle Ansatz der Benutzerschnittstelle soll bewertet werden und Anregungen für die Aufnahme von zusätzlichen Anforderungen an Programmgestaltung und Funktionalität gewonnen werden.

- **Einschätzung der „Nützlichkeit“ der im Programm implementierten Funktionalität**
  Die Aufbereitung & Visualisierung der Literaturdaten soll auf ihre grundsätzliche Nützlichkeit für die Zielgruppe bewertet werden. In der Evaluation soll allerdings nicht geklärt werden, ob die Programmfunktionalität der Bildung von Dokumentengruppen grundsätzlich besser ist, als eine „klassische“ Suche im Internet.

7.2 Methodik

Die zur Verfügung stehenden Methoden einer Evaluation wurden in Kapitel 3.2 detailliert beschrieben. Für die Evaluation des vorliegenden Projekts wurden die Technik des „Laut-Mitdenken“ sowie ein abschließender Fragebogen gewählt (analog zu Vorschlägen in [Heg03, S. 35]).
Damit konnten die Vorteile einer verhaltensbasierten Technik mit meinungsbasierter Methodik kombiniert werden, ohne den hohen Aufwand der direkten Beobachtungstechniken eingehen zu müssen.

Um die Einschätzung der „Nützlichkeit“, bzw. der implementierten Aufbereitung zu ermöglichen, wurde die Evaluation als *Quasi-Experiment* geplant. Dazu wurden zwei Gruppen der Versuchspersonen gebildet: Experimental- und Kontrollgruppe. Eine Gruppe hatte dabei alle Funktionen des Programms zur Verfügung, insbesondere mit den Funktionen zur Gruppierung von Dokumenten. Die andere Gruppe hatte eine Version des Programms zur Verfügung, die alle Funktionen bot, bis auf die Möglichkeit Gruppen von Dokumenten zu bilden und Dokumente in Gruppen zu verschieben.

Die Bildung von Dokumentengruppen, sowie das Arbeiten mit diesen (Verschieben von Dokumenten, Erstellung von Gruppenbeschreibungen) sind als unabhängige Variablen zu betrachten. Das Ziel dieses Teils der Evaluation (siehe oben) ist die Beurteilung der „Nützlichkeit“ der im Programm implementierten Funktionalität. Eine qualitative, vergleichende Auswertung der von den Versuchspersonen angefertigten Ergebnisse sollte hier angewendet werden.

7.3 Versuchspersonen / Zielgruppe

Da die Zielgruppe des Projekts (siehe Kapitel 1.1) aus Studenten und Wissenschaftlern besteht, sollten die VP auch aus dieser Gruppe stammen.
Die Datenbasis des Programms hat einen Schwerpunkt in den Fachrichtungen Informatik und Mathematik. Daher sollten die Versuchspersonen auch diesen Hintergrund mitbringen.
Ein nützlicher Einsatz des Programms ergibt sich in den meisten Fällen für einen Anwender nur, wenn dieser mit der Anfertigung einer wissenschaftlichen Arbeit betraut ist. Daher sollten Versuchspersonen mindestens schon ein Mal eine solche Arbeit erstellt haben, was zumeist bei Studierenden im fortgeschrittenen Hauptstudium der Fall sein sollte. Bei wissenschaftlichen Mitarbeitern wird die Voraussetzung ebenso als erfüllt angenommen.

7.4 Vorbereitung und Versuchsbeschreibung

Für die Evaluation wurde ein Szenario entwickelt, das Versuchspersonen mit jeweils zwei Aufgaben aus unterschiedlichen Wissensgebieten betraute. Ziel war es, relevante Dokumente zu einem Fachthema zu finden und Unterthemen zu identifizieren. Die Unterthemen sollten mit Dokumenten, idealerweise sogenannten Kerndokumenten, als sog. Mindmap oder auch mit Überschriften und Autorennamen dargestellt werden. Die Versuchsaufgabenbeschreibung wurde dabei gemeinsam mit Fr. Prof. B. Berendt und Beate Krause.

Als Ausgangspunkt des Szenarios war die Recherche zu einer Facharbeit anzunehmen, wobei die erste Aufgabe als Training zu verstehen war, um mit dem System in Kontakt zu kommen.
Die Aufgabenstellung war für alle Versuchspersonen gleich, allerdings wurde das konkrete Wissensgebiet jeder Versuchsperson individuell aus einer vorher vorbereiteten Liste von Gebieten zugeteilt, abhängig von der eigenen Einschätzung. Dabei sollten alle Teilnehmer ein Wissensgebiet bearbeiten, von dem sie ein Grundwissen mitbrachte, aber gleichzeitig auch keine Spezialisten waren. Damit sollte verhindert werden, dass die Versuchsperson ihr schon vorhandenes Wissen zu Papier bringen würden, unabhängig von der Benutzung des Programms, oder im anderen Extrem, keine Unterthemen identifizieren könnten, weil das Fachgebiet ihnen im Ganzen als zu „undurchsichtig“ erschiene. Die zur Auswahl stehenden Fachgebiete, die von der Versuchsperson in fünf verschiedenen Kategorien (von „verstehe ich gar nichts“ über „mittel“ bis zu „verstehe ich sehr viel“) eingeschätzt werden sollten, waren:

- **Web mining**
- **Machine learning**
- **Kernel**
- **Network**

- **Cluster**
- **Information Retrieval**
- **Social networks**
- **Semantic Web**
- **RFID**
- **Grammar**

Die konkrete Auswahl zweier Wissensgebiete wurde vom Versuchsleiter nach den oben angegeben Grundsätzen durchgeführt.

Zur Auswertung wurde ein Fragebogen entworfen mit Fragen zur Benutzbarkeit des Systems und Problemen sowie zu möglichen Erweiterungen und Wünschen der VP. Eine genauere Beschreibung des Fragebogens findet sich unten in Abschnitt 7.4.1; der Fragebogen selbst sowie die Aufgabenbeschreibung ist in Anhang D zu finden.

Um einen Eindruck zu bekommen, ob die implementierten Funktionalitäten zur Gruppierung für die Zielgruppe einen Nutzen bringen, wurden die VP in zwei Gruppen unterteilt. Eine Gruppe sollte alle Funktionen des Programms, insbesondere die Gruppierung, nutzen, während die andere Gruppe nur die Suchfunktionalität und Listendarstellung zur Auswahl hatte. Durch diese Einteilung in zwei Gruppen waren die Aufgabenbeschreibungen und die Fragebögen leicht unterschiedlich. Die Aufgabenbeschreibungen waren detailliert und mit den entsprechenden Icons bebildert, um die Orientierung der Versuchsteilnehmer mit dem Programm zu erleichtern.
Die Unterschiede zwischen Kontrollgruppe und Experimentalgruppe wurde mit zusätzlichen Fragepunkten gelöst, da eine Auswertung der produzierten Mindmaps bzw. Literaturlisten als nur schwer möglich klassifiziert wurde.

7.4.1 Fragebogen

Der Fragebogen wurde nach den fünf Kriterien konzipiert (siehe in Abschnitt 3.2.3, Seite 52), die schon für die Evaluation einer früheren Form des Programms von [Pap07] entworfen wurde.
Nach diesen Kriterien wurden 21 Fragen (siehe Anhang D) erstellt, die den entsprechenden Kriterien zugeordnet sind.
Zusätzlich konnten 16 Fragen bei der Experimentalgruppe bzw. 9 Fragen bei der Kontrollgruppe beantwortet werden, mit spezifischen Fragen zu einzelnen Programmelementen und -funktionen.
Für beide Gruppen bestand die Möglichkeit weitere Wünsche und Kommentare zu der Software entweder verbal zu äußern (Erfassung durch das „Laut-Mitdenken“ Protokoll) oder schriftlich zu formulieren.

Die Zuordnung der Fragebogenfragen zu den Kriterien:

Kriterium:	Nummer der Frage:
Satisfaction (Zufriedenheit)	7, 9, 13, 21
Efficiency (Effizienz/Produktivität)	16, 17, 18
Usefulness (Zweckmäßigkeit)	14, 15, 20
Learnability (Erlernbarkeit)	1, 2, 3, 6, 11, 19
Control (Kontrolle / Fehlertoleranz)	4, 5, 8, 10, 12

Tabelle 7.1: Zuordnung: Fragebogen – Kriterien

Die einzelnen Fragebogenpunkte konnten in einem 5-teiligen Likert-Schema von „*Stimme voll und ganz zu*“ bis zu “*Stimme überhaupt nicht zu*“ beantwortet werden. Dabei waren alle Punkte als Behauptung formuliert.
Im zweiten Teil des Fragebogens (der für Kontroll- und Experimentalgruppe unterschiedlich war), wurde dagegen nach konkreten Punkten der Verbesserung, bzw. einzelnen Programmfunktionen gefragt, die auch in einer 5-teiligen Skala beantwortet werden konnten von „*Sehr hilfreich*“ bis zu „*Kann ich nichts mit anfangen*“.
Zusätzlich konnte jeder Punkt des Frageboges mit „*Nicht zutreffend*“ beantwortet werden. Dies hatte den Sinn, einer Versuchsperson die Möglichkeit zu geben, Fragen zu Programmteilen (bspw. Hilfeseiten etc.) explizit nicht zu beantworten um sich nicht Antworten „ausdenken“ zu müssen.

Zum Abschluss waren noch drei Fragen zu bereits selbst verfassten wissenschaftlichen Arbeiten mit „Ja“ oder „Nein“ zu beantworten. Diese Fragen sollten die Voraussetzung an die Versuchsteilnehmer verifizieren und hatten keine inhaltliche Bedeutung.

7.5 Durchführung

Die Evaluation wurde im August 2007 durchgeführt mit Studenten im Hauptstudium der Informatik der HU Berlin und der Universität Kassel, die zuvor über mehrere Mailinglisten angeworben wurden. Hinzu kamen wissenschaftliche Mitarbeiter aus dem Fachbereich Wirtschaftsinformatik der Humboldt Universität zu Berlin und dem Fachbereich Informatik der Universität Kassel.
Allen Teilnehmern wurde der Zeitaufwand mit Kinogutscheinen bzw. bei den Teilnehmern aus Kassel mit 20 Euro vergütet.

Die Evaluation wurde jeweils in Räumen der Universität durchgeführt. An Hardware standen dabei Computer sowohl mit MS-Windows XP, als auch Ubuntu-Linux zur Verfügung. Es wurde der Firefox-Webbrowser in Version 2 verwendet. Die Bildschirme waren immer TFT-Geräte mit 1280×1024 Pixel Auflösung.

Es wurden bis zu maximal drei Versuchspersonen gleichzeitig in einem Durchgang betreut, so dass auf Fragen individuell reagiert werden konnte. Die Teilnehmer wurden außerdem gebeten Gedankengänge, Probleme, Wünsche und sonstige Vorschläge laut zu artikulieren, was im Protokoll mitgeschrieben wurde.

Alle Teilnehmer erhielten eine anonyme Nummer, um Datenschutzanforderungen gerecht zu werden. Zusätzlich wurden die folgenden Informationen von jedem Teilnehmer aufgenommen:

- Studiengang und -stadium (sofern anwendbar)
- Bekanntheit und Nutzung von Literaturdatenbanken
- Falls Literaturdatenbanken verwendet wurden: Bekanntheit der *CiteSeer* Datenbank

Ein Durchgang dauerte zwischen 60 und 120 Minuten. Wenn eine Versuchsperson merklich nicht mit der Aufgabe weiter kam, wurde vom Versuchsleiter geholfen.
Insgesamt wurden 18 Versuchspersonen befragt. Davon gehörten 4 Personen zur Kontrollgruppe.

Während einer Versuchsdurchläufe gab es einige technische Schwierigkeiten aufgrund der nicht-Verfügbarkeit einiger essentieller Internetserver (namentlich der *CiteSeer*-Server), aufgrund derer das Programm nicht wie vorgesehen funktionierte oder extrem langsam war.

7.5.1 Auswertung

Die Auswertung der Evaluation stützt sich hauptsächlich auf die Analyse der Fragebögen. Die Ergebnisse des „Laut-Mitdenken“ Protokolls wurden zusammengefasst und als Stichwörter für Verbesserungen herangezogen.

