UNIVERSITÄT PADERBORN
*Die Universität der Informationsgesellschaft*

Fakultät für Elektrotechnik, Informatik und Mathematik
Institut für Informatik
Fachgebiet Didaktik der Informatik

# INSPIRE: INSIGHT TO SCIENTIFIC PUBLICATIONS AND REFERENCES – VERTEILTE BERECHNUNG VON BIBLIOMETRIKEN AUF GROSSEN DATENMENGEN

**Masterarbeit**

vorgelegt von

**Adrian Wilke**

Paderborn, 30. April 2013

Betreuer

Dr. Wolfgang Reinhardt

Gutachter

Professor Dr. Johannes Magenheim
Juniorprofessor Dr. Christian Plessl

NACHTRÄGLICHE ÄNDERUNGEN ZUR VERÖFFENTLICHUNG

Entfernung persönlicher Angaben, Hinweis Copyright, Layout Titelblatt, keine inhaltlichen Änderungen.

URN: urn:nbn:de:hbz:466:2-28348
DOI: 10.17619/UNIPB/1-88

*INSPIRE: Insight to Scientific Publications and References –*
*Verteilte Berechnung von Bibliometriken auf großen Datenmengen*
Adrian Wilke

# ZUSAMMENFASSUNG

Die vorliegende Arbeit behandelt die Berechnung von Literaturempfehlungen für wissenschaftliche Publikationen. Dazu wird ein Softwaresystem entwickelt, mit dem eine automatische Zitationsanalyse durchgeführt werden kann. Dies umfasst die Untersuchung eines unterliegenden Dokumentennetzwerkes durch die Kombination verschiedener Ansätze der Bibliometrie.

Die Ergebnisse der Zitationsanalyse hängen von der Menge und Qualität der unterliegenden Datenbasis ab. Diese wird durch ein Dokumentennetzwerk repräsentiert und besteht im Wesentlichen aus Publikationen und Referenzierungen zwischen diesen. Um diese Datenbasis aufzubauen, werden im Vorfeld Daten verschiedener Formate verarbeitet.

Zunächst findet eine Extraktion von Volltexten aus PDF-Dateien statt. Aus den Volltexten werden anschließend Metadaten zu Publikationen und Metadaten zu Referenzen der Literaturverzeichnisse extrahiert. Dieser Extraktionsvorgang verwendet externe Open Source Anwendungen. Da die Datenqualität der Extraktion gering ist, findet eine zusätzliche Nachbearbeitung der extrahierten Daten statt. Die redundant vorliegenden Metadaten werden dabei in ein integriertes Format zusammengeführt.

Extrahierte Metadaten zu Publikationen und Referenzen werden anschließend genutzt, um ein Dokumentennetzwerk aufzubauen. Eine passende Lösung zur Datenhaltung wird im Vorfeld durch einen Benchmark ermittelt. Aus dem vorbereiteten Dokumentennetzwerk kann abschließend eine Publikation gewählt werden, für die Literaturempfehlungen ermittelt und präsentiert werden.

Da die benötigten Berechnungen zeitintensiv sind und als Grundlage mehrere Hunderttausend Dokumente dienen, werden sie verteilt in einem Rechnercluster durchgeführt. Für die verteilten Berechnungen findet dabei das Hadoop Framework Verwendung.

# INHALTSVERZEICHNIS

# AKRONYME

| | |
|---|---|
| AAN | Artefact-Actor-Networks |
| CPI | Citation Proximity Index |
| CRF | Conditional Random Field |
| DOAJ | Directory of Open Access Journals |
| DOI | Digital Object Identifier |
| DSI | Distance Similarity Index |
| ER | Entity-Relationship |
| HCPA | Hadoop Cluster for Large Scale Publication Analyses |
| ID | Identifikationsbezeichnung |
| ISBN | International Standard Book Number |
| ISSN | International Standard Serial Number |
| NoSQL | Not only SQL |
| TAM | Technology Acceptance Model |
| URI | Uniform Resource Identifier |
| URL | Uniform Resource Locator |

# 1 EINLEITUNG

## 1.1 MOTIVATION

Die Menge frei verfügbarer digitaler Dokumente hat in den vergangenen Jahrzehnten stark zugenommen. Die Verfügbarkeit ist zum einen auf den Ausbau von Internetzugängen zurückzuführen. Die Anzahl der Internetzugänge von Haushalten in der Europäischen Union ist in den Jahren 2006 bis 2012 von 49 % auf 76 % gestiegen [Eur12]. Zum anderen entwickeln sich auch vielfältige Möglichkeiten des Datenaustauschs. So kann das Portable Document Format (PDF) seit seiner Veröffentlichung 1993 für den Austausch von Dokumenten verwendet werden.

PDF-Dateien werden häufig für digitale Veröffentlichungen von wissenschaftlichen Publikationen genutzt. Auch beim Vorgang des Publizierens wissenschaftlicher Artikel ist eine Veränderung festzustellen. Nach einer Untersuchung zu wissenschaftlichen Publikationen, die zum freien Zugang veröffentlicht werden, stieg die Anzahl der Artikel von 247 im Jahr 1993 auf 191.851 im Jahr 2009 [LWB+11]. Veröffentlichungen mit der Erlaubnis, ein Dokument gebührenfrei herunterzuladen, zu lesen und zu verlinken, werden mit dem Begriff *Open Access* bezeichnet. Neben Veröffentlichungen über den goldenen Weg des Open Access, eine primäre frei zugängliche Veröffentlichung, stellen viele Autoren ihre Dokumente über den grünen Weg zur Verfügung. Auf diesem Weg werden Dokumente unabhängig von ihrer ursprünglichen Veröffentlichungsart im Web zum Download bereitgestellt. Dadurch ergibt sich insgesamt eine sehr große Menge wissenschaftlicher Publikationen, die frei zugänglich ist.

Die im Web frei zugänglichen Publikationen stehen oftmals als Listen zur Verfügung, die sich aus den Veröffentlichungen einer Person zusammensetzen. Vielen Studierenden und Forschern ist die Situation bekannt, in der man durch Empfehlungen oder Verlinkungen auf einzelne Dokumente stößt, deren Inhalte oder Ergebnisse relevant für die eigene Arbeit sind. Beim Fund so eines Dokuments stellt sich die Frage danach, ob Publikationen mit ähnlichen Inhalten oder Ergebnissen existieren. Konnten die Ergebnisse bereits in ähnlicher Form reproduziert werden? Welche anderen Arbeiten ähneln der vorliegenden Publikation oder stützen sich auf ihre Ergebnisse?

Diese und ähnliche Situationen sind Ausgangspunkte einer Literaturrecherche. Es schließt sich oftmals eine Untersuchung des Litera-

turverzeichnisses der gefundenen Publikation an. Durch eine Suche nach weiteren Veröffentlichungen der Autoren kann auch zusätzliche relevante Literatur gefunden werden. Insgesamt wäre eine Möglichkeit wünschenswert, Literaturempfehlungen für die gefundene Publikation auf direktem Weg zu erhalten. Dies kann auch Publikationen einschließen, die möglicherweise erst nach der Veröffentlichung des gefundenen Dokuments entstanden sind und deshalb nicht durch die Lektüre des Literaturverzeichnisses der betrachteten Publikationen gefunden werden können.

Als Grundlage einer solchen Literaturempfehlung können Open Access Publikationen dienen. Diese sind frei zugänglich, daher entstehen keine zusätzlichen Barrieren für den Zugriff. Die Menge relevanter Publikationen kann sehr groß ausfallen. Daher kann die zu durchsuchende Datenmenge auch bei einer automatischen Verarbeitung viel Zeit in Anspruch nehmen.

## 1.2 ZIELSETZUNG

Ziel der vorliegenden Arbeit ist die Realisierung eines Systems zur Berechnung von Literaturempfehlungen für wissenschaftliche Publikationen. Die Berechnungen stützen sich dabei auf eine Datenbasis mehrerer Hunderttausend Publikationen, die als PDF-Dateien vorliegen. Um diese verteilt und parallelisiert zu verarbeiten, findet das Hadoop-Framework Verwendung. Aus den vorliegenden PDF-Dateien werden Metadaten extrahiert, mit denen ein Dokumentennetzwerk erstellt wird. Dieses bildet die Grundlage für die Anwendung von Bibliometriken, die wiederum die Basis der Literaturempfehlungen darstellen.

Für diese Arbeit ergeben sich daraus die folgenden Aufgaben:

- Zur Datenhaltung des Dokumentennetzwerks soll eine geeignete Softwarelösung ausgewählt werden. Die Eignung ergibt sich aus der Anforderung einer performanten Berechnung von Bibliometriken. Im Vorfeld dieser Arbeit wurden dafür eine relationale Datenbank, eine Graphdatenbank und eine verteilte Graphdatenbank als mögliche Lösungen ausgewählt
- Als Datenbasis dieser Arbeit dienen mehrere hunderttausend wissenschaftliche Publikationen, die im PDF-Format vorliegen. Aus diesen sollen Metadaten extrahiert werden, die zur Identifizierung von Publikationen dienen. Dies umfasst Metadaten zu den Publikationen und den zugehörigen Literaturangaben. Beispiele für Metadaten sind Autoren, Titel und das Jahr der Veröffentlichung einer Publikation. Es soll analysiert werden, welche Metadaten extrahiert werden können und welche der Metadaten sich zu Identifizierung von Publikationen eignen.