Zur Auswertung wurde jeder Fragebogenpunkt zunächst einzeln betrachtet. Um mehrere Punkt vergleichen zu können wurden die numerischen Werte des Likert-Schemas ($1-5$) normiert, in dem Sinne, dass die höchste Zustimmung zu einer Aussage auch die höchste Bewertung erhielt. Bewertungen negativ formulierter Aussagen wurden angepasst. Anschließend wurde der Median über alle Fragebögen zu jeder Frage berechnet. Fragen, die nicht oder mit „Nicht Zutreffend“ beantwortet wurden, wurden mit 0 Punkten bewertet und in der Medianberechnung nicht berücksichtigt.
Um eine Bewertung der oben genannten Kriterien (Tab. 7.1) zu erlangen, wurden die dem entsprechenden Kriterium zugehörigen Medianwerte gemittelt.

Bewertung Usability

Auswertung der einzelnen Kriterien:

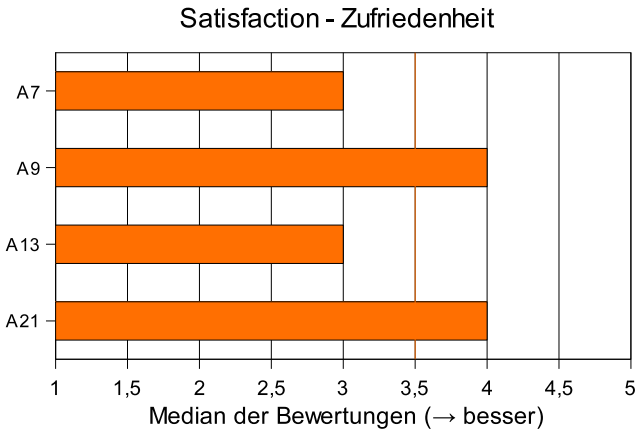


Abbildung 7.1: Auswertung: *Satisfaction – Zufriedenheit*

In Abb. 7.1 sind die Resultat der Zufriedenheitskriterien dargestellt. Der Mittelwert aller Bewertungen liegt bei $= 3,5$. Diese Bewertung lässt sich möglicherweise mit „verhalten Positiv“ beschreiben und ist vermutlich auch einigen technischen Problemen während des Versuchs geschuldet. Einige Versuchsteilnehmer äußerten außerdem Wünsche zu bestimmten Funktionen, die ihre Arbeit mit der Software ihrer Meinung nach vereinfacht hätten.

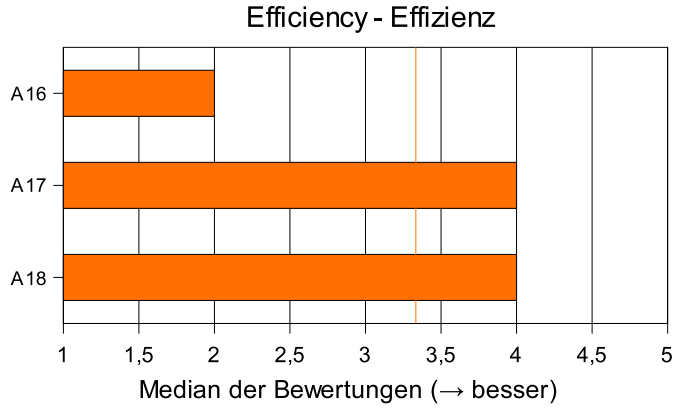


Abbildung 7.2: Auswertung: *Efficiency – Effizienz / Produktivität*

Abbildung 7.2 zeigt die Produktivitäts-Bewertungen der Versuchsteilnehmer. Im Allgemeinen wird damit der Arbeit mit dem Programm eine gute Effizienz nachgewiesen, allerdings war die Geschwindigkeit der Verarbeitung alles andere als Zufriedenstellend (Frage 16 zielte auf die Arbeitsgeschwindigkeit). Daher kann für die Effizienz nur ein mittlerer Wert von $= 3,3$ festgestellt werden.

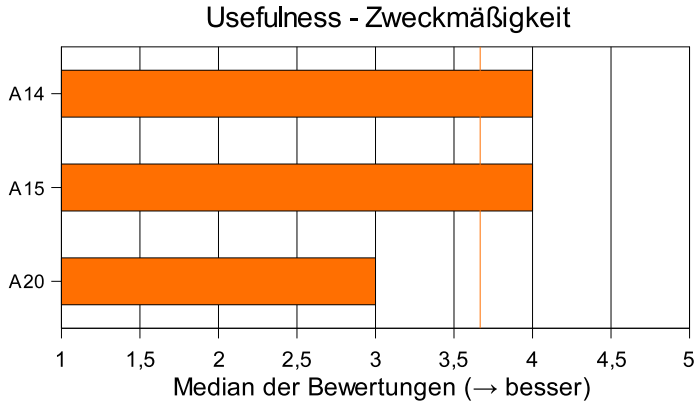


Abbildung 7.3: Auswertung: *Usefulness – Zweckmäßigkeit*

Die Auswertung der Zweckmäßigkeit des Programms in Abb. 7.3 ergab einen mittleren Wert von $= 3,67$. Die beiden zu dieser Gruppe gehörenden Fragen 14 und 15 ergaben dabei ein gutes Bild von dem Programm, insbesondere scheint die Idee der Gruppenbildung und der Suche in wissenschaftlichen Literaturdatenbanken bei den Versuchsteilnehmern positiv aufgegriffen worden zu sein. In Frage 20 war die Frage nach der spezifischen Nützlichkeit des vorliegenden Programms gefragt (die Frage war negativ formuliert). Diese Frage wurde im Median mit einer 3.0 bewertet, was möglicherweise auf Defizite im Programm hindeutet.

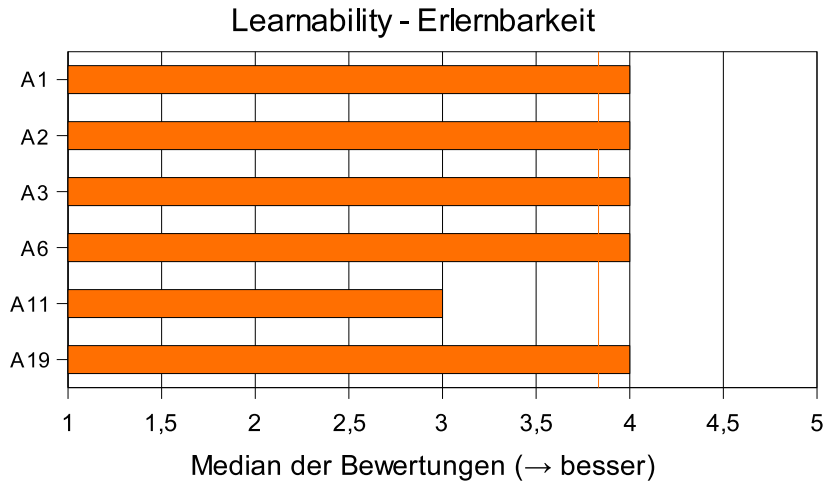


Abbildung 7.4: Auswertung: *Learnability – Erlernbarkeit*

Abbildung 7.4 zeigt die Auswertung der Fragen zur Erlernbarkeit der Programms. Dabei wurden durchweg gute Bewertungen von den Versuchsteilnehmern vergeben mit einem Mittelwert von $= 3,8$. Die Fragen zur Erlernbarkeit enthielten dabei hauptsächlich Fragen zur Bewertung der Oberfläche und des Design. Diese Zustimmung kann daher auch als Zuspruch zum implementierten Ansatz der Datenvisualisierung, bzw. der zugrundeliegende Designidee, gesehen werden.
Die Frage 11 wurde hingegen etwas geringer bewertet, was auf Defizite bei der Hilfestellung und möglicherweise auf Probleme einiger Versuchspersonen bestimmte Hilfestellungen als solche zu erkennen hindeutet.

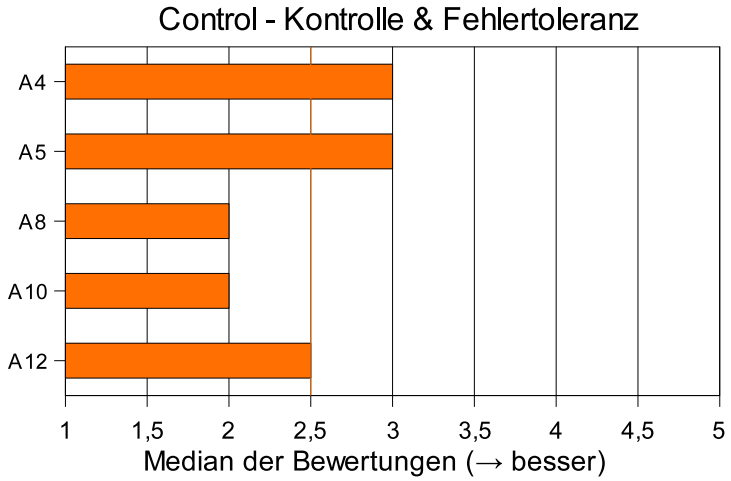


Abbildung 7.5: Auswertung: *Control – Kontrolle / Fehlertoleranz*

In Abbildung 7.5 ist die Auswertung der Kontrolle & Fehlertoleranz Kriterien dargestellt mit einem Mittelwert von $= 2,5$. Dieser eher schlechte Wert unterstützt die oben genannte These zu Frage 11, dass die Hilfefunktionen im Programm von Benutzern nicht gefunden wurden bzw. nicht ausreichend

gekennzeichnet waren.
Zusätzlich haben viele Versuchspersonenen angegeben, dass die Geschwindigkeit und insbesondere die Visualisierung des Beschäftigungszustands unzureichend war. Dies deckt sich auch mit den während des Versuchs aufgezeichneten Kommentaren.

Bewertung der Funktionalität

Die Fragen zur Funktionalität waren unterschiedlich für Experimental- und Kontrollgruppe.
In der Auswertung waren hier insbesondere die Ergebnisse der Experimentalgruppe wichtig für die Einschätzung der Funktionalität der Gruppenbildung und der graphischen Aufbereitung. Durch die Kontrollgruppe konnten vor allem durch die Kommentare während des Versuchs Informationen gesammelt werden, aber auch die Antworten der Fragen zu der Funktionalität und insbesondere zu der gewünschten Funktionalität waren aufschlußreich für eine Beurteilung.

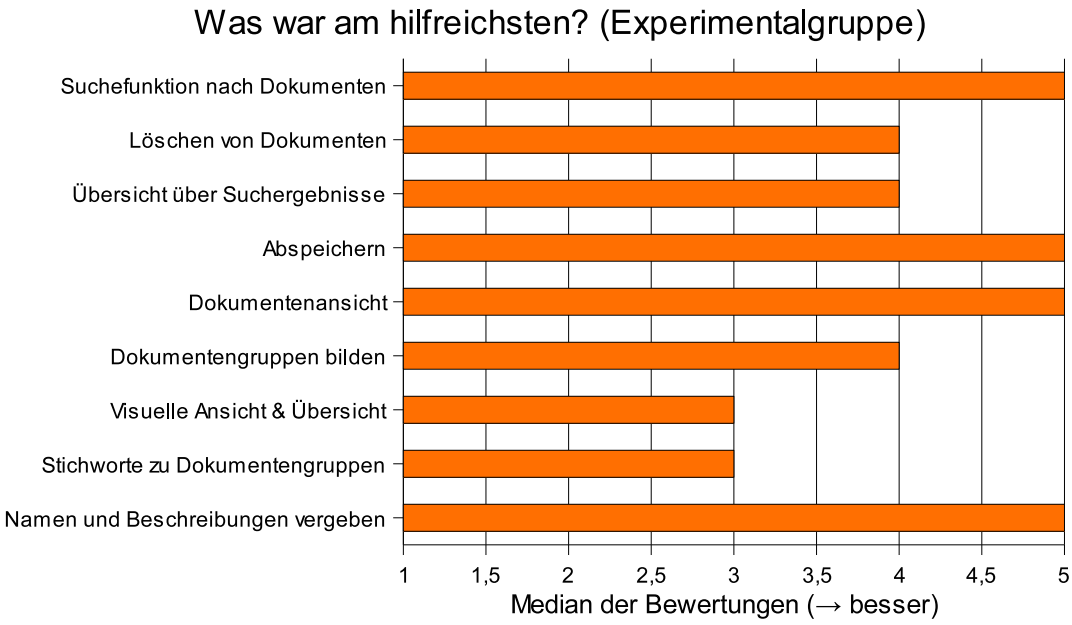


Abbildung 7.6: Auswertung: Die hilfreichsten Funktionen, Experimentalgruppe

Abbildung 7.6 zeigt die Antworten der Versuchsteilnehmer in der Experimentalgruppe zu den Fragen zu den im verwendeten Programm identifizierten Funktionen. Danach ist die überwiegende Einschätzung zu den einzelnen Funktionen positiv. Einschränkungen gab es bei der Bewertung der visuellen Ansicht & Übersicht sowie den generierten Stichwörtern zu Dokumentengruppen. Dies deckt sich zum Teil mit den beobachteten Reaktionen und Kommentaren während des Versuchs.