- Im Vorfeld dieser Arbeit war bekannt, dass extrahierte Metadaten qualitative Mängel aufweisen. Die extrahierten Daten verschiedener Softwarelösungen sind lückenhaft. Dem soll durch eine Integration verschiedener Formate entgegengewirkt werden. Es ist ein Verfahren zu entwickeln, mit dem die extrahierten Daten kombiniert werden.
- Bei der Generierung eines Dokumentennetzwerkes muss für jede Publikation, die hinzugefügt werden soll, überprüft werden, ob bereits ein Datensatz im System vorhanden ist, der dieser Publikation entspricht. Dadurch wird vermieden, dass Duplikate im System erstellt werden. Es ist eine Implementierung für diese Deduplikation zu entwickeln.
- Es sollen die Bibliometriken *Referenzen*, *Zitiert-von*, *Bibliografische Kopplung*, *Kozitation* und die *Distanz von Zitationen* berechnet werden. Die Vorstellung der Bibliometriken geschieht innerhalb dieser Arbeit. Je nach Einbindung einer Publikation im Dokumentennetzwerk ergeben die Berechnungen der Bibliometriken unterschiedlich gute Ergebnisse. Um diesen Unterschieden zu begegnen soll, eine Kombination der Bibliometriken entwickelt werden.
- Für die Ermittlung der Distanz von Zitationen muss neben den Referenzierungen einer Publikation eine Analyse ihres Volltextes vorgenommen werden. Unter Verwendung der extrahierten Daten soll eine Lösung entwickelt werden, mit der die Distanzen von Zitationen für Literaturempfehlungen genutzt werden können.
- Es ist außerdem eine Benutzerschnittstelle zu entwickeln, mit der die Ergebnisse der Berechnung von Literaturempfehlungen präsentiert werden können. Über diese soll eine Navigation zwischen Publikationen möglich sein.

## 1.3 VERWANDTE PROJEKTE UND ARBEITEN

In meiner Bachelorarbeit [Wil10] habe ich mich mit der Integration verschiedener Datenquellen in ein System zur Datenhaltung und Analyse von Artefact-Actor-Networks (AAN) beschäftigt. AAN stellen eine Konsolidierung verschiedener Einzelnetzwerke dar. Dabei werden Nutzerkonten von Webseiten, darüber veröffentlichte Dokumente und Schlüsselwörter zu diesen verwendet und deren Beziehungen untersucht.

Das bestehende AAN-System wurde später in einer Partnerarbeit zum Seminar „Future Social Learning Networks“ [SW11] durch ein zusätzliches Netzwerk erweitert. In dieser Arbeit wurden Daten zu Zeitschriften und Artikeln aus dem Directory of Open Access Journals

(DOAJ) [DOAJ] angebunden. Über die im DOAJ veröffentlichten Metadaten entstand die Möglichkeit, Koautor-Beziehungen wissenschaftlicher Publikationen zu analysieren. Volltexte und Metadaten, die direkte Verweise zwischen Dokumenten darstellen, konnten über das DOAJ nicht bezogen werden. Die Extraktion von Volltexten und Metadaten wurde in der Seminararbeit als mögliche Erweiterung in der Zukunft genannt.

In der Projektgruppe knowAAN [knowAAN] wurde die Möglichkeit der Extraktion von Volltexten und Metadaten genutzt. Basierend auf den extrahierten Daten fanden ein Clustering und eine Analyse von Volltexten sowie die Untersuchung von Koautor- und Kozitationsnetzwerken statt.

Die folgende Projektgruppe PUSHPIN[1] [PUSHPIN] baute auf diesen Ergebnissen auf. In der PG PUSHPIN wurde ein soziales Netzwerk entwickelt und die Analysen von wissenschaftlichen Publikationen um eine Verarbeitung mit Werkzeugen für große Datenmengen erweitert. In diesem Zusammenhang entstand auch der Hadoop Cluster for Large Scale Publication Analyses (HCPA) [HCPA], der von Tobias Varlemann [Var13] und Wolfgang Reinhardt (der Betreuer dieser Arbeit) aufgebaut und betreut wird.

Aus den erwähnten Projekten entstand die Möglichkeit der Anfertigung von Abschlussarbeiten für einzelne Spezialbereiche. Tobias Varlemann [Var13] beschäftigte sich mit der Findung von Ähnlichkeiten von Volltexten. Nicolas Schelp [Schck] behandelt in seiner Arbeit die Erkennung ähnlicher Strukturen in Volltexten durch dynamisch erweiterbare Clusterings. In der vorliegenden Arbeit werden Literaturempfehlungen durch eine auf extrahierten Metadaten basierenden Zitationsanalyse berechnet. Dabei werden innerhalb der erwähnten Projekte erstmals Distanzen von Zitationen in Volltexten berücksichtigt. Für mich ergibt sich mit dieser Arbeit erstmalig die Möglichkeit der Nutzung eines Rechnerclusters für verteilte, parallele Berechnungen.

Ich bedanke mich bei Dr. Wolfgang Reinhardt für seine Betreuung und bei Prof. Dr. Johannes Magenheim für die Ermöglichung dieser und der vergangenen Arbeiten.

## 1.4 STRUKTUR DER ARBEIT

Die vorliegende Arbeit wird im zweiten Kapitel mit Grundlagen zum Verständnis der weiteren Arbeit fortgeführt. Diese setzen sich aus vier Abschnitten zusammen. In Abschnitt 2.1 werden Grundlagen der Bibliometrie vorgestellt. Anschließend werden in Abschnitt 2.2 Verfahren und Software-Anwendungen zur Extraktion von Metada-

1 Der Autor dieser Arbeit war kein Teilnehmer der Projektgruppe PUSHPIN

ten aus wissenschaftlichen Publikationen behandelt. Darauf folgend werden in Abschnitt 2.3 verwandte Lösungsansätze vorgestellt. Das in dieser Arbeit entstehende System verwendet die Möglichkeit verteilter Berechnungen. Die technischen Grundlagen dazu werden in Abschnitt 2.4 beschrieben.

Das dritte Kapitel umfasst drei Abschnitte, in denen jeweils Analysen durchgeführt werden. In Abschnitt 3.1 werden XML-Dateien untersucht, die das Resultat der Extraktion von Metadaten darstellen. Dabei wird die Häufigkeit der beinhalteten XML-Elemente ermittelt. Ein Teil der verfügbaren XML-Dateien enthält Daten, die zur Ermittlung der Distanz von Zitationen verwendet werden können. Dazu müssen die Daten mit Volltexten der Publikationen abgeglichen werden. Welche Möglichkeiten hier bestehen, wird in Abschnitt 3.2 analysiert. Der Abschluss der Untersuchungen bildet die Durchführung eines Benchmarks in Abschnitt 3.3. Dieser wird verwendet, um eine geeignete Lösung zur Datenhaltung für Dokumentennetzwerke auszuwählen.

Im vierten Kapitel wird das System zur Berechnung von Literaturempfehlungen konzipiert. Dazu erfolgt zunächst eine Einteilung in fünf aufeinanderfolgende Phasen. Abschnitt 4.1 beschreibt den Aufbau des 5-Phasen-Modells. Die durchzuführenden Berechnungen sollen in einem Rechnercluster durchgeführt werden. Ein Entwurf der verteilten Berechnung wird in Abschnitt 4.2 vorgestellt. Im Anschluss werden die fünf Phasen Konvertierung, Extraktion, Integration, Deduplikation und Präsentation detailliert beschrieben.

Details der Implementierung werden im fünften Kapitel behandelt. Die Gliederung dieses Kapitels ist erneut in die fünf Phasen unterteilt.

In Kapitel sechs werden die erarbeiteten Ergebnisse und eine Evaluierung vorgestellt. Dabei werden in Abschnitt 6.1 zunächst Ergebnisse der Extraktion behandelt. Dies sind benötigte Laufzeiten und extrahierte Datenmengen. Die Gegenüberstellung der benötigten Ausführungszeiten zu den Phasen ist Bestandteil von Abschnitt 6.2. Ergebnisse zur Präsentations-Phase werden im Anschluss gesondert vorgestellt. Dies geschieht als Test zur Performanz der Benutzerschnittstelle in Abschnitt 6.3. Der letzte Teil dieses Kapitels ist die Evaluierung des entwickelten Systems. Die Ergebnisse der dazu durchgeführten Befragung sind in Abschnitt 6.4 aufgeführt.

Eine abschließende Diskussion und mögliche Erweiterungen sind Inhalte des siebten Kapitels.