Die Auswertung der Kontrollgruppe (siehe Abb. 7.7) ergab insgesamt ein etwas verhalteners Bild. Die Bewertungen sind zwar im großen und ganzen noch als gut anzusehen, aber es gibt deutlich geringere Zustimmungswerte zu der Bewertung einzelner Teile. Insbesondere die Dokumentenansicht wurde hier nur mit einer ’3’ bewertet, was unterschiedlich zu den Ergebnissen der Experimentalgruppe ist.

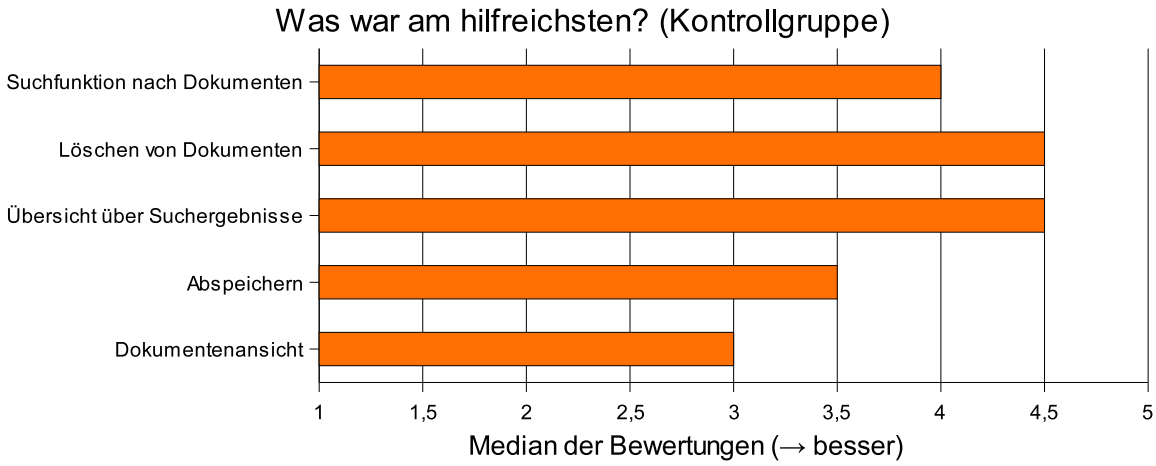


Abbildung 7.7: Auswertung: Die hilfreichsten Funktionen, Kontrollgruppe

Die Auswertung der Fragen nach gewünschten Funktionen in der Software ist in Abbildung 7.8 für die Experimentalgruppe und in Abbildung 7.9 für die Kontrollgruppe zu sehen.
Danach wurden sehr oft Funktionen gewünscht, wie sie in üblichen Literaturdatenbanken zu finden sind, wie bspw. das Sortieren von Dokumenten oder die Ausgabe einer vorformatierten Literaturreferenzangabe.
Den Wunsch nach mehr Dokumenten bzw. anderen Disziplinen hatten deutlich weniger Versuchsteilnehmer, dies mag allerdings auch an der gestellten Aufgabe liegen, die keine Dokumente anderer Fachrichtungen verwendete. Der Austausch von Dokumenten oder Suchergebnissen mit anderen Teilnehmern war ebenfalls von der Mehrzahl der Teilnehmer kein hoch priorisierter Wunsch.
Die Frage nach einer automatischen Kategorisierung von Dokumenten wurde von Teilnehmern beider Gruppen nicht mit einer hohen Priorität beantwortet. Allerdings war dieser Fragepunkt nur unzureichend erklärt, wie diverse Rückfragen während der Versuchsdurchführungen belegten.
Die Frage nach einer möglichen Gruppierung von Dokumenten in der Kontrollgruppe, zielte auf die subjektive Einschätzung der Nützlichkeit einer Gruppierung von Dokumenten ab, ohne eine solche Implementation im Versuch direkt gesehen zu haben. Daher musste dieser Fragepunkt auch teilweise vom Versuchsleiter separat erklärt werden. Alle Versuchsteilnehmer bewerteten daraufhin diese Funktionalität als wünschenswert.

Die Auswertung der von den Teilnehmern produzierten Listen wurde, u. a. wegen zu starker Variationen aufgegeben (siehe auch im Abschnitt 7.6 unten).

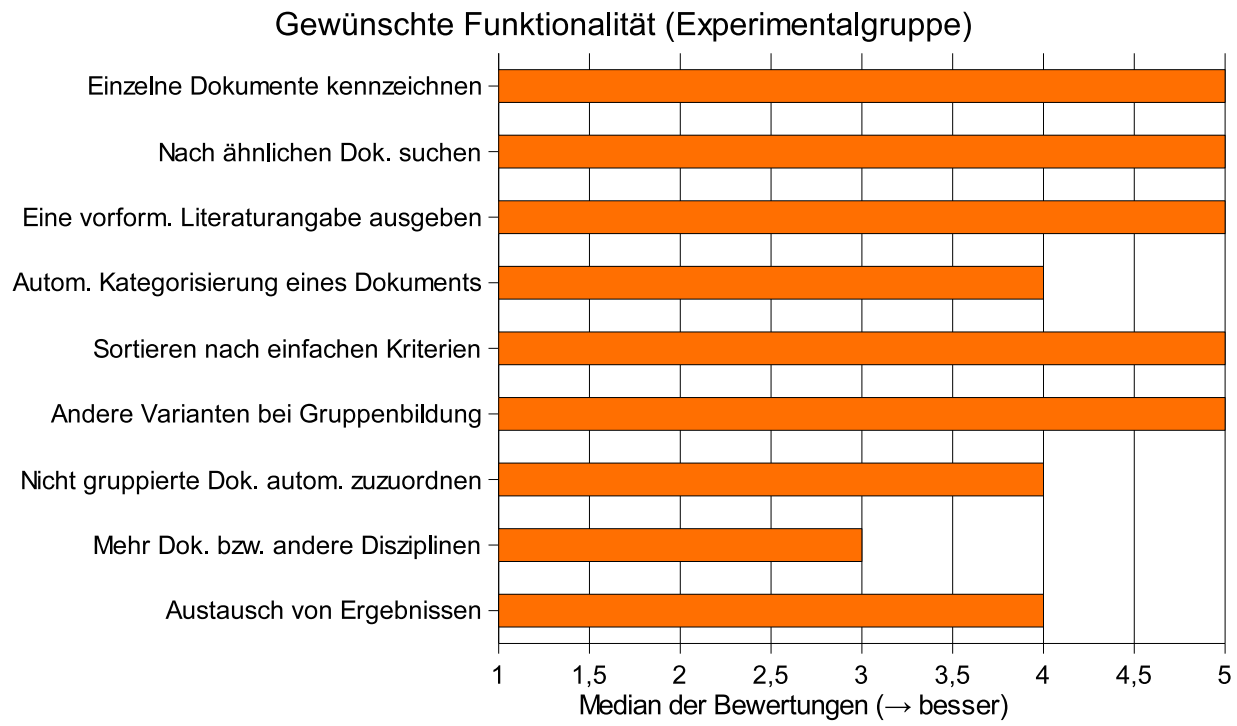


Abbildung 7.8: Auswertung: Gewünschte Funktionen, Experimentalgruppe

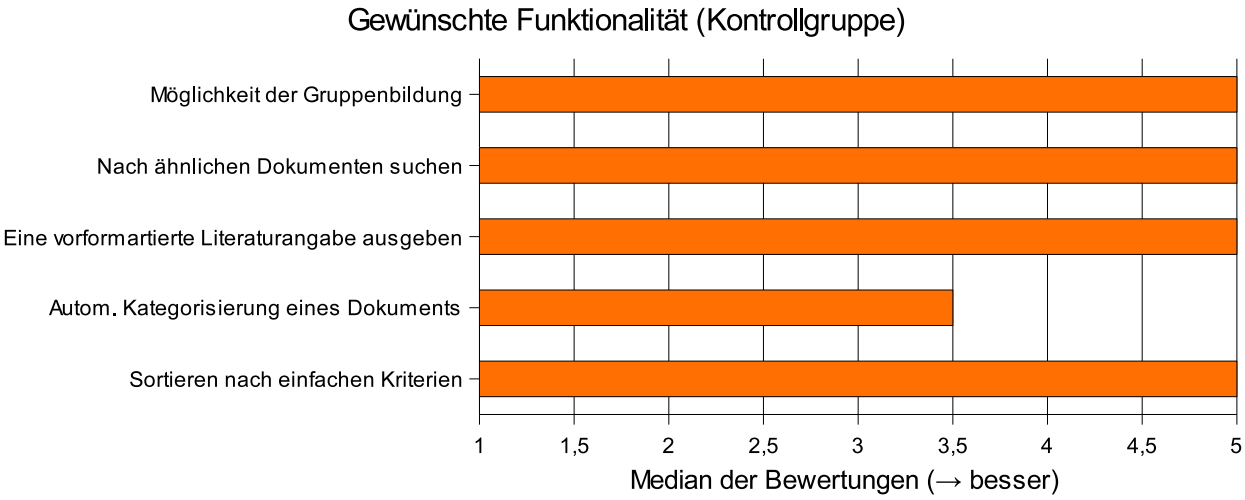


Abbildung 7.9: Auswertung: Gewünschte Funktionen, Kontrollgruppe

Eine Auswertung der Kommentare von Versuchsteilnehmern während des Versuchs, sowie der im Fragebogen vorhandenen Möglichkeiten zur freien Formulierung von Wünschen ergab die folgende Liste von Problemen, Wünschen und Anregungen:

Probleme und Unklarheiten	• Langsamkeit in Bildschirmaufbau und vielen Funktionen, zu lange Wartezeiten • Keine ausreichenden Rückmeldungen beim bei bestimmten Operationen, wie Löschen von Dokumenten • Duplikate in Suchergebnissen[121] • Umgruppierungen umständlich (mehrfach klicken) • Nicht-intuitive GUI-Elemente (Icons, Doppelklick) • „Auge"-Symbol (um Dokumentendetails zu sehen) ist zu umständlich, da mehrfache Klicks nötig • Links zu PDF-Dokumenten funktionieren manchmal nicht • Meldungsboxen mit Informationen über Änderungen manchmal überflüssig • Informationen zu einzelnen Dokumenten sind manchmal sehr spärlich
Wünsche und Anregungen	• Sortieren nach Kriterien wie Autor, Publikationsjahr, Wichtigkeit etc. • Gruppen von Dokumenten manuell erstellen • Leere Gruppen löschen • Klarere Stichwörter und Beschreibungen von Gruppen • Suche nach ähnlichen Dokumenten oder vom selben Autor • Suche nach Begriffen im Suchresultat • Mehrfachauswahl von Dokumenten und Anwendung einer Funktion (Löschen, Clustern etc.) nur auf diese Dokumente • Tags, Label, eigene Beschreibungen zu Dokumenten • Weitere Informationsquellen wie OPAC[122] etc. • Autorennamen invariant darstellen (z. B. „Vorname Nachname") • Visualisierung der Zitations- / Ähnlichkeitsstruktur • Priorisierung / Markierung von einzelnen Dokumenten (im Sinne einer schnell erkennbaren Markierung)

Tabelle 7.2: Probleme, Wünsche und Anregungen

7.6 Resümee

Das Ergebnis der Evaluation zeigt, dass der Ansatz einer inhaltlichen Gruppierung von wissenschaftlichen Dokumenten in der Zielgruppe angenommen wird. Im Bereich der Visualisierung und Aufbereitung liegen allerdings noch Defizite in der vorliegenden Software.
Ein Problem dieser Evaluation waren zweifelsfrei die diversen technischen Probleme, die zu einigen schlechteren Bewertungen geführt haben dürften. Trotzdem konnten nicht nur Bewertungen ermittelt werden, sondern auch viele wertvolle Vorschläge und Ideen, die von Teilnehmern geäußert haben, verwendet werden, um das Programm weiterzuentwickeln.

Die Evaluation, der von den Teilnehmern produzierten Ergebnisse erwies sich aus zwei Gründen als nicht praktikabel. Die Ergebnisse waren nur schwer zu vergleichen, weil zum einen die konkrete Ausprägung (Mindmap, Literaturliste, Liste mit Unterthemen etc.) den Versuchsteilnehmern freigestellt war. Zum anderen variierten die abgegebenen Ergebnisse beträchtlich in ihrer Detailliertheit.
Während einige Teilnehmer sehr viele Begriffe und Verbindungen erfasst haben, gab es andere, die nur sehr wenige Überschriften aufgeschrieben haben. Manche Teilnehmer haben ausschließlich die vom Programm angezeigten Stichwörter zu den Dokumentengruppen verwendet, während andere wiederum sorgfältig alle Dokumente untersucht haben, um dann einen treffenden Begriff zu finden.

Aus diesem Grund kann mit den vorliegenden Ergebnissen nicht zweifelsfrei beantwortet werden, ob das angebotene Programm wirklich zu einer effektiveren und effizienteren Literaturrecherche führt. Die subjektiven Einschätzungen der Versuchsteilnehmer auf dem Fragebogen lassen aber auf eine positive Akzeptanz und Verwendung schließen.

In einer nachfolgenden Evaluation dieses oder eines nachfolgenden Programms, bei dem die von Benutzern produzierten Ergebnisse bewertet werden sollen, sollte daher ein anderer Weg gefunden werden.

[121] Die benutzten Daten von *CiteSeer* haben einen geringen, aber durch Auswertungen dieser Art oft deutliche sichtbaren Anteil von Duplikaten. In der Clusteranalyse stehen diese Dokumente oft direkt hintereinander und führten bei den Versuchspersonen anfänglich oft zu Irritationen.

[122] Ein *Online Public Access Catalog* (OPAC) wird oft von Bibliotheken angeboten für die Recherche nach Monographien.

8 Zusammenfassung und Ausblick

> Dabei genügt es nicht, etwa öffentliche Bibliotheken zu gründen; es muss auch durch Vorträge und Merkhefte die Technik der Benutzung vorhandener Bücherschätze, die gar nicht so einfach ist, den Bildungssuchenden geläufig gemacht werden.
>
> Aus: Wilhelm Ostwald: Zur Biologie des Forschers[123] – Quelle: [Hap04]

8.1 Zusammenfassung

Nach Vorstellung der Idee und Motivation der Arbeit in Kapitel 1 wurden zunächst verschiedene Literaturdatenbanken im Detail in Kapitel 2 vorgestellt sowie eine Eingruppierung in verschiedene Typen von Literaturdatenbanken vorgenommen. Eine kurze Vorstellung der dieser Arbeit vorangegangenen relevanten Projekte der Arbeitsgruppe folgte in Abschnitt 2.3.
In Kapitel 3 wurden die theoretischen Grundlagen besprochen, auf denen u. a. einige der vorgestellten Literaturdatenbanken beruhen (Abschnitt 3.1.1). Abschnitt 3.2 widmete sich den Möglichkeiten, die Gebrauchstauglichkeit einer Software zu messen.
Um ein Programm zu entwickeln, das den in Abschnitt 1.4 dargestellten Bedürfnissen gerecht wird, wurden in Kapitel 4 die wichtigsten Anforderungen an die Software erstellt.
In Kapitel 5 folgte dann eine technisch orientierte Konkretisierung dieser Anforderungen mit einer Auswahl von Softwarekomponenten. Zusätzlich wurden die Auswahl der Daten einer Literaturdatenbank, ihre Anpassung und ihr Import in eine eigene Datenbank beschrieben.

Das aus diesen Anforderungen und Softwarekomponenten entwickelte Programm ist in Kapitel 6 präsentiert, eine Vorstellung der Arbeitsweise einiger der Algorithmen aus Abschnitt 3.1.1 anhand eines Testdatensatzes in Abschnitt 6.3. Im Anschluss des Kapitels (Abschnitt 6.3.5) sind einige Fragen aus der initialen Fragestellung des Projekts (Kapitel 1.3) wieder aufgegriffen und beantwortet.

Aufbauend auf den in Abschnitt 3.2 vorgestellten Möglichkeiten, die Gebrauchstauglichkeit einer Software zu testen, wurde eine Nutzerstudie durchgeführt, die in Kapitel 7 beschrieben ist. Diese Evaluation wurde mit einer nicht-fertigen Programmversion gemacht, um weitere Ideen und konzeptionelle Probleme identifizieren zu können. Einige der Ideen konnten daher zeitnah in die Software integriert werden.

Die Ergebnisse der Evaluation, vorgestellt in Abschnitt 7.6, lassen den Schluß zu, dass der verfolgte Ansatz einer inhaltlichen Gruppierung für die Zielgruppe hilfreich sein kann. Der realisierte Ansatz einer Benutzerschnittstelle wurde von der Zielgruppe positiv aufgenommen, bietet aber Raum für Verbesserungen.

[123] Wilhelm Ostwald, Zur Biologie des Forschers : Vortrag, gehalten während der 350. Jahresfeier der Universität Genf, in: Actes du Jubilé de 1909 / Université de Genève, Librairie Georg & Cie / Genève, 1910, S. 114-121, hier S. 118. Englische Fassung: Biology of the savant : a study in the psychology of personality, in: Scientific American, 9. September 1911, Supplement No. 1862: S. 169-171. Französische Fassung: A propos de la biologie du savant, in: Bibliotheque universelle et revue suisse, 1910, Annee 115, 60, No. 178: 157-168.

8.2 Ausblick

Es sind in verschiedenen Bereichen des Projekts Erweiterungen und Verbesserungen denkbar. Dazu gehören die benutzten Daten, die eingesetzten Algorithmen im Programm, die Bedienschnittstelle bzw. die Visualisierung für Nutzer und zu guter Letzt auch die Evaluation.

Daten

Die benutzte Literaturdatenbank *CiteSeer* hat sich in mehrerer Hinsicht als ungünstig erwiesen. Problematisch ist zum einen, dass *CiteSeer* fast nur Dokumente eines sehr begrenzten Zeitraums (der Jahre 1994–2004) enthält. Zum anderen lässt auch die Qualität der Daten zu wünschen übrig. Insbesondere der nicht unwesentliche Anteil an Duplikaten, falsch zugeordneten Feldern und die sehr unvollständigen Referenzlisten erhöhen den Wunsch nach anderen Datenquellen. Fehler dieser Art führen nicht nur bei Benutzern des Systems zu Irritationen bei der Anzeige, sondern verfälschen auch die Analysen, die aufgrund der hohen Ähnlichkeitswerte von Duplikaten zu anderen Ergebnissen kommen können.

Ein interessanter Ansatz wäre es, die Daten von *Citebase* (siehe Seite 16) zu verwenden oder sogar auf Basis der Analyse einer der in Abschnitt 2.2.1 vorgestellten Literaturdatenbanken, wie die *ScientificCommons* Datenbank, arbeiten zu können.

Algorithmen

Die verwendeten Algorithmen und Analysemethoden haben sich sowohl in der Nutzerstudie als auch in der detaillierten Betrachtung einzelner Analyseschritte als praktikabel erwiesen.
Dennoch gibt es Raum für Verbesserungen, z. B. eine genauere Untersuchung und Anwendung der in Abschnitt 3.1.1.1 vorgestellten Varianten der Verfahren für die Kozitationsanalyse. Es ist aber davon auszugehen, dass mit einer anderen Datenbasis (und einer deutlich höheren Anzahl von Referenzen pro Dokument) die bisher implementierten Algorithmen zu zufriedenstellenden Ergebnissen führen. Interessant könnte die Implementation von Algorithmen sein, die mit weiteren Daten arbeiten, wie beispielsweise die in der Literaturdatenbank *Citebase* verwendeten Downloadzahlen oder mit einem Ähnlichkeitsmaß auf Basis von Suchmaschinen, wie es in [VB07] vorgeschlagen wird.

Weitere Verbesserungen betreffen das Zusammenführen der verschiedenen Ergebnisse aus bibliographischer und Textanalyse. Eine Implementation von „Fisher's inverse chi-square method“, wie in [Jan07] vorgeschlagen, könnte sich positiv auswirken.

Viel Raum für Veränderungen lassen die Methoden der Generierung von aussagekräftigen Beschreibungen für Dokumentengruppen. In der Evaluation hat sich die implementierte Variante als nicht geeignet erwiesen, so dass eine andere Methode gesucht werden sollte. Die in Abschnitt 3.1.4 angesprochene Analyse auf N-gram Basis verspricht hier möglicherweise Verbesserungen.

Ein Punkt, der im bisherigen Programm bisher nur am Rande betrachtet wurde, ist das Ranking von Dokumenten innerhalb einer Dokumentengruppe. In der aktuellen Software werden Dokumente nach starren Kriterien wie Publikationsjahr sortiert. Aussagekräftiger für einen potentiellen Benutzer wäre sicher ein anderes Maß, wie etwa das zeitbezogene Zitationsmaß, das in [BH07] vorgeschlagen wird. Eine andere Möglichkeit wäre die Implementation von *PageRank* oder *HITS* für ein Ranking der Dokumente.

Um die Anforderungen an Reaktionszeit und Geschwindigkeit zu erfüllen, wurden bestimmte Werte wie die Anzahl der Zitationen pro Dokument vorberechnet. Für ein größeres System mit vielen Benutzern, sollten weitere Ähnlichkeitswerte vorausberechnet werden. Dies wurde in der bisherigen Version aus Ressourcengründen (z. B. Speicherkapazitäten) ausgelassen.

Benutzerschnittstelle

Wie die Ergebnisse der Evaluation gezeigt haben, kann mit der bisherigen Form der Benutzerschnittstelle Nutzern ausreichend demonstriert werden, welche Möglichkeiten sich durch die Dokumentenclusterung für die Literaturrecherche ergeben. Allerdings ist sie kein vollwertiger Ersatz für ein komplexes System zur Literatursuche und -verwaltung.
In der Evaluation wurde der Wunsch nach einer umfangreicheren Visualisierung geäußert. Eine deutlichere Darstellung der Zitations- und Ähnlichkeitsstruktur ist damit naheliegend.
Darüberhinaus sind aus der Evaluation viele Verbesserungs- und Erweiterungsvorschläge entstanden, wie eine Suche nach ähnlichen Dokumenten oder nach Dokumenten vom selben Autor.

Eine weitere Erweiterungsmöglichkeit ist in der Einbeziehung von Nutzerinteraktionen in die Bewertungs- und Stichwortgenerierung innerhalb des Prgrogramms zu sehen. Eine interessante Variante wäre ebenfalls die Möglichkeit bearbeitete Suchergebnisse innerhalb des Programms anderen Nutzern zur Verfügung zu stellen.

Evaluation

Die durchgeführte Evaluation war durchaus aussagekräftig, was die Beurteilung der Verwendbarkeit anging. Die Antwort auf die Frage, ob die Gruppierung von Dokumenten in dieser Form hilfreich für die Zielgruppe ist, kann nur tendenziell positiv beurteilt werden. Eine größer angelegte Nutzerstudie könnte hier weitere Aufklärung bringen.
Alternativ könnte eine entsprechende Funktionalität in ein bestehendes System integriert werden, um auf Basis der Nutzungsdaten eine klarere Aussage zu erlangen.