# 2 GRUNDLAGEN

Dieses Kapitel enthält Grundlagen, die für das Verständnis der weiteren Arbeit benötigt werden. In Abschnitt 2.1 werden Bibliometriken vorgestellt, die zur Berechnung der Literaturempfehlungen verwendet werden. Softwarelösungen für die Extraktion von Metadaten werden in Abschnitt 2.2 vorgestellt. Abschnitt 2.3 enthält einen Überblick bestehender Lösungen, die mit dieser Arbeit verwandt sind. Zuletzt werden die technischen Grundlagen zur verteilten Berechnung in Abschnitt 2.4 vorgestellt.

## 2.1 BIBLIOMETRIE

Dieser Abschnitt enthält zunächst eine Einführung zu Metadaten, die sowohl für wissenschaftliche Publikationen, als auch für deren Referenzen vorliegen. Anschließend werden die Bibliometriken *Bibliografische Kopplung* und *Kozitation* vorgestellt. Der junge Ansatz der *Distanz von Zitationen* wird anschließend behandelt.

### 2.1.1 *Metadaten und Referenzen*

Der Aufbau wissenschaftlicher Publikationen folgt in der Regel einem Muster. Im Kopfteil sind dabei zumindest die Namen der Autoren und der Titel der Publikation aufgeführt. Bei vielen Publikationen sind zusätzlich Daten zur beinhaltenden Zeitschrift oder dem Publikationsjahr vermerkt. Diese Angaben sind Metadaten. Metadaten sind Daten, die dazu verwendet werden, Datensätze oder Dokumente zu beschreiben.

Innerhalb des textuellen Teils von Publikationen wird – häufig im Rahmen der Verwendung von Zitaten – auf weitere Dokumente verwiesen. Diese Verweise bestehen zunächst aus Markierungen im Text. Am Ende einer wissenschaftlichen Publikation werden diese Markierungen in einem Verzeichnis der verwendeten Literatur nochmals aufgeführt. Zu den Markierungen werden Metadaten angegeben, mit denen die für die Publikation verwendete Literatur identifiziert werden kann.

Eine solche Identifikation wird in einer wichtigen Arbeit zur Zitationsanalyse beschrieben. Die betreffende Arbeit wird im folgenden Abschnitt besprochen. Kessler verwendet in dieser Arbeit eine Kom-

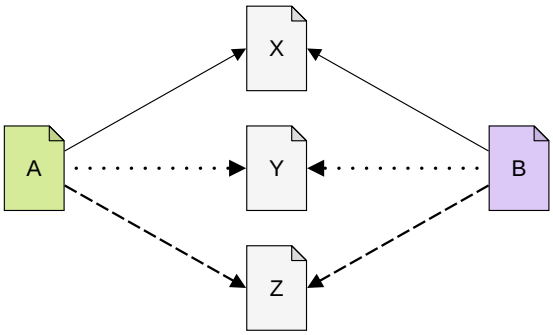



Abbildung 1: Bibliografische Kopplung: Die Publikationen A und B referenzieren jeweils X, Y und Z. Die Stärke der Bibliografischen Kopplung zwischen A und B ist somit 3.

bination aus Titel, Autoren, Zeitschriften-Titel, Zeitschrift-Band und Seitennummer, um Publikationen zu identifizieren. „Diese Informationen wurden extrahiert und auf Magnetband gespeichert. [...] Die Daten identifizieren jedes bibliografische Element“[1] [Kes63].

Die Referenzen im Literaturverzeichnis können als Literaturempfehlungen von Autoren interpretiert werden. Sie stellen eine wichtige Quelle zur Literaturrecherche dar.

### 2.1.2 *Bibliografische Kopplung*

Eine Bibliografische Kopplung zwischen zwei Publikationen besteht, wenn beide Publikationen mindesten eine gemeinsame Referenz aufführen. Der Begriff Bibliografische Kopplung wurde von M. M. Kessler [Kes63] geprägt. Kessler beschreibt eine Kopplungseinheit als eine einzelne Referenz, die von zwei Publikationen genutzt wird. Die Stärke der Bibliografischen Kopplung zweier Publikationen ergibt sich nach Kessler durch die Anzahl der Kopplungseinheiten, also durch die Anzahl der gemeinsamen Referenzen. Ein Beispiel für eine Bibliografische Kopplung der Stärke 3 zeigt Abb. 1.

Eine Eigenschaft der Bibliografischen Kopplung ist ihre Stabilität. Ein für zwei Publikationen berechneter Wert ändert sich nicht. Dies wird auch von den Entdeckern der Kozitation (siehe Abschnitt 2.1.3) erwähnt. Small beschreibt die Bibliografische Kopplung als fest und dauerhaft [Sma73]. Auch Marshakova schreibt, dass sich diese Verbindung mit der Zeit nicht ändert [Mar73].

Im Gegensatz zur Kozitation kann die Bibliograische Kopplung dazu verwendet werden, kürzlich veröffentlichte Publikationen miteinander zu vergleichen. Der Wert der Kozitation hängt von der Verwendung zweier zu vergleichenden Publikationen in anderen Publikatio-

1 Übersetzung des Autors

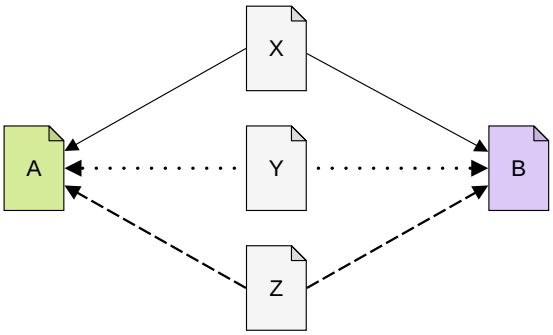



Abbildung 2: Kozitation: Die Publikationen A und B werden jeweils von X, Y und Z referenziert. Die Stärke der Kozitation zwischen A und B ist somit 3.

nen ab. Die Anzahl der Verwendungen kann mit der Zeit wachsen. Bei der Bibliografischen Kopplung besteht diese Abhängigkeit nicht.

Die Bibliografische Kopplung kann verwendet werden, um thematisch verwandte Publikationen zu finden. Publikationen, die eine hohe Anzahl gleicher Referenzen aufweisen, stützen sich womöglich auf ähnliche Ergebnisse. Dies ist allerdings nur als Indikator zu verstehen. Die Inhalte, auf die durch Zitate verwiesen wird, können stark voneinander abweichen. Selbst wenn eine gleiche Aussage in zwei Publikationen verwendet wird, kann sie in einem Fall zustimmend gebraucht werden, während sie im anderen Fall abgelehnt wird. Weiterhin kann durch das Fehlen einer Bibliografischen Kopplung nicht darauf geschlossen werden, dass die Inhalte zweier Publikationen nicht miteinander verwandt sind.

Neben der Verwendung absoluter Werte bestehen Möglichkeiten, die Stärke der Bibliografischen Kopplung im Verhältnis zu anderen Größen zu nutzen. Beispielsweise können Minima, Maxima oder das arithmetische Mittel der Länge von Referenzlisten zweier betrachteter Publikation genutzt werden.

### 2.1.3 *Kozitation*

Die Kozitation wurde 1973 von Small [Sma73] und Marshakova [Mar73] unabhängig voneinander entdeckt. Small definiert diese Form der Kopplung von Dokumenten als Häufigkeit, mit der zwei Publikationen gemeinsam zitiert werden. Abb. 2 zeigt ein Beispiel einer Kozitation der Stärke 3. Small weist darauf hin, dass die Kozitation eine Beziehung beschreibt, die von den zitierenden Autoren hergestellt wird. In einer Analyse zeigt er für zwei nachweislich thematisch sehr ähnliche Publikationspaare eine hohe Kozitationsstärke und zugleich eine niedrige Stärke der Bibliografischen Kopplung. Diese Beobachtungen sprechen für die Kozitation als einen besseren Indikator für

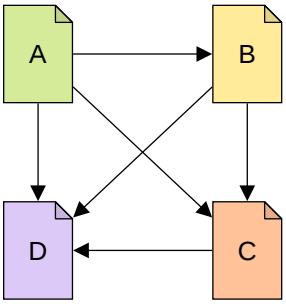



Abbildung 3: Netzwerk aus Publikationen. Die Stärken der Bibliografischen Kopplung und der Kozitation sind in den Tabellen 1 und 2 dargestellt.

thematische Ähnlichkeit als den der Bibliografischen Kopplung. Er beschreibt außerdem, dass eine starke Kozitation für in Verbindung stehende Ideen oder ihrem gemeinsamen Auftreten spricht. Marshakova beschreibt die Methode als logisches Gegenstück der Bibliografischen Kopplung.

Im Gegensatz zur Bibliografischen Kopplung kann der Wert der Kozitation zweier Publikationen mit der Zeit steigen. Durch Referenzierungen in neu erscheinenden Publikationen entstehen weitere Kozitations-Paare.

Für Publikationen, deren Veröffentlichungsdatum noch nicht weit zurückliegt, kann über die Kozitation häufig kein Zusammenhang zu anderen Publikationen festgestellt werden. Der Grund hierfür ist, dass neue Publikationen noch nicht gelesen wurden und somit die Möglichkeit einer Referenzierung gering ist.

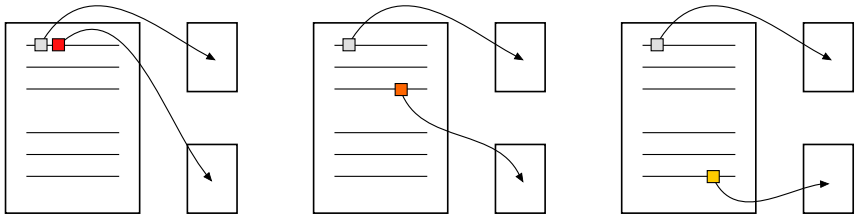
Ein Beispiel zur Anschauung von Bibliografischer Kopplung und Kozitation zeigt Abb. 3. Die entsprechenden Werte dazu sind in Tabelle 1 und Tabelle 2 aufgeführt.

Tabelle 1: Stärken der Bibliografischen Kopplung für das Beispiel in Abb. 3

| | B | C | D |
|---|---|---|---|
| A | 2 | 1 | 0 |
| B | · | 1 | 0 |
| C | · | · | 0 |

Tabelle 2: Stärken der Kozitation für das Beispiel in Abb. 3

| | B | C | D |
|---|---|---|---|
| A | 0 | 0 | 0 |
| B | · | 1 | 1 |
| C | · | · | 2 |



(a) Starke Gewichtung: Gleicher Satz (b) Mittlere Gewichtung: Gleicher Absatz (c) Schwache Gewichtung: Gleiches Dokument

Abbildung 4: Beispiele für Distanzen von Zitationen

### 2.1.4 *Distanz von Zitationen*

Die hier beschriebene Distanz von Zitationen wurde unter den Namen „in-text citation distance analysis (ICDA)“ [GBH09] und „Citation Proximity Analysis (CPA)“ [GB09] veröffentlicht.

Die Analyse der Distanz von Zitationen berücksichtigt die Positionierung von Zitationen innerhalb eines Textes. Dabei wird die Struktur des Textes analysiert, indem der Text in Distanz-Klassen unterteilt wird. Beispiele für Distanz-Klassen sind Satz, Absatz oder Kapitel. Für alle Paare von Zitationen innerhalb eines Dokumentes wird untersucht, welche Distanz-Klassen einem Paar zugewiesen werden können. Gewählt wird diejenige, die der geringsten Distanz eines Paares entspricht. Beispielsweise befinden sich zwei Zitationen, die im gleichen Satz auftreten, auch im gleichen Kapitel. In diesem Fall ist die Klasse *Satz* die Distanz-Klasse mit der geringeren Distanz.

Bei der hier beschriebenen Metrik wird angenommen, dass bei Zitationen mit geringer Distanz auf einen starken semantischen Zusammenhang der referenzierten Dokumente zurückgeschlossen werden kann. Dies können beispielsweise Publikationen sein, die ein spezielles Problem behandeln und deshalb im gleichen Satz aufgezählt werden.

Die Verwendung von Distanzen von Zitationen als Indikator für die Stärke des thematischen Zusammenhangs wurde bereits analysiert. In einer Untersuchung von 22 Zeitschriften wurde das Vorkommen von Distanzen der Klassen Dokument, Abschnitt, Absatz und Satz erfasst. Die Analyse zeigt, dass der prozentuale Anteil von Kozitationen der Klasse Satz mit ansteigendem Koziations-Wert steigt. Je häufiger zwei Publikationen gemeinsam referenziert werden, desto häufiger geschieht dies im gleichen Satz. Dies spricht dafür, dass Kozitationen der Distanz-Klasse Satz potenziell effizienter für die Kozitations-Analyse sind als Kozitationen anderer Klassen [LC12].

Tabelle 3: Gewichtung des DSI

| Auftreten | Wert |
|---|---|
| Satz | 1 |
| Absatz | 1/2 |
| Abschnitt | 1/4 |
| Kapitel | 1/8 |
| Sonstiges | 1/16 |

Tabelle 4: Gewichtung des CPI

| Auftreten | Wert |
|---|---|
| Satz | 1 |
| Absatz | 1/2 |
| Abschnitt | 1/4 |
| Gleiche Zeitschrift / gleiches Buch | 1/8 |
| Gleiche Zeitschrift, aber andere Auflage | 1/16 |

Beispiele für drei Zitations-Paare sind in Abbildung 4 aufgeführt. In jedem der drei Fälle sind jeweils zwei Dokumente referenziert. In allen Fällen hat eine der beiden Zitierungen die gleiche Position. Die Position der zweiten Zitierung unterscheidet sich je nach Fall, sodass hier eine Aufteilung in drei verschiedene Distanz-Klassen möglich ist. Im ersten Fall (Abb. 4a) werden zwei Dokumente im gleichen Satz referenziert. Aus der geringen Distanz der Zitationen wird auf einen starken Zusammenhang zurückgeschlossen. Es ergibt sich dementsprechend eine starke Gewichtung. Die Zitationen im zweiten Fall (Abb. 4b) befinden sich im gleichen Absatz. Die Distanz ist hier größer als im ersten Fall, daher fällt die Gewichtung geringer aus. Im dritten Fall (Abb. 4c) befinden sich die beiden gezeigten Zitationen im gleichen Dokument, die Distanz ist aber im Vergleich zu den anderen beiden Fällen am größten. Dementsprechend wird dieser Zusammenhang schwach gewichtet.

Für die Datenhaltung der Distanzen werden Indizes verwendet. Der „Distance Similarity Index (DSI)“ [GBH09] sowie der „Citation Proximity Index (CPI)“ [GB09] verwenden je nach Distanz-Klasse für jedes Zitationspaar einen Wert im Intervall $(0, 1]$. Die verwendeten Werte sind für die Distanz-Klassen *Satz*, *Absatz* und *Abschnitt* bei beiden Indizes identisch. Im DSI (siehe Tabelle 3) werden als weitere Distanz-Klassen *Kapitel* und *Sonstiges* verwendet, im CPI (siehe Tabelle 4) wird danach gewichtet, ob Publikationen in der *gleichen Zeitschrift* veröffentlicht wurden.

Neben der Verwendung von Distanz-Klassen besteht die Möglichkeit, Positionsangaben zu nutzen. Diese Angaben können sich auf die Position von Zeichen oder Bytes beziehen und die absoluten oder relativen Positionen verwenden. Abstände von Positionen können zusätzlich normalisiert werden, indem ihr Verhältnis zur Gesamtgröße eines Dokuments betrachtet wird. Diese Variante wurde bei einem Clustering von Kozitationen genutzt [BSK12].

Die in diesem Abschnitt beschriebenen Metadaten zu wissenschaftlichen Publikationen liegen sowohl bei gedruckten Dokumenten als auch in der digitalen Variante vor. Bei letzterer kann die Möglichkeit einer Extraktion genutzt werden. Dies ist Thema des folgenden Abschnitts.

## 2.2 EXTRAKTION VON METADATEN

Dieser Abschnitt gibt zunächst einen Überblick der verschiedenen Extraktionsmodelle. Anschließend werden die Extraktionsanwendungen ParsCit und GROBID vorgestellt.

Die automatische Extraktion von Metadaten ist eine extrem schwierige Aufgabe. Die Herausforderung entsteht unter anderem durch die vielen Variationen der Verwendung von Trennzeichen [DTS+07]. Dabei hängt die Qualität von Suchmaschinen für Publikationen sehr von den verwendeten Extraktionskomponenten ab [PM04].

### 2.2.1 *Extraktionsmodelle*

Man kann zwischen zwei Kategorien der Extraktion von Metadaten unterscheiden: Template-basierte Lösungen und Ansätze des maschinellen Lernens [CGK08]. Bei Template-basierten Ansätzen werden zunächst Vorlagen für mögliche syntaktische Zusammensetzungen von Referenzen definiert. Anschließend wird für jede zu analysierende Referenz geprüft, welche Vorlage am besten zur zu prüfenden Zeichenkette passt und abschließend die Extraktion durchgeführt.

Für das maschinelle Lernen werden zunächst Modelle trainiert. Dazu werden als Eingabe mögliche Zeichenketten einer Klasse zugeordnet. Beim Training werden solche ausgezeichneten Zeichenketten als Eingabe genutzt. Basierend auf einem trainierten Modell können anschließend für eine zu analysierende Referenz Teile Labels zugeordnet werden.

Im Vergleich von Template-basierten Ansätzen und maschinellem Lernen wird beobachtet, dass erstere bessere Ergebnisse erzielen. Dies zeigt z.B. der Vergleich von ParsCit (Präzision: 95,7 [CGK08]) und ParaTools (Präzision: 30% - 79% [HHKL04]) bzw. BLAST (Präzision: 89% [HHKL04]).

Im Bereich des maschinellen Lernens haben sich drei verschiedene Techniken entwickelt: Hidden Markov Models (HMM), Support Vector Machines (SVM) und Conditional Random Field (CRF).