Anhang

Anhang A

Rechenbeispiel LSA & Textähnlichkeit

Beispiel für eine LSA Operation mit 5 Dokumenten und je 10 Termen. Ausgewählt wurden 5 Dokumente aus der Datenbank (IDs: `708729, 296243, 534093, 24040, 357930`), die durch das bibliografische Verfahren schon als „ähnlich“ klassifiziert worden sind. Zur Übersichtlichkeit sind hier nur jeweils die ersten 10 Terme von jedem Dokument angegeben und und es wurden auch nur die 'rohen' Häufigkeitswerte der Terme notiert.
Sei $\vec{d} \in [\vec{d}_1, \vec{d}_2, ..., \vec{d}_n]$ und $n = 5$ mit: (hier mit den entsprechenden Termen dargestellt)

$$\vec{d}_1 = \left[\begin{array}{l|l} support & 1.0 \\ outline & 1.0 \\ actual & 1.0 \\ population & 2.0 \\ nature & 1.0 \\ adaptive & 2.0 \\ intelligent & 2.0 \\ image & 1.0 \\ complement & 2.0 \\ mine & 3.0 \end{array}\right] \quad \vec{d}_2 = \left[\begin{array}{l|l} pertain & 1.0 \\ profile & 1.0 \\ content-based & 1.0 \\ rate & 1.0 \\ technique & 2.0 \\ web & 5.0 \\ combine & 2.0 \\ distinguish & 1.0 \\ product & 1.0 \\ recent & 1.0 \end{array}\right] \quad \vec{d}_3 = \left[\begin{array}{l|l} pertain & 1.0 \\ profile & 1.0 \\ content-based & 1.0 \\ rate & 1.0 \\ technique & 1.0 \\ web & 5.0 \\ scalability & 1.0 \\ recent & 1.0 \\ mechanism & 1.0 \\ richer & 1.0 \end{array}\right]$$

$$\vec{d}_4 = \left[\begin{array}{l|l} statistical & 1.0 \\ architecture & 1.0 \\ technique & 1.0 \\ web & 4.0 \\ activity & 1.0 \\ workbench & 1.0 \\ step & 1.0 \\ tool & 2.0 \\ webclass & 3.0 \\ conclusion & 1.0 \end{array}\right] \quad \vec{d}_5 = \left[\begin{array}{l|l} status & 1.0 \\ compare & 1.0 \\ dynamic & 1.0 \\ architecture & 1.0 \\ describe & 3.0 \\ propose & 2.0 \\ tak & 1.0 \\ technique & 6.0 \\ web & 9.0 \\ combine & 1.0 \end{array}\right]$$

Aus diesen Dokument-Vectoren kann nun die Ausgangsmatrix **A** konstruiert werden. Dazu werden alle eindeutigen Terme gesammelt und die Häufigkeiten zugeordnet.

$$\mathbf{A} = \left[\vec{d}_1, \vec{d}_2, \ldots, \vec{d}_n\right]$$

Dabei entsteht in den allermeisten Fällen eine relativ spärlich besetzte Matrix, die sich effizient speichern lässt.

Die Vector-Space-Matrix **A** mit insgesamt 37 Termen: (mit den zur entsprechenden Zeile gehörenden Termen)

$$\mathbf{A} = \left[\begin{array}{l|llll}
support & 1.0000 & & & & \\
outline & 1.0000 & & & & \\
actual & 1.0000 & & & & \\
population & 2.0000 & & & & \\
nature & 1.0000 & & & & \\
adaptive & 2.0000 & & & & \\
intelligent & 2.0000 & & & & \\
image & 1.0000 & & & & \\
complement & 2.0000 & & & & \\
mine & 3.0000 & & & & \\
pertain & & 1.0000 & 1.0000 & & \\
profile & & 1.0000 & 1.0000 & & \\
content-based & & 1.0000 & 1.0000 & & \\
rate & & 1.0000 & 1.0000 & & \\
technique & & 2.0000 & 1.0000 & 1.0000 & 6.0000 \\
web & & 5.0000 & 5.0000 & 4.0000 & 9.0000 \\
combine & & 2.0000 & & & 1.0000 \\
distinguish & & 1.0000 & & & \\
product & & 1.0000 & & & \\
recent & & 1.0000 & 1.0000 & & \\
scalability & & & 1.0000 & & \\
mechanism & & & 1.0000 & & \\
richer & & & 1.0000 & & \\
statistical & & & & 1.0000 & \\
architecture & & & & 1.0000 & 1.0000 \\
activity & & & & 1.0000 & \\
workbench & & & & 1.0000 & \\
step & & & & 1.0000 & \\
tool & & & & 2.0000 & \\
webclass & & & & 3.0000 & \\
conclusion & & & & 1.0000 & \\
status & & & & & 1.0000 \\
compare & & & & & 1.0000 \\
dynamic & & & & & 1.0000 \\
describe & & & & & 3.0000 \\
propose & & & & & 2.0000 \\
tak & & & & & 1.0000
\end{array}\right]$$

Die *SVD* Operation spaltet diese Matrix nun auf in die drei Bestandteile:

$$\mathbf{A} = \mathbf{U}\,\Sigma\,\mathbf{V}^T$$

hier mit den Dimensionen:

A : (37x5)
U : (37x5)
Σ : (5x5)
$\mathbf{V}^T$: (5x5)

In der Diagonalmatrix Σ sind die Singulärwerte der Matrix zu finden:

$$\Sigma = \text{diag}\left[\ 14.2797, 5.4773, 4.5432, 4.0952, 2.1634\ \right]$$

sowie den beiden Matrizen $\mathbf{U}$ und $\mathbf{V}^T$. Dabei wird $\mathbf{U}$ in der *LSA* als Termmatrix bezeichnet und $\mathbf{V}^T$ als Dokumentenmatrix. Diese beiden Matrizen entstehen aus den Eigenvektoren der Ausgangsmatrix und sind im allgemeinen vollbesetzte Matrizen, für die außerdem die unitäre Eigenschaft gilt: $\mathbf{U}^T \cdot \mathbf{U} = \mathbf{I}$.
Ein *Scree-Plot* der singulären Werte ist in Abbildung A.1 zu sehen.

Mit der LSA wird nun der Rank der Singulärwertmatrix Σ reduziert, was einer Informationsreduzierung entspricht. Technisch werden Eigenwerte verworfen bzw. auf 0 gesetzt. Dazu müssen natürlich auch die Matrizen $\mathbf{U}$ und $\mathbf{V}^T$ angepasst werden.
Im Beispiel wurde die Dimension des Diagonalmatrix Σ auf $k = 3$ reduziert, was zur Matrix $\mathbf{A}_{k=3}$ führt. Um rechnerische Ungenauigkeiten und nur geringe Abhängigkeiten auszublenden, werden Werte in der Matrix auf 0 gesetzt, wenn sie kleiner einer bestimmten Schranke ε sind.
Im Beispiel ist $\varepsilon = 0.1$ gewählt.

$$\mathbf{A_{k=3}} = \begin{bmatrix}
1.0000 & & & & \\
1.0000 & & & & \\
1.0000 & & & & \\
2.0000 & & & & \\
1.0000 & & & & \\
2.0000 & & & & \\
2.0000 & & & & \\
1.0000 & & & & \\
2.0000 & & & & \\
3.0000 & & & & \\
 & 0.3566 & 0.3748 & 0.5633 & 0.3827 \\
 & 0.3566 & 0.3748 & 0.5633 & 0.3827 \\
 & 0.3566 & 0.3748 & 0.5633 & 0.3827 \\
 & 0.3566 & 0.3748 & 0.5633 & 0.3827 \\
 & 2.2109 & 1.7363 & 0.5539 & 5.7390 \\
 & 4.8944 & 4.4595 & 4.3079 & 9.1745 \\
 & 0.5956 & 0.5003 & 0.3120 & 1.3596 \\
 & 0.1780 & 0.1786 & 0.2388 & 0.2396 \\
 & 0.1780 & 0.1786 & 0.2388 & 0.2396 \\
 & 0.3566 & 0.3748 & 0.5633 & 0.3827 \\
 & 0.1786 & 0.1962 & 0.3245 & 0.1431 \\
 & 0.1786 & 0.1962 & 0.3245 & 0.1431 \\
 & 0.1786 & 0.1962 & 0.3245 & 0.1431 \\
 & 0.2388 & 0.3245 & 0.7454 & \\
 & 0.4784 & 0.4676 & 0.5798 & 0.7148 \\
 & 0.2388 & 0.3245 & 0.7454 & \\
 & 0.2388 & 0.3245 & 0.7454 & \\
 & 0.2388 & 0.3245 & 0.7454 & \\
 & 0.4776 & 0.6490 & 1.4909 & \\
 & 0.7164 & 0.9736 & 2.2363 & \\
 & 0.2388 & 0.3245 & 0.7454 & \\
 & 0.2396 & 0.1431 & & 0.8804 \\
 & 0.2396 & 0.1431 & & 0.8804 \\
 & 0.2396 & 0.1431 & & 0.8804 \\
 & 0.7187 & 0.4292 & & 2.6412 \\
 & 0.4792 & 0.2862 & & 1.7608 \\
 & 0.2396 & 0.1431 & & 0.8804
\end{bmatrix}$$

Das Verfahren lässt sich nun z. B. noch folgendermassen Anpassen um eine Verbesserung zu erzielen:

- Auslassen von Termen, die nur in einem Dokument vorkommen. Diese Terme tragen keinen Beitrag zur Ähnlichkeit des enthaltenen Dokumentes zu anderen bei.
  Der Vorteil ist außerdem, dass die Matrix deutlich kleiner wird und so rechentechnisch leichter zu verarbeiten ist.
- Statt einzeln vorkommenden Termen können auch Dokumente ausgelassen werden, deren Terme nicht in anderen Dokumenten vorkommen. (Dies macht in der zu entwickelnden Anwendung keinen Sinn, da ja Ähnlichkeiten zwischen allen Dokumenten ermittelt werden sollen.)
- Statt den rohen Termfrequenz-Werten können gewichtete Werte verwendet werden, die zum Beispiel die „Wichtigkeit“ eines Terms in der Gesamtdokumentenmenge wiederspiegeln.
  Hier hat sich der *TF-IDF* Wert als günstig erwiesen.

Beispiel: Matrix **A** mit reduzierter Zeilenanzahl:

$$\tilde{\mathbf{A}} = \left[\begin{array}{l|cccc} \mathit{pertain} & 1.00 & 1.00 & & \\ \mathit{profile} & 1.00 & 1.00 & & \\ \mathit{content-based} & 1.00 & 1.00 & & \\ \mathit{rate} & 1.00 & 1.00 & & \\ \mathit{technique} & 2.00 & 1.00 & 1.00 & 6.00 \\ \mathit{web} & 5.00 & 5.00 & 4.00 & 9.00 \\ \mathit{combine} & 2.00 & & & 1.00 \\ \mathit{recent} & 1.00 & 1.00 & & \\ \mathit{architecture} & & & 1.00 & 1.00 \end{array}\right] \quad \tilde{\mathbf{A}}_{\mathbf{k=3}} = \left[\begin{array}{cccc} 1.0528 & 0.9301 & 0.0928 & \\ 1.0528 & 0.9301 & 0.0928 & \\ 1.0528 & 0.9301 & 0.0928 & \\ 1.0528 & 0.9301 & 0.0928 & \\ 2.1571 & 0.7918 & 1.2761 & 5.9110 \\ 4.9509 & 5.0651 & 3.9136 & 9.0278 \\ 1.7677 & 0.3079 & & 1.1317 \\ 1.0528 & 0.9301 & 0.0928 & \\ & 0.2559 & 0.6605 & 1.1095 \end{array}\right]$$

Daraus berechnete Ähnlichkeitsmatrix $\mathbf{S}_{text}$:

$$\mathbf{S} = \begin{bmatrix} 1.0000 & & & & \\ & 1.0000 & 0.8677 & 0.5798 & 0.7999 \\ & 0.8677 & 1.0000 & 0.6002 & 0.7500 \\ & 0.5798 & 0.6002 & 1.0000 & 0.6145 \\ & 0.7999 & 0.7500 & 0.6145 & 1.0000 \end{bmatrix}$$

Da das 1. Dokument keine mit einem anderen Dokument gemeinsamen Terme hat, ist die Ähnlichkeit zu diesem auch immer 0 (außer zu sich selbst).