Der CRF Ansatz hat sich dabei als vielversprechendster herausgestellt. Im Vergleich der drei Ansätze konnte bei CRF eine signifikante Reduktion von Fehlern gezeigt werden [PM04]. Dabei wurden sowohl

Metadaten aus Kopfteilen von Publikationen als auch Referenzangaben, wie sie in Literaturverzeichnissen vorkommen, geprüft.

Im Folgenden werden zwei Anwendungen zur Extraktion vorgestellt, die auf dem CRF Ansatz basieren.

### 2.2.2 *ParsCit*

ParsCit [CGK08] ist eine Open Source Software zur Extraktion von Referenzen aus dem Jahr 2008. Mit der Software können Metadaten und Referenzen aus Volltexten wissenschaftlicher Publikationen extrahiert werden. Dabei werden bis zu 13 verschiedene Felder gefunden: die Autoren, der Titel der Publikation, der Buchtitel, das zugehörige Datum, Angaben zum Fachbericht, der Herausgeber, aufgeführte Institutionen, Notizen, der Ort, Seitenzahlen, der Verlag, der Band und der Zeitschriftentitel. Die Software wird unter [ParsCit] bereitgestellt.

Im ParsCit Ansatz werden Heuristiken zur Textanalyse mit Methoden des maschinellen Lernens kombiniert. Um einen Einblick in die Arbeitsweise zu geben, ist der Extraktionsvorgang im Folgenden kurz beschrieben.

Zunächst wird in der Vorbereitungsphase nach Variationen von Zeichenketten wie „References“ (Referenzen) oder „Bibliography“ (Bibliografie) im hinteren Teil des Textes gesucht. Dadurch wird die Position der Referenzen einer Publikation ausgemacht und anschließend einzelne Referenzen segmentiert.

An dieser Stelle liegen einzelne Referenzen vor. Um die einzelnen Bestandteile einer Referenz zu bestimmen, bedienen sich die Autoren des maschinellen Lernens. Dafür wird ein CRF genutzt, mit dem ein Modell zunächst erstellt und später für die Extraktion genutzt werden kann. Für das Erstellen des Modells verwenden die Autoren gekennzeichnete Trainingsdaten. Es werden also z.B. verschiedene Namen als eben solche ausgezeichnet und einem Modell hinzugefügt. Das so trainierte Modell kann verwendet werden, um bisher unbekannte Zeichenketten einer der weiter oben aufgeführten 13 Klassen zuzuordnen. Eine Referenz wird also in kleinere Einheiten zerlegt, die anschließend „gelabeled“ werden. Dabei wird die Groß- und Kleinschreibung, Zeichensetzung und die Lage einer Zeichenkette berücksichtigt. Außerdem werden Wörterbücher eingesetzt, um Vor-, Nach- und Ortsnamen, sowie Verlage oder Monate zu erkennen.

Nachdem verschiedene Bestandteile der Referenzen erkannt wurden, schließt sich eine Nachbereitung an. In dieser werden die einzelnen Felder normalisiert. Dabei wird etwa das Format von Seitenzahlen vereinheitlicht oder die Repräsentation und Reihenfolge einzelner Bestandteile von Autorennamen nachbearbeitet. Der gesamte Volltext

wird abschließend durchlaufen und via regulärer Ausdrücke nach Markern von Zitaten (z.B. „[1]“) durchsucht. Gefundene Marker werden genutzt, um 200 Zeichen vor und nach der Markerposition zu extrahieren. Diese Zeichen werden dem XML-Ausgabeformat hinzugefügt und damit ein Kontext zur zitierenden Textstelle bereitgestellt.

### 2.2.3 *GROBID*

GROBID: GeneRatiOn of BIbliographic Data [Lop09] ist eine Software zur Extraktion von Metadaten und Referenzen aus wissenschaftlichen Publikationen und Dokumenten ähnlichen Inhalts. Für die Entwicklung von GROBID wird derzeit die Git-Versionsverwaltung genutzt. Die Software kann unter [GRO-Git] kostenlos bezogen werden. Eine weitere, jedoch nicht mehr aktuelle Quelle ist die Subversion-Versionsverwaltung, die unter [GRO-SVN] erreichbar ist.

Die Erkennung von Bestandteilen von Referenzen ist mit CRF umgesetzt worden. Neben der Extraktion aus wissenschaftlichen Publikationen kann GROBID ebenso für die Extraktion aus Patentdokumenten genutzt werden [LR10]. Außerdem besteht die Möglichkeit, Volltexte zu analysieren und daraus Stichwortlisten ableiten zu lassen [Lop10].

GROBID bietet drei Ausgabeformate an: BibTeX, ein Programm zur Literaturverwaltung in LaTeX, COinS [COinS], eine Spezifikation für bibliometrische Angaben in HTML und TEI [TEI], einen Standard für Textrepräsentationen. Eine Demonstration der Software wird unter [GRO-Demo] bereitgestellt.

Die in diesem Abschnitt vorgestellten Anwendungen zur Extraktion finden in verschiedenen Systemen zur Findung von thematisch verwandten Publikationen Verwendung. Einige dieser Systeme werden im Folgenden vorgestellt.

## 2.3 BESTEHENDE LÖSUNGSANSÄTZE

In den Bereichen Zitationsdatenbanken, Zitationsanalyse und Recommendations für wissenschaftliche Publikationen entstanden in der Vergangenheit verschiedene Lösungsansätze. In diesem Abschnitt werden zunächst die nicht-kommerziellen Lösungen CiteSeer und Scienstein vorgestellt, auf deren Ergebnisse zurückgegriffen werden soll. Abschließend werden weitere nennenswerte Projekte beschrieben.

*CiteSeer* [GBL98] war eine der ersten Zitationsdatenbanken, deren Datenbestand durch das Durchsuchen des World Wide Web und das Indizieren von Publikationen automatisch und autonom erweitert wird. Publikationen im Postscript Format wurden in Reintext konvertiert

und Metadaten sowie Referenzen extrahiert, um eine Zitationsanalyse durchzuführen. Um die Performance zu verbessern, wurde das über 8 Jahre alte System durch *Next Generation CiteSeer* [LCB+06] ersetzt. Verändert wurden das Datenmodell und die Architektur, um eine bessere Skalierbarkeit und eine höhere Flexibilität zu erreichen. Nutzern der Webseite wird durch die Eingabe eines Suchtexts die Möglichkeit geboten, Metadaten von Publikationen zu durchsuchen. Für eine Publikation wird eine Liste von weiteren Publikationen angezeigt, die nach Anzahl von Kozitationen sortiert ist.

Im *Scienstein* Recommender System [GBH09] wird eine Kombination verschiedener Ansätze vorgeschlagen. Einen Teil bildet die Zitationsanalyse mit einfachen Relationen von zitierenden und zitierten Publikationen sowie den darauf aufbauenden Metriken Kozitation und Bibliografische Kopplung. Zusätzlich wird festgehalten, wie oft ein Dokument zitiert wird (*in-text citation frequency analysis*, ICFA) und wie groß diese Häufigkeit im Verhältnis zu allen Zitaten ist (*In-text Impact Factor*, ItIF). Außerdem wird miteinbezogen, wie nah zwei Zitate im Text beieinanderstehen (*in-text citation distance analysis*, ICDA) und daraus eine Metrik für die Ähnlichkeit zweier Publikationen abgeleitet (*Distance Similarity Index*, DSI). Weiterhin werden Notizen von Anwendern des Systems analysiert, um weitere Vorschläge für Publikationen zu generieren.

Neben diesen Systemen existieren diverse kommerzielle Lösungen. Zu nennen sind die Suchvariante Google Scholar [Scholar] und Microsoft Academic Search [MSAS].

Neuere Entwicklungen sind die Literaturverwalungs-Software Mendeley [Mendeley], deren Online-Suche Zugriff auf Publikationen ermöglicht und das soziale Netzwerk ResearchGate [RGate].

## 2.4 TECHNISCHE GRUNDLAGEN

In diesem Abschnitt werden das *MapReduce*-Programmiermodell, das Framework für verteilte Berechnungen *Hadoop* sowie die verteilte Datenbank *HBase* vorgestellt. Die drei entsprechenden Abschnitte bauen jeweils aufeinander auf.

### 2.4.1 *MapReduce*

MapReduce [DG04] ist ein Programmiermodell zur Verarbeitung großer Datenmengen. Es wurde von Google entwickelt und entstand aus der Notwendigkeit, große Mengen von Daten in möglichst kurzer Zeit, und verteilt auf Hunderten oder Tausenden von Prozessoren zu verarbeiten. Dabei flossen Anregungen aus der funktionalen Programmierung in die Entwicklung des Programmiermodells ein. Mit

MapReduce wird versucht, ein skalierbares System zur Verfügung zu stellen, dessen Leistung linear mit der Anzahl hinzugefügter Maschinen steigt [Geo11, S. 289][2].