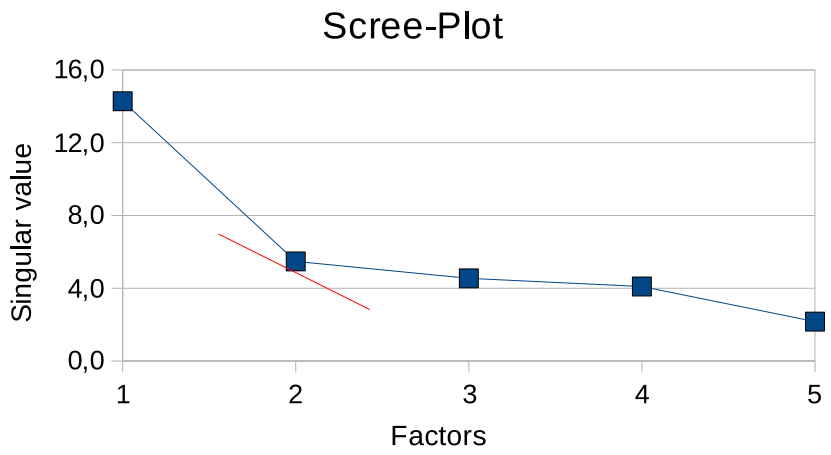


Abbildung A.1: Scree-Plot der singulären Werte

Nach dem Ellenbogenkriterium wäre die optimale Anzahl von singulären Werten als 2 anzugeben, nach dem Kaiser-Guttmann Kriterium wären alle 5 Werte (da > 1) zu berücksichtigen.

Anhang B

DTD für Export & Importformat

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    DTD for CiteseerCluster document-repository.
    @author Sebastian Kolbe &lt;SebastianKolbe at gmx dot net&gt;
    @version $Id: docRepository_1.0.dtd 156 2008-01-05 18:07:40Z sebi $
-->
<!ELEMENT repository ( collection+ ) >

<!ELEMENT collection ( identifier, (keywords)?, (desc)?, createdate, moddate,
  sublist+ ) >

<!ELEMENT sublist ( name, (description)?, (generatedDescription)?, (document)* ) >
<!ATTLIST sublist default NMTOKEN #IMPLIED >
<!ATTLIST sublist unreferred NMTOKEN #IMPLIED >

<!ELEMENT document ( abstract | bibtex | access | author | (authorEntity)*
  | chapter | description | doi | edition | editor | import | isbn | issn
  | language | link | keywords | medium | number | organization | organizationaddress
  | pages | price | publisher | publisheraddress | subject | subtitle | title
  | translator | url | userSummary | userTitle | userURL | version | volume
  | worktitle | year | label | etiquette )* >

<!ATTLIST document instance NMTOKEN #REQUIRED >
<!ATTLIST document key NMTOKEN #REQUIRED >
<!ATTLIST document type NMTOKEN #IMPLIED >

<!ELEMENT abbreviation ( #PCDATA ) >
<!ELEMENT abstract ( #PCDATA ) >
<!ELEMENT access ( #PCDATA ) >

<!ELEMENT affiliation ( #PCDATA ) >
<!ELEMENT author ( #PCDATA ) >

<!ELEMENT authorEntity ( fullname, ( firstname | lastname | abbreviation | note
  | urllink | affiliation )* ) >
<!ATTLIST authorEntity id NMTOKEN #REQUIRED >

<!ELEMENT createdate ( #PCDATA ) >
<!ELEMENT chapter ( #PCDATA ) >
<!ELEMENT desc ( #PCDATA ) >
<!ELEMENT description ( #PCDATA ) >

<!ELEMENT doi ( #PCDATA ) >
<!ELEMENT edition ( #PCDATA ) >
```

```
<!ELEMENT editor ( #PCDATA ) >
<!ELEMENT fullname ( #PCDATA ) >
<!ELEMENT firstname ( #PCDATA ) >

<!ELEMENT generatedDescription ( #PCDATA ) >
<!ATTLIST generatedDescription omit NMTOKEN #REQUIRED >

<!ELEMENT identifier ( #PCDATA ) >

<!ELEMENT import EMPTY >
<!ATTLIST import id NMTOKEN #REQUIRED >
<!ATTLIST import source NMTOKEN #REQUIRED >

<!ELEMENT isbn ( #PCDATA ) >
<!ELEMENT issn ( #PCDATA ) >
<!ELEMENT keywords ( #PCDATA ) >
<!ELEMENT language ( #PCDATA ) >

<!ELEMENT lastname ( #PCDATA ) >
<!ELEMENT link ( #PCDATA ) >
<!ELEMENT medium ( #PCDATA ) >
<!ELEMENT moddate ( #PCDATA ) >

<!ELEMENT name ( #PCDATA ) >
<!ELEMENT note ( #PCDATA ) >
<!ELEMENT number ( #PCDATA ) >
<!ELEMENT organization ( #PCDATA ) >

<!ELEMENT organizationaddress ( #PCDATA ) >
<!ELEMENT pages ( #PCDATA ) >
<!ELEMENT price ( #PCDATA ) >
<!ELEMENT publisher ( #PCDATA ) >

<!ELEMENT publisheraddress ( #PCDATA ) >
<!ELEMENT subject ( #PCDATA ) >
<!ELEMENT subtitle ( #PCDATA ) >
<!ELEMENT title ( #PCDATA ) >

<!ELEMENT translator ( #PCDATA ) >
<!ELEMENT url ( #PCDATA ) >
<!ELEMENT urllink ( #PCDATA ) >
<!ELEMENT userSummary ( #PCDATA ) >

<!ELEMENT userTitle ( #PCDATA ) >
<!ELEMENT userURL ( #PCDATA ) >
<!ELEMENT version ( #PCDATA ) >
<!ELEMENT volume ( #PCDATA ) >

<!ELEMENT worktitle ( #PCDATA ) >
<!ELEMENT year ( #PCDATA ) >
<!ELEMENT label ( #PCDATA ) >
<!ELEMENT etiquette ( #PCDATA ) >
<!ELEMENT bibtex ( #PCDATA ) >
```

Anhang C

Abfrage von Metadaten über OAI


```
− <OAI-PMH xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/
  http://www.openarchives.org/OAI/2.0/OAI-PMH.xsd">
    <responseDate>2008-02-17T16:39:05Z</responseDate>
    <request identifier="oai:arXiv.org:adap-org/9710001" verb="GetRecord"
    metadataPrefix="oai_dc"> http://www.citebase.org/oai </request>
  − <GetRecord>
    − <record>
      − <header>
          <identifier>oai:arXiv.org:adap-org/9710001</identifier>
          <datestamp>2007-08-16</datestamp>
        </header>
      − <metadata>
        − <oai_dc:dc xsi:schemaLocation="http://www.openarchives.org/OAI/2.0/oai_dc/
          http://www.openarchives.org/OAI/2.0/oai_dc.xsd">
            <dc:creator>Brandt, Paolo Sibani Michael</dc:creator>
            <dc:creator>Alstroem, Preben</dc:creator>
          − <dc:subject>
              Nonlinear Sciences - Adaptation and Self-Organizing Systems
            </dc:subject>
            <dc:subject>Quantitative Biology</dc:subject>
            <dc:subject>Adaptation, Noise, and Self-Organizing Systems</dc:subject>
            <dc:date>1997-10-06</dc:date>
            <dc:description>Comment: 30 pages 9 figures LaTeX</dc:description>
          − <dc:description>
              Macroevolution is considered as a problem of stochastic dynamics in a system with
              many competing agents. Evolutionary events (speciations and extinctions) are triggered
              by fitness records found by random exploration of the agents' fitness landscapes. As a
              consequence, the average fitness in the system increases logarithmically with time,
              while the rate of extinction steadily decreases. This dynamics is studied by numerical
              simulations and, in a simpler mean field version, analytically. We also study the effect of
              externally added `mass' extinctions. The predictions for various quantities of
              paleontological interest (life-time distributions, distribution of event sizes and behavior
              of the rate of extinction) are robust and in good agreement with available data. Brief
              version of parts of this work have been published as Letters. (PRL 75, 2055, (1995) and
              PRL, 79, 1413, (1997))
            </dc:description>
            <dc:identifier>http://www.citebase.org/abstract?id=2618</dc:identifier>
          − <dc:identifier>
              http://www.citebase.org/fulltext?format=application%2Fpdf&id=2618
            </dc:identifier>
            <dc:relation>http://arxiv.org/abs/adap-org/9710001</dc:relation>
            <dc:type>text</dc:type>
          − <dc:title>
              Evolution and extinction dynamics in rugged fitness landscapes
            </dc:title>
          </oai_dc:dc>
        </metadata>
      </record>
    </GetRecord>
  </OAI-PMH>
```

Anhang D

Aufgabenbeschreibung für Evaluation

Es gibt zwei unterschiedliche Versionen der Aufgabenbeschreibung und des Fragebogens; je eine für die Experimentalgruppe und eine Version für die Kontrollgruppe.
Die wurden zusammen mit Prof. Berendt und Beate Krause entwickelt.

D.1 Experimentalgruppe

D.2 Kontrollgruppe

Aufgabenbeschreibung und Fragebogen für die Kontrollgruppe.

Anhang E

Inhalt DVD

Die beigefügte DVD beinhaltet eine PDF-Version dieser Arbeit, das entwickelte Programm (sowohl in Quelltexten, als auch als WAR-Archiv), einen Datenbankauszug mit den *CiteSeer*-Daten und einen Teil der verwendeten Literatur, sofern diese elektronisch (als PDF, o. ä.) verfügbar war.
Die Verzeichnisstruktur:

```
|
thesis/     -Die elektronische Version (PDF) dieser Arbeit
evaluation/ -Material für und aus der Evaluation (Fragebogen, Protokolle, Auswer
literatur/  -Verwendete Quellen, sofern verfügbar
data/       -Datenbankdump mit Tabellenstruktur und Datensätzen
software/   -Die entwickelte Software als WAR-Archiv und den Quelldateien
progdoc/    -Programmdokumentation (JavaDoc)
```

Glossar

Abstract

Englische Bezeichnung für eine (Kurz-)Zusammenfassung eines wissenschaftlichen Dokumenten. Der Name *Abstract* hat sich eingebürgert.

Applet

Ein Applet ist ein (zumeist kleines) Programm, welches als eigenständiges Programm innerhalb eines anderen Programms (z. B. Webbrowser) gestartet wird. Der Name leitet sich aus dem (engl.) Wort *application* ab. Oft verwendet, um grafische Ausgaben oder interaktive Elemente in eine Webseite zu integrieren. Der Begriff *Applet* wird daher sehr häufig für in der Programmiersprache Java geschriebene Programme verwendet.

Crawler

Bezeichnet hier ein Computerprogramm, dass selbsttätig im Internet sucht und Webseiten nach Informationen analysiert. Dabei findet der *Crawler* neue Seiten über die Linkstruktur in den abgesuchten Seiten.

MVC

Das *Model-View-Controller*-Architekturmuster, gelegentlich auch als Programmierkonzept bezeichnet, beschreibt die Trennung zwischen Darstellung/Präsentation (*View*), der Daten- und Verarbeitungssschicht (*Model*) und der Programmsteuerungsschicht (*Controller*).
Ziel und Zweck dieser Trennung ist eine flexibles Programmdesign, Übersichtlichkeit und Ordnung z. B. auch für spätere Änderungen am Programm. MVC beschreibt zunächst aber nur eine sehr grobe Aufteilung des Programmdesigns, zumeist sind weitere Unterteilungen nötig wie z. B. Eine Schicht für Datenzugriffe, Verarbeitungslogik / Businesslogik usw.

OAI

Open Archives Initiative – Initiative zur Förderung der Zusammenarbeit digitaler Archive. Ist hauptsächlich bekannt durch das gleichnamige Protokoll, dass zur automatischen Abfrage und Suche nach Dokumenten in teilnehmenden Archiven dienen kann.

Paper

Wissenschaftliche Arbeiten werden oft als *Paper* bezeichnet. Sehr oft bezieht sich diese Bezeichnung auf Veröffentlichungen, die nach dem sog. *Peer Review* Verfahren bewertet werden, bei dem andere Wissenschaftler die Arbeit unabhängig bewerten.
Im Idealfall werden dadurch für das jeweilige Fachgebiet relevante Arbeiten veröffentlicht, bzw. auch besonders wegweisende Veröffentlichungen ausgezeichnet.
Im technischen Umfeld werden Verfahrensbeschreibungen oft als *White Paper* bezeichnet.

PlugIn

Software, die nicht eigenständig als Programm funktioniert, sondern als (heraustrennbarer) Teil einer größeren Anwendung fungiert.
Wird oft als optionaler Teil zu einem Basisprogramm (z. B. Webbrowser, Office-Programm) angeboten. Die *PlugIn*-Technik gilt im allgemeinen als besonders modulare Art des Programmdesigns.

Preprint

Zur Veröffentlichung in Zeitschriften bestimmte wissenschaftliche Arbeiten (Papers). Unter Preprint wird zumeist die Vorlage für den Verlag oder eine Kopie des Artikels verstanden. Manchmal fällt auch eine allgemein zugängliche Version einer Arbeit unter diese Definition.

Servlet

Als Servlet wird ein Programm (zumeist bezieht sich die Verwendung diese Begriffs auf in Java geschriebene Programme) bezeichnet, das als Webapplikation verwendet wird, also dynamische Webseiten auf Serverseite erzeugt. Der Name leitet sich als Kunstwort aus der Kombination von *Server* und *Applet* ab.

SQL

SQL (*Structured Query Language*) hat sich als Standardabfragesprache für Relationale Datenbanken (RDBS) entwickelt. Praktisch alle Hersteller solcher Systeme verwenden bzw. unterstützen diese Sprache.

URI

Unified Resource Identifier – Einheitlicher Bezeichner für eine Resource, die entweder mit ihrem Speicherort (z. B. URL) oder mit ihrem abstrakten Namen (URN) angegeben wird.

URN

Uniform Resource Name – Einheitlicher Name für eine Resource (ID). Dabei wird eine *URI* mit dem Schema `urn` für die Resource erzeugt; der genaue Speicherort der Resource kann dabei (empfohlen) über einen globalen Service verwaltet werden. Innerhalb einer URN ist dabei immer ein Namensraum spezifiziert, der international von der Organisation IANA verwaltet wird.

Abkürzungsverzeichnis

AHC Agglomerative Hierarchical Clustering – Clustermethode

AJAX Asynchronous JavaScript and XML – Siehe Seite 64

BOW Bag of words

ECMA European Computer Manufacturers Association – Normierungs- und Kooperationsorgan einer Vereinigung internationaler Computerhersteller

IDE Integrierte Entwicklungsumgebung / Integrated Developing Environment

JNI Java Native Interface – Java Schnittstelle um externen Programmcode zu verwenden

JSP Java Server Pages

LSA Latent Semantic Analysis – Statistisches Analyseverfahren

OAI Open Archives Initiative

PDF Dokumentenformat mit weitgehend identischer Ausgabe auf verschiedenen Rechnern.

RB Repeated Bisections – Clustermethode

SQL Structured Query Language – Eine Datenbankabfragesprache

SVD Singular Value Decomposition – Singulärwertzerlegung

UML Unified Modelling Language – Eine Modellierungssprache für verschiedene Aspekte der Softwareentwicklung

VP Versuchsperson

VSM Vector-Space-Model

W3C World Wide Web Consortium – Standardisierungsinstitution für Internettechniken

Literaturverzeichnis

[AXP07] *Java Ajax Frameworks*. http://ajaxpatterns.org/Java_Ajax_Frameworks. Version: 2007. – [Online, Stand 2007-10-28]

[BDH06] BERENDT, Bettina ; DINGEL, Kai ; HANSER, Christoph: Intelligent Bibliography Creation and Markup for Authors: A Step Towards Interoperable Digital Libraries. In: GONZALO, Julio (Hrsg.) ; THANOS, Costantino (Hrsg.) ; VERDEJO, M. F. (Hrsg.) ; CARRASCO, Rafael C. (Hrsg.): *ECDL* Bd. 4172, Springer, 2006 (Lecture Notes in Computer Science). – ISBN 3–540–44636–2, S. 495–499

[BDO95] BERRY, Michael W. ; DUMAIS, Susan T. ; O'BRIEN, Gavin W.: Using linear algebra for intelligent information retrieval. In: *SIAM Rev.* 37 (1995), Nr. 4, S. 573–595. http://dx.doi.org/http://dx.doi.org/10.1137/1037127. – DOI http://dx.doi.org/10.1137/1037127. – ISSN 0036–1445

[BE80] BICHTELER, J. ; EATON, E.A.: The combined use of bibliographic coupling and cocitation for document retrieval. In: *Journal of the American Society for Information Science* 31 (1980), Nr. 4, S. 278–282

[BFS03] BALDI, Pierre ; FRASCONI, Paolo ; SMYTH, Padhraic: *Modeling the Internet and the Web: Probabilistic Methods and Algorithms*. West Sussex, England : John Wiley and Sons, 2003. – ISBN 0470849061

[BH07] BERENDT, Bettina ; HAVEMANN, Frank: Beschleunigung der Wissenschaftkommunikation durch Open Access und neue Möglichkeiten der Qualitätssicherung. In: HAVEMANN, Frank (Hrsg.) ; PARTHEY, Heinrich (Hrsg.) ; UMSTÄTTER, Walther (Hrsg.): *Wissenschaftsforschung Jahrbuch 2007*. Berlin : Gesellschaft für Wissenschaftsforschung, 2007, S. 137–158

[BMR91] BRAAM, R.R. ; MOED, H.F. ; RAAN, A.F.J. van: Mapping of science by combined co-citation and word analysis. I. Structural aspects. In: *Journal of the American Society for Information Science* 42 (1991), Nr. 4, S. 233–251

[Bor05] BORTZ, Jürgen: *Statistik für Human- und Sozialwissenschaftler*. 6. Heidelberg : Springer Verlag, 2005. – ISBN 3–540–21271–X

[Bra06] BRAND, Matthew: Fast low-rank modifications of the thin singular value decomposition. In: *Linear Algebra and its Applications* 415 (2006), Nr. 1, S. 20–30. http://dx.doi.org/http://dx.doi.org/10.1016/j.laa.2005.07.021. – DOI http://dx.doi.org/10.1016/j.laa.2005.07.021

[bro05] *Der Brockhaus: Computer und Informationstechnologie*. 2. Mannheim : F.A. Brockhaus, 2005

[Bru98] BRUGBAUER, Ralf: Bibliothekarische Erfahrungen mit dem Impact Factor. In: *Bibliotheksdienst* 32 (1998), Nr. 3, 506–512. http://bibliotheksdienst.zlb.de/1998/1998_03_Erwerbung01.pdf. – [Online, Stand 2008-02-17]

[BYRN+99] BAEZA-YATES, R. ; RIBEIRO-NETO, B. u. a.: *Modern information retrieval*. Harlow, England : Addison-Wesley, 1999. – ISBN 0–201–39829–X

[Cat66] CATTELL, Raymond B.: The Scree Test For The Number Of Factors. In: *Multivariate Behavioral Research* 1 (1966), Nr. 2, S. 245–276

[CC00] COHN, David ; CHANG, Huan: Learning to Probabilistically Identify Authoritative Documents. In: LANGLEY, Pat (Hrsg.): *ICML*, Morgan Kaufmann, 2000. – ISBN 1–55860–707–2, S. 167–174

[DDF+90] DEERWESTER, S. ; DUMAIS, S.T. ; FURNAS, G.W. ; LANDAUER, T.K. ; HARSHMAN, R.: Indexing by latent semantic analysis. In: *Journal of the American Society for Information Science* 41 (1990), Nr. 6, S. 391–407

[DP07] DITTMANN, Michael ; PREUSS, Thomas: Nach Schema W – Fünf PHP-Template Systeme. In: *iX Special* 1 (2007), S. 57–61

[EG08] ERRAMI, Mounir ; GARNER, Harold: A tale of two citations. In: *Nature* 451 (2008), Nr. 7177, S. 397–9

[Gar79] GARFIELD, Eugene: *Citation indexing: Its theory and application in science, technology and humanities*. 1. New York, NY, USA : John Wiley and Sons, 1979

[Gar01] GARFIELD, Eugene: *From Bibliographic Coupling to Co-Citation Analysis via Algorithmic Historio-Bibliography*. http://www.garfield.library.upenn.edu/papers/drexelbelvergriffith92001.pdf. Version: 2001. – [Online, Stand 2007-12-05]

[GBL98] GILES, C. L. ; BOLLACKER, Kurt D. ; LAWRENCE, Steve: CiteSeer: an automatic citation indexing system. In: *DL '98: Proceedings of the third ACM conference on Digital libraries*. New York, NY, USA : ACM, 1998. – ISBN 0–89791–965–3, S. 89–98

[GHD02] GEDIGA, Günther ; HAMBORG, Kai-Christoph ; DÜNTSCH, Ivo: Evaluation of Software Systems. In: *Encyclopedia of Computer Science and Technology* 72 (2002), S. 166–192

[GHS01] GOLDSMITH, John A. ; HIGGINS, Derrick ; SOGLASNOVA, Svetlana: Automatic Language-Specific Stemming in Information Retrieval. In: *CLEF '00: Revised Papers from the Workshop of Cross-Language Evaluation Forum on Cross-Language Information Retrieval and Evaluation*. London, UK : Springer, 2001. – ISBN 3–540–42446–6, S. 273–284

[Gin95] GINSPARG, Paul: *First Steps towards Electronic Research Communication*. http://people.ccmr.cornell.edu/~ginsparg/blurb/blurb.ps.gz. Version: 1995. – [Online, Stand 2007-11-29]

[Gmü03] GMÜR, M.: Co-citation analysis and the search for invisible colleges: A methodological evaluation. In: *Scientometrics* 57 (2003), Nr. 1, S. 27–57

[Ham89] HAMERS, L.: Similarity Measures in Scientometric Research: The Jaccard Index versus Salton. In: *Information Processing and Management* 25 (1989), Nr. 3, S. 315–18

[Han08] HANDL, Andreas: *Multivariate Analysemethoden. Theorie und Praxis multivariater Verfahren unter besonderer Berücksichtigung von S-PLUS*. 1. Berlin : Springer Verlag, 2008. – ISBN 978–3540433866

[Hap04] HAPKE, Thomas: Ordnung, Fragmentierung und Popularisierung - Wilhelm Ostwald zur wissenschaftlichen Information und Kommunikation. In: *Vorträge zu dem Symposium anlässlich des 150. Geburtstages von Wilhelm Ostwald am 18. September 2003 in Großbothen*, Wilhelm-Ostwald-Gesellschaft zu Großbothen, 2004, S. 63–78. – Available online: http://doku.b.tu-harburg.de/volltexte/2007/360/