Mit der Verwendung von MapReduce bekommen Entwickler die Möglichkeit, durch die Implementierung zweier Methoden, *map* und *reduce*, eine automatische Verteilung und parallele Verarbeitung in Rechnerclustern zu nutzen. Die Komplexität des parallelen Rechnens wird dabei verborgen. Entwicklern wird die Umsetzung der Parallelisierung, Lastbalancierung und Fehlertoleranz von Berechnungen abgenommen.

Für das Programmiermodell muss eine vorgegebene Schnittstelle implementiert werden. Diese besteht aus den Funktionen *map* und *reduce*, deren Eingabe- und Ausgabeparameter folgendermaßen definiert sind:

$$\begin{aligned} &map(Schluessel_1, Wert_1) \rightarrow Liste[(Schluessel_2, Wert_2)] \\ &reduce(Schluessel_2, Liste[Wert_2]) \rightarrow Liste[Wert_2] \end{aligned} \tag{1}$$

Die map-Funktion erwartet ein Schlüssel-Wert-Paar als Eingabe. Dieses wird zur Berechnung von einer Liste aus neuen Schlüssel-Wert-Paaren verwendet. Wie an den Typen der Parameter zu sehen ist, werden die Domänen von $Schlüssel_2$ und $Wert_2$ für die anschließende reduce-Funktion wiederverwendet. Für jeden Schlüssel, der einer reduce-Funktion übergeben wird, enthält die zugehörige Liste alle Werte, die zuvor von map-Funktionen berechnet wurden. In der MapReduce-Publikation [DG04, S. 4] ist beschrieben, dass der Rückgabewert der reduce-Funktion einer finalen Ausgabedatei angehängt wird. Da der jeweilige Schlüssel in der reduce-Funktion bekannt ist, ist es auch möglich, die finale Liste in Verbindung des jeweiligen Schlüssels abzuspeichern.

Ein Beispiel für eine Wortzählung zeigt Abb. 5. Im Beispiel sei zunächst eine Textdatei gegeben. Diese wird in ihre Zeilen aufgeteilt. Für die Map-Phase werden die Zeilen als Tupel (Startposition, Zeichenkette) an Map-Komponenten übergeben. Diese führen die Methode *map* aus, in der zunächst alle Satzzeichen aus der Zeichenkette entfernt und Großbuchstaben durch Kleinbuchstaben ersetzt werden. Die daraus entstehende Zeichenkette wird an jedem Leerzeichen getrennt. Die resultierenden einzelnen Worte werden nacheinander mit ihrer Anzahl (im Beispiel jeweils 1) einer Liste hinzugefügt. Diese Liste mit Tupeln der Form (Wort, Anzahl) wird von der map-Methode zurückgegeben. Die anschließende Zuweisung der Rückgabewerte zu Reduce-Komponenten geschieht über eine *Partitionierungsfunktion*. Dies ist üblicherweise eine Hashfunktion, die die entsprechende Domäne gleichmäßig den verfügbaren Reduce-Komponenten zuteilt. Im

2 Übersetzung des Autors

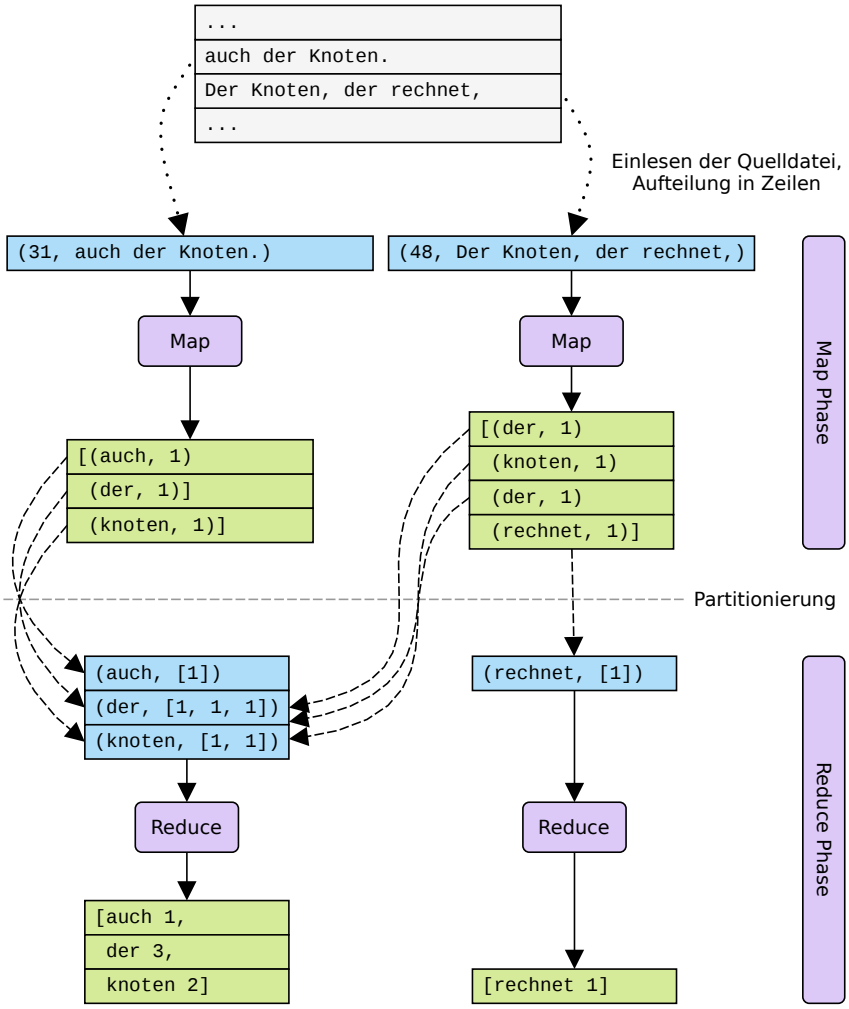



Abbildung 5: MapReduce-Beispiel für eine Wortzählung

Beispiel wurde eine Aufteilung A–M und N–Z für die zwei zur Verfügung stehenden Reduce-Komponenten gewählt. In der Reduce-Phase werden der Methode *reduce* Tupel der Form $(Wort, Liste[Anzahl])$ übergeben. Innerhalb der reduce-Methode werden die Werte addiert und als $Liste[Zeile]$ zurückgegeben. Diese Zeilen können abschließend als Datei gespeichert werden. Damit ist das Beispiel zur Zählung von Worten abgeschlossen.

Ein Framework, dass das MapReduce Programmiermodell umsetzt ist Hadoop. Dieses wird im folgenden Abschnitt beschrieben.

### 2.4.2 *Hadoop*

Apache Hadoop ist ein Open Source System, das eine Analyse mittels MapReduce und eine zuverlässige, verteilte Datenhaltung bereitstellt [Whi12, S. 4]. Hadoop kann unter [Hadoop] bezogen werden.

Durch den Einsatz von Hadoop kann eine Vielzahl unterschiedlicher Probleme behandelt werden. Beispiele hierfür sind Entwurfsmuster, die zur Behandlung der Probleme angewendet werden können. Es existiteren u.a. Entwurfsmuster zur Zusammenfassung, Filterung, Organisation und dem Zusammenfügen von Daten [MS13].

Die Signaturen der Hadoop map- und reduce-Methoden ist überwiegend mit den Signaturen aus der MapReduce Publikation (siehe Gleichung 1) identisch. Lediglich die Ausgabe der reduce Funktion weicht ab; dort ist ein weiteres Schlüssel-Wert-Paar definiert [Whi12, S. 223]. Dies verdeutlicht, dass sowohl die Ausgabe der map-Funktion, als auch die der reduce-Funktion, unabhängig von ihrer Eingabe verwendet werden können:

$$\begin{aligned} &map(Schluessel_1, Wert_1) \rightarrow Liste[(Schluessel_2, Wert_2)] \\ &reduce(Schluessel_2, Liste[Wert_2]) \rightarrow Liste[(Schluessel_3, Wert_3)] \end{aligned} \tag{2}$$

Das MapReduce-Programmiermodell ist in Hadoop durch den drei Phasen *Map*, *Shuffle* und *Reduce* umgesetzt. In einem MapReduce-Ablauf stehen im Regelfall eine Reihe von Prozessoren zur Verfügung, auf denen mehrere Instanzen von Map- und Reduce-Komponenten parallel ausgeführt werden. Abbildung 6 zeigt ein Beispiel, bei dem jeweils drei Map- und drei Reduce-Komponenten verwendet werden. Das Beispiel zeigt den Datenfluss zwischen und innerhalb der drei Phasen. Zunächst wird eine Eingabe gelesen. Eingaben können aus unterschiedlichen Quellen stammen und in verschiedenen Formaten vorliegen. Beispielsweise sind Text- oder Binärdateien aus verteilten Dateisystemen oder Datenbanken möglich. Die Eingabe wird im ersten Schritt in *Splits* geteilt (Abb. 6-a). Ein Split besteht aus einer Sammlung von Schlüssel-Wert-Paaren. Jeder Map-Komponente