[Heg03] HEGNER, Marcus: *Methoden zur Evaluation von Software*. IZ, InformationsZentrum Sozialwissenschaften, 2003 http://www.social-science-gesis.de/Publikationen/Berichte/IZ_Arbeitsberichte/pdf/ab_29.pdf. – [Online, Stand 2008-02-24]

[Hö04] HÖLLER, Jürgen: Die Rückkehr der POJOs – Das Spring-Framework. In: *JavaMagazin* 9 (2004). http://www.javamagazin.de/spring. – [Online, Stand 2007-11-04]

[Hud07] HUDE, Marlis von d.: *Clusteranalyse*. [Online, Stand 2008-02-20]. http://vdh.inf.fh-bonn-rhein-sieg.de/upload/07Wintersemester/BCS5-StatDataMining2/Methoden/Teil2-Cluster.pdf. Version: 2007. – Vorlesungsunterlagen zur Veranstaltung: Statistisches Data Mining II

[HV07] HENNING, Peter A. ; VOGELSANG, Holger: *Handbuch Programmiersprachen*. 1. München : Hanser Verlag, 2007. – ISBN 978–3–446–40558–5

[Jac93] JACKSON, D.A.: Stopping Rules in Principal Components Analysis: A Comparison of Heuristical and Statistical Approaches. In: *Ecology* 74 (1993), Nr. 8, S. 2204–2214

[Jan07] JANSSENS, Frizo: *Clustering of scientific fields by integrating text mining and bibliometrics*. Katholieke Universiteit Leuven, ESAT-SCD, Kasteelpark Arenberg 10, B-3001 Leuven, Belgium, Katholieke Universiteit Leuven, Diss., 2007. http://hdl.handle.net/1979/847. – [Online, Stand: 2007-12-05]

[Kad06] KADEN, Ben: Über „Google Scholar“. (2006). http://www.ib.hu-berlin.de/~ben/texte/google_scholar_kaden.pdf. – [Online, Stand 2008-02-17]

[KLB05] KOVÁCS, F. ; LEGÁNY, C. ; BABOS, A.: Cluster Validity Measurement Techniques. (2005). http://www.bmf.hu/conferences/mtn2005/KovacsFerenc.pdf. – [Online, Stand 2008-02-22]

[Kle99] KLEINBERG, Jon M.: Authoritative sources in a hyperlinked environment. In: *J. ACM* 46 (1999), Nr. 5, S. 604–632. http://dx.doi.org/http://doi.acm.org/10.1145/324133.324140. – DOI http://doi.acm.org/10.1145/324133.324140. – ISSN 0004–5411

[LA99] LARSEN, Bjornar ; AONE, Chinatsu: Fast and effective text mining using linear-time document clustering. In: *KDD '99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA : ACM, 1999. – ISBN 1–58113–143–7, S. 16–22

[Ley08] LEYDESDORFF, Loet: On the Normalization and Visualization of Author Co-Citation Data: Salton's Cosine versus the Jaccard Index. In: *Journal of the American Society of Information Science & Technology* 59 (2008), Nr. 1, S. 77–85

[LMS07] LEISEGANG, Christoph ; MINTERT, Stefan ; SPANNEBERG, Bastian: Die Nullnummern – Fünf serverseitige Ajax-Frameworks. In: *iX Special* 1 (2007), S. 35–41

[MHSW03] MERLO, Paola ; HENDERSON, James ; SCHNEIDER, Gerold ; WEHRLI, Eric: Learning Document Similarity Using Natural Language Processing. In: *Linguistik online* 17 (2003), Nr. 5, 99–115. http://www.linguistik-online.de/17_03/merlo.pdf. – ISSN 1615–3014. – [Online, Stand 2008-02-20]

[ND06] NEUHAUS, Christoph ; DANIEL, Hans-Dieter: Data sources for performing citation analysis: An overview. In: *Journal of Documentation* (in press) (2006). http://e-collection.ethbib.ethz.ch/show?type=bericht&nr=490&part=text. – [Online, Stand 2007-12-15]

[Nie93] NIELSEN, Jakob: *Usability Engineering*. 1. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1993. – ISBN 0–12–518406–9

[Nie00] NIELSEN, Jakob: Why You Only Need to Test With 5 Users. (2000). http://www.useit.com/alertbox/20000319.html. – [Online, Stand: 2007-12-05]

[Nie03] NIELSEN, Jakob: *Introduction to Usability*. http://www.useit.com/alertbox/20030825.html. Version: 2003. – [Online, Stand 2008-02-23]

[Nie04] NIELSEN, Jakob: *Heuristics for User Interface Design: Ten Usability Heuristics*. http://www.useit.com/papers/heuristic/heuristic_list.html. Version: 2004. – [Online, Stand 2008-02-24]

[NL06] NIELSEN, Jakob ; LORANGER, Hoa: *Web Usability*. 1. München : Addison-Wesley Verlag, 2006. – ISBN 978–3–8273–2448–1

[O'R04] O'ROURKE, Cameron: A Look at Rich Internet Applications: Lookinq at technoloqies for going beyond the aging HTML standard. In: *Oracle Magazine* 18 (2004), Nr. 4, S. 59–60

[O'R05] O'REILLY, Tim: *What Is Web 2.0 - Design Patterns and Business Models for the Next Generation of Software*. http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html. Version: 2005. – [Online; Stand 2007-10-04]

[OSW04] OSINSKI, S. ; STEFANOWSKI, J. ; WEISS, D.: Lingo: Search Results Clustering Algorithm Based on Singular Value Decomposition. In: *Intelligent Information Processing And Web Mining: Proceedings of the International IIS: IIPWM'04 Conference held in Zakopane, Poland, May 17-20, 2004* (2004)

[Pap07] PAPITSCH, Alexander: *Information Literacy and electronic learning environments: Requirements, concerns, factors, usability and solutions*. Humboldt-Universität zu Berlin, Wirtschaftswissenschaftliche Fakultät, Master thesis, 2007

[Pet02] PETER SIMONS AND RALPH BABEL: *FastCGI – The Forgotten Treasure*. http://cryp.to/publications/fastcgi/. Version: 2002. – [Online, Stand 2007-11-13]

[PN76] PINSKI, G. ; NARIN, F.: Citation Influence for Journal Aggregates of Scientific Publications: Theory, with Application to the Literature of Physics. In: *Information Processing and Management* 12 (1976), S. 297–312

[Poh07] POHL, Klaus: *Requirements Engineering: Grundlagen, Prinzipien, Techniken*. 1. Heidelberg : dpunkt.verlag, 2007. – ISBN 978–3–89864–342–9

[Por77] PORTER, A.L.: Citation Analysis: Queries and Caveats. In: *Social Studies of Science* 7 (1977), Nr. 2, S. 257–267

[PVR91] PETERS, HPF ; VAN RAAN, AFJ: Structuring scientific activities by co-author analysis. In: *Scientometrics* 20 (1991), Nr. 1, S. 235–255

[Que03] QUESENBERY, Whitney: Dimensions of Usability: Defining the Conversation, Driving the Process. In: *Proceedings of the Usability Professional's Association (UPA) conference on Ubiquitous Usability, June 23-27, 2003*. Scottsdale, Arizona, USA : Usability Professional's Association, 2003

[Qus03] QUSAY H. MAHMOUD – SUN MICROSYSTEMS: *Servlets and JSP Pages Best Practices*. http://java.sun.com/developer/technicalArticles/javaserverpages/servlets_jsp/. Version: 2003. – [Online, Stand 2007-11-04]

[Sch05] SCHMID, Karl: *Web-Frameworks bei der Entwicklung von Web-Applikationen anhand von Apache Struts und JavaServer Faces"*. Fachhochschule Köln, Fachbereich Informatik, Diplomarbeit, 2005

[SEW03] STEIN, B. ; EISSEN, S.M. zu ; WISSBROCK, F.: On Cluster Validity and the Information Need of Users. In: *Proceedings of the 3rd IASTED International Conference on Artificial Intelligence and Applications (AIA 03), Benalmádena, Spain* (2003), S. 216–221

[SG03] STREHL, A. ; GHOSH, J.: Cluster ensembles- A knowledge reuse framework for combining multiple partitions. In: *Journal of Machine Learning Research* 3 (2003), Nr. 3, S. 583–617

[SKK00] STEINBACH, M. ; KARYPIS, G. ; KUMAR, V.: A comparison of document clustering techniques. In: *KDD Workshop on Text Mining* 34 (2000), S. 35

[SM83] SALTON, Gerard ; MACGILL, Michael J.: *Introduction to modern information retrieval*. New York : McGraw-Hill Book Comp., 1983. – ISBN 0–07–054484–0

[Sma73] SMALL, Henry: Co-citation in the scientific literature: A new measure of the relationship between two documents. In: *Journal of the American Society for Information Science* 24 (1973), Nr. 4, S. 265–269. http://dx.doi.org/http://dx.doi.org/10.1002/asi.4630240406. – DOI http://dx.doi.org/10.1002/asi.4630240406

[Smi97] SMITH, Richard: Peer review: reform or revolution? In: *BMJ* 315 (1997), Nr. 7111, S. 759–60

[SP63] SOLLA PRICE, Derek J.: *Little Science, Big Science*. 1. New York, NY, USA : Columbia University Press, 1963

[SS85] SMALL, Henry ; SWENEEY, E.: Clustering the Science Citation Index using co-citations. In: *Scientometrics* 7 (1985), Nr. 3-6, S. 391–409

[SS08] SEVERIENS, Thomas ; SCHIRMBACHER, Peter: Aufbau eines Netzwerkes von zertifizierten Open Access Repositories. In: *Workshop: „Förderung der wissenschaftlichen Informationslandschaft in Deutschland" – Chancen und Strategien beim Aufbau vernetzter Repositorien, Berlin, Februar 2008*, Deutsche Initiative für Netzwerkinfomation E.V. (DINI), 2008. – http://www.dini.info/fileadmin/workshops/februar-2008/Severiens_Schirmbacher.pdf [Online, Stand 2008-02-24]

[Sta05] STANGL, Werner: *Test und Experiment in der Psychologie*. http://testexperiment.stangl-taller.at/experimentdefinition.html. Version: 2005. – [Online, Stand 2008-02-23]

[Ste07] STEGBAUER, Michael: *Zitationsanalyse für das Gebiet 'Lernen in Spielen'*. Universität Darmstadt, Fachbereich Informatik – Knowledge Engineering, Diplomarbeit, 2007

[Tho03] THOMAS BAYER – ORIENTATION IN OBJECTS GMBH: *Webframeworks*. http://www.oio.de/webframeworks.htm. Version: 2003. – [Online, Stand 2007-11-05]

[TSK05] TAN, Pang N. ; STEINBACH, M. ; KUMAR, V.: *Introduction to Data Mining*. Boston, MA, USA : Addison-Wesley Longman Publishing Co., 2005

[Ulr02] ULRICH, Winfried: *Wörterbuch Linguistische Grundbegriffe*. 5. Berlin : Gebrüder-Borntraeger-Verl.-Buchh., 2002. – ISBN 3–443–03111–0

[VB07] VAN, Thanh-Trung ; BEIGBEDER, Michel: Web Co-citation: Discovering Relatedness Between Scientific Papers. In: *Advances in Intelligent Web Mastering – Proceedings of the 5th Atlantic Web Intelligence Conference – AWIC'2007, Fontainbleau, France, June 25 – 27, 2007*. Berlin : Springer Verlag, 2007. – ISBN 978–3–540–72574–9, S. 343–348

[Wik07a] WIKIPEDIA: *Adobe Flash — Wikipedia, Die freie Enzyklopädie*. http://de.wikipedia.org/w/index.php?title=Adobe_Flash&oldid=37457026. Version: 2007. – [Online; Stand 2007-10-04]

[Wik07b] WIKIPEDIA: *Search/Retrieve via URL — Wikipedia, Die freie Enzyklopädie*. http://de.wikipedia.org/w/index.php?title=Search/Retrieve_via_URL&oldid=36691772. Version: 2007. – [Online; 2008-02-16]

[Wik07c] WIKIPEDIA: *Servlet — Wikipedia, Die freie Enzyklopädie*. http://de.wikipedia.org/w/index.php?title=Servlet&oldid=37339903. Version: 2007. – [Online; Stand 2007-10-28]

[Wik07d] WIKIPEDIA: *Technologie — Wikipedia, Die freie Enzyklopädie*. http://de.wikipedia.org/w/index.php?title=Technologie&oldid=38454375. Version: 2007. – [Online; Stand 2007-11-09]

[Wik08a] WIKIPEDIA: *Matthäuseffekt — Wikipedia, Die freie Enzyklopädie*. http://de.wikipedia.org/w/index.php?title=Matth%C3%A4useffekt&oldid=41245643. Version: 2008. – [Online; Stand 2008-02-08]

[Wik08b] WIKIPEDIA: *Stemming — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Stemming&oldid=183816431. Version: 2008. – [Online; Stand 2008-01-23]

[Wik08c] WIKIPEDIA: *The Collection of Computer Science Bibliographies — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=The_Collection_of_Computer_Science_Bibliographies&oldid=192764536. Version: 2008. – [Online; 2008-02-22]

[WMW07] WANG, X. ; MCCALLUM, A. ; WEI, X.: Topical N-grams: Phrase and Topic Discovery, with an Application to Information Retrieval. In: *Proceedings of the 7th IEEE International Conference on Data Mining (ICDM), 2007*, IEEE, 2007. – [Online: Stand: 2008-02-23]

[ZE98] ZAMIR, O. ; ETZIONI, O.: Web document clustering: a feasibility demonstration. In: *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval* (1998), S. 46–54

[ZK02] ZHAO, Ying ; KARYPIS, George: Evaluation of hierarchical clustering algorithms for document datasets. In: *CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management*. New York, NY, USA : ACM, 2002. – ISBN 1–58113–492–4, S. 515–524