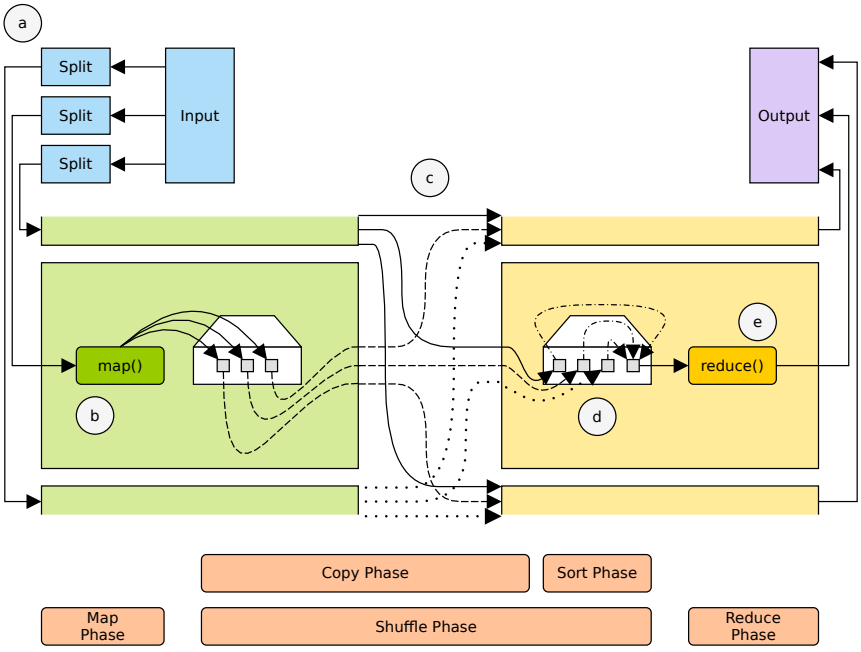



Abbildung 6: Hadoop Phasen und Datenfluß

wird ein Split als Eingabe zugewiesen. In der Map-Phase wird die Eingabe als Argumente für Aufrufe der map-Methode (siehe Gleichung 2) genutzt (Abb. 6-b). Die Ausgaben werden in Dateien geschrieben, die später Reduce-Komponenten zugewiesen werden. Die Dateien setzen sich aus einer Menge von Schlüssel-Wert-Paaren zusammen, die in Reihenfolge ihrer Schlüssel geordnet sind. Die Sortierung kann als Teil der folgenden Shuffle-Phase gesehen werden (Abb. 6-c). Die parallelen Berechnungen der Map-Komponenten können zu unterschiedlichen Zeitpunkten beendet sein. Sobald eine Berechnung fertiggestellt ist, beginnt der Kopiervorgang von Map- zu Reduce-Komponenten. Diese können auf unterschiedlichen Rechnern im Cluster ausgeführt werden. In der Copy-Phase werden also Netzwerkressourcen beansprucht. Der zweite Teil der Shuffle-Phase ist die Sort-Phase. In dieser werden die einzelnen Ausgabedateien von Map-Kompontenen zu jeweils einer Eingabedatei für die anschließende Reduce-Phase zusammengefasst (Abb. 6-d). Dies beinhaltet erneut eine Sortierung der vorhandenen Schlüssel-Wert-Paare. Die in reduce-Methoden (Abb. 6-e) berechneten Ausgaben werden abschließend gespeichert. Für die Speicherung können wie bei der initialen Eingabe verschiedene Datenformate und Datenhaltungssysteme verwendet werden.

Die Hadoop Java API für MapReduce wurde in der Version 0.20 erneuert. Ab dieser Version stehen zwei APIs zur Verfügung, die inkompatibel zueinander sind. Dabei kann die alte API weiterverwendet werden, sie gilt aber als veraltet („deprecated“) [Had-0.20]. Dies

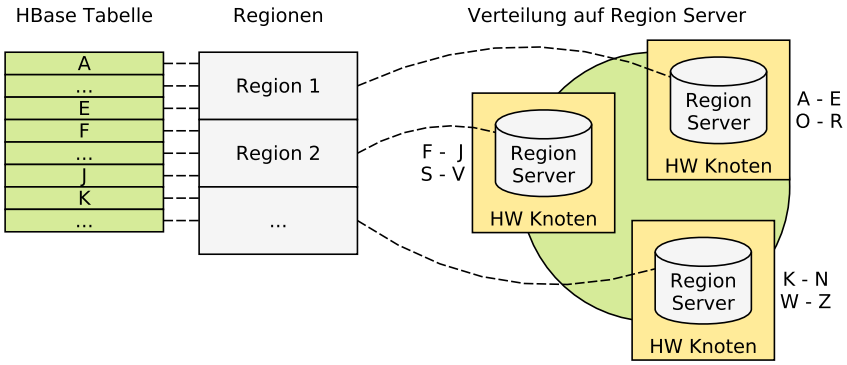



Abbildung 7: Verteilte Datenhaltung in HBase

hat zufolge, dass Code, bei dem die alte API genutzt wird, umgeschrieben werden muss, um Features zu nutzen, die bei der Weiterentwicklung der neuen API entstehen.

Ein Hadoop-Feature, von dem für die Analysen dieser Arbeit Gebrauch gemacht wird, sind *Hadoop-Counter*. In Mapper- und Reducer-Komponenten kann auf ein Kontext-Objekt der Verarbeitung zugegriffen werden. Dieses stellt Methoden zur Verfügung, mit denen für frei wählbare Schlüssel ein initialer Wert von 0 inkrementiert werden kann. Dadurch können innerhalb der verteilten Verarbeitung Zählervariablen manipuliert werden, deren Werte während der Ausführung überwacht werden können.

Mit Hadoop steht das verteilte Dateisystem HDFS zur Verfügung. Daten in HDFS werden automatisch repliziert. Damit wird einem Datenverlust bei Ausfällen von Hardware vorgebeugt. Von diesem Feature wird auch von der verteilten Datenbank HBase Gebrauch gemacht.

### 2.4.3 *HBase*

„HBase ist ein verteiltes [...] Datenhaltungs-System mit fast-optimaler Schreib- [...] und exzellenter Leseperformance“[3] [Geo11, S. 29]. Dabei zählt HBase zu den nichtrationalen Datenbanken und wurde nach dem Vorbild von Googles Datenbank BigTable [CDG+06] gestaltet. Es ermöglicht einen Schlüssel-basierten Zugriff, mit dem auch auf ganze Schlüsselintervalle zugegriffen werden kann. HBase ist OpenSource und kann unter [HBase] bezogen werden.

Abb. 7 zeigt das Prinzip der verteilten Datenhaltung in HBase. Es werden Tabellen genutzt, deren Zeilen über Schlüssel (im folgenden *Rowkeys* genannt) identifiziert werden. Zeilen werden in Regionen zusammengefasst. Diese Regionen werden automatisch und für den Be-

3 Übersetzung des Autors

nutzer transparent gesplittet, wenn ihre Größe einen Schwellenwert überschreitet. Regionen werden auf RegionServer (Rechner im Cluster) gespeichert. Dabei ist eine Region einem RegionServer zugeteilt, ein RegionServer kann dagegen für mehrere Regionen zuständig sein. In Abb. 7 ist beispielsweise eine Verteilung von Rowkeys *A* bis *Z* dargestellt. Dabei bilden die Intervalle A-E, F-J, K-N, O-R, S-V und W-Z einzelne Regionen. Diese sind auf die RegionServer verteilt.

Datensätze (Zellen) in HBase werden über die Kombination der *Rowkeys* und drei weiteren Bezeichnern adressiert. Diese Bezeichner legen die zu verwendene Tabelle, die *ColumnFamily* und den *ColumnQualifier* fest. Für ColumnFamilies können Eigenschaften wie die maximale Anzahl unterschiedlicher Versionen oder eine Kompression festgelegt werden. ColumnQualifier stellen eine weitere Aufteilung der ColumnFamilies dar. Die Notation zur Adressierung erfolgt durch Trennung eines Doppelpunktes in der Form columnfamily:columnqualifier.

# 3 ANALYSE

Dieses Kapitel enthält Untersuchungen und deren Ergebnisse, auf die in der weiteren Arbeit zurückgegriffen wird. Zunächst wird in Abschnitt 3.1 untersucht, welche Metadaten aus PDF-Dateien extrahiert werden können, um Publikationen mit diesen zu identifizieren. Abschnitt 3.2 beschreibt die Beschaffenheit extrahierter Texte und XML-Dateien und die daraus entstehenden Möglichkeiten, um Distanzen von Zitationen zu ermitteln. Zuletzt werden in Abschnitt 3.3 drei verschiedene Lösungen zur Datenhaltung verglichen, von denen eine zur Speicherung und Abfrage von Beziehungen innerhalb von Dokumentennetzwerken genutzt werden soll.

## 3.1 METADATEN ZUR IDENTIFIZIERUNG VON PUBLIKATIONEN

Dieser Abschnitt führt in die Verwendung von Metadaten zu wissenschaftlichen Publikationen ein. Anschließend wird analysiert, welche Metadaten durch eine automatische Extraktion aus PDF-Dateien erschlossen werden können. Basierend darauf findet eine Auswahl von Metadaten statt, die sich zur Identifizierung von wissenschaftlichen Publikationen anbieten.

### 3.1.1 *Metadaten zu wissenschaftlichen Publikationen*

Metadaten eignen sich zur Beschreibung von wissenschaftlichen Publikationen und zu ihrer Identifizierung, falls genaue Einzelangaben zur Verfügung stehen. Je nach Umfang und Beschaffenheit der Metadaten können sie damit einen wichtigen Bestandteil zur Filterung in großen Dokumentenmengen ausmachen. Dies setzt voraus, dass Konventionen für ihre Verwendung genutzt werden. Beispielsweise sollten Namen von XML-Elementen konstant verwendet werden.

Eine Alternative zur Identifizierung über Metadaten ist die Verwendung von Identifikatoren. Ein Beispiel für einen weitverbreiteten Typ von Bezeichnern zur Identifizierung sind Uniform Resource Identifiers (URIs). Eine Untermenge hiervon bilden Uniform Resource Locators (URLs), die einen primären Zugriffsmechanismus beschreiben, um eine Ressource zu identifizieren[1] [URL]. URLs können verwendet wer-

1 Sinngemäße Übersetzung des Autors

den, um Web-Ressourcen zu adressieren und einen erneuten Datenabruf zu ermöglichen, sofern die Ressource erhalten bleibt.

Für wissenschaftliche Publikationen kann das Digital Object Identifier (DOI) System zur Identifizierung genutzt werden. Durch dieses wird eine „technische und soziale Infrastruktur für die Registrierung und Nutzung von persistenten, kompatiblen Identifikatoren zur Nutzung über digitale Netzwerke [bereitgestellt]“[2] [DOI]. Ist eine DOI zu einem Dokument bekannt, besteht dadurch i.d.R. die Möglichkeit, weitere Metadaten zum betreffenden Dokument zu beziehen. Häufig wird auch eine Bezugsmöglichkeit für das Dokument selbst zur Verfügung gestellt.

Nicht für alle wissenschaftlichen Publikationen existiert eine DOI. Für Ausschnitte aus Zeitschriften und Büchern besteht die Möglichkeit einer Identifizierung über Standards wie International Standard Book Number (ISBN) oder International Standard Serial Number (ISSN). Allerdings sind dem Leser einer Publikation in manchen Fällen nur der dort aufgeführte Titel und die Autoren bekannt.

Autoren wissenschaftlicher Publikationen stellen im Regelfall ein Literaturverzeichnis zur Verfügung, das Verweise zu zierten Dokumenten enthält. Einträge dieses Verzeichnisses bestehen aus den wichtigsten Randdaten zu den referenzierten Dokumenten. Stehen Publikationen zudem in einem digitalen Format zur Verfügung, können Software-Anwendungen genutzt werden, um Metadaten zu Publikationen und ihren Referenzen zu extrahieren. Darauf aufbauend besteht die Möglichkeit, die erschlossenen Metadaten zu verwenden, um Dokumente zu identifizieren.

### 3.1.2 *Extraktion von Metadaten*

Mit dieser Arbeit entsteht ein System, das eine automatisierte Verarbeitung von wissenschaftlichen Publikationen bereitstellt. Dies beinhaltet die Extraktion von Metadaten und eine Identifizierung von mehrfach vorkommenden Publikationen. Im Folgenden soll im Vorfeld untersucht werden, welche Metadaten nach einer automatischen Extraktion zur Verfügung stehen.

Zur Analyse verwendbarer Metadaten wurden aus PDF-Dateien Volltexte extrahiert. Dazu wurden auf den verfügbaren Datenbestand aus den Arbeiten von Tobias Varlemann und Nicolas Schelp [Var13, Schck] zurückgegriffen und weitere Daten beschafft [PubUPB]. Es stehen damit Daten zu 571.897 Publikationen zur Verfügung.

Die Daten wurden als Eingabe für die Softwarepakete ParsCit und GROBID verwendet. Dadurch wurden XML-Dateien generiert, die in

2 Übersetzung des Autors

drei verschiedene Typen unterteilt sind. ParsCit XML-Dateien enthalten Metadaten zum jeweils verwendeten Eingabedokument und zu den im Dokument enthaltenen Referenzen. Durch Verwendung des GROBID-Softwarepaketes werden zwei Datentypen generiert: XML-Dateien mit Metadaten zum Eingabedokument und XML-Dateien für Einträge der Literaturverzeichnisse. Insgesamt stehen nach der Extraktion diese Daten zur Verfügung:

- 571.897 PDF-Dateien als Eingabe
- 498.904 ParsCit Datensätze
- 321.386 GROBID Datensätze mit Metadaten zu Publikationen
- 134.279 GROBID Datensätze mit Metadaten zu Listen mit Referenzen

Die unterschiedliche Größe zwischen diesen Datentypen hat mehrere Ursachen. Die Dokumentensammlung besteht hauptsächlich aus wissenschaftlichen Publikationen. Vereinzelt befinden sich darunter auch PDF-Dateien, die andere Formate, z.B. Präsentationen, beinhalten. Weitere Ursachen für die Unterschiede in der Anzahl sind Zeichenkodierungen und verwendete natürliche Sprachen, durch deren Verwendung keine Extraktion stattfinden kann.

Für die Analyse der Daten wurde im Rahmen dieser Arbeit ein SAX-Parser implementiert. Mit diesem wird eine XML-Datei durchlaufen und eine Liste mit allen gefundenen XML-Elementen erstellt. Zunächst wird für eine XML-Datei nur festgehalten, dass ein Element existiert und nicht, wie oft es auftritt. Das Hadoop-Framework wurde anschließend verwendet, um die Ausführung von Instanzen des SAX-Parsers zu parallelisieren. Dabei wurde jede der verfügbaren XML-Dateien geparst. Zur Aggregation der XML-Elemente wurden Hadoop-Counter genutzt. Damit wurde für jedes XML-Element die Anzahl der Dateien festgestellt, die das Element verwenden.

Die Ergebnisse der Extraktion mittels ParsCit sind in Tabelle 5 (Metadaten zu Publikationen, auf Seite 26) und in Tabelle 6 (Metadaten zu Referenzen) abgebildet. Die Ergebnisse der GROBID-Durchläufe zeigen Tabelle 7 und Tabelle 8 (auf Seite 27). Die Tabellenzeilen enthalten das jeweilige XML-Element (*Schlüssel*), die Anzahl der Dateien, in denen der Bezeichner gefunden wurde, und das Vorkommen des Schlüssels im Verhältnis zur Größe der Eingabe, bestehend aus 571.897 Einträgen.

Aus den Einträgen der Tabellen 5 und 7 kann entnommen werden, in welchem Verhältnis die jeweiligen Schlüssel in den Eingabedateien vorkommen. Mit Einträgen der Tabellen 6 und 8 ist dies auch für Listen von Referenzen möglich. Für einzelne Referenzen ist dies jedoch nicht repräsentativ. Die Eingabedateien repräsentieren Listen, die eine Vielzahl von Referenzen enthalten können. Ein Anteil von 84,74 % für das Titel-Element bedeutet, dass in 84,74 % aller Listen

Tabelle 5: ParsCit Metadaten zu Publikationen bei 571.897 Datensätzen

| Schlüssel | Anzahl | Anteilig |
|---|---|---|
| abstract | 471.179 | 82,39 % |
| note | 440.625 | 77,05 % |
| title | 428.654 | 74,95 % |
| author | 413.186 | 72,25 % |
| affiliation | 267.387 | 46,75 % |
| address | 235.508 | 41,18 % |
| keyword | 165.359 | 28,91 % |
| intro | 151.910 | 26,56 % |
| email | 110.880 | 19,39 % |
| date | 93.908 | 16,42 % |
| phone | 58.004 | 10,14 % |
| pubnum[a] | 10.863 | 1,9 % |
| web | 8.623 | 1,51 % |
| degree[b] | 5.183 | 0,91 % |
| page | 57 | 0,01 % |

*a* Enthält in den meisten Fällen ganze Zahlen

*b* Zeichenketten mit Angaben wie z.B. „Dr“,„Professor“ oder „President“

Tabelle 6: ParsCit Metadaten zu Referenzen bei 571.897 Datensätzen

| Schlüssel | Anzahl | Anteilig |
|---|---|---|
| rawString[a] | 498.059 | 87,09 % |
| marker[b] | 498.059 | 87,09 % |
| author | 496.828 | 86,87 % |
| date | 496.455 | 86,81 % |
| title | 484.633 | 84,74 % |
| journal | 476.676 | 83,35 % |
| pages | 476.219 | 83,27 % |
| volume | 472.923 | 82,69 % |
| location | 414.174 | 72,42 % |
| context[c] | 392.327 | 68,60 % |
| booktitle | 386.321 | 67,55 % |
| publisher | 326.216 | 57,04 % |
| institution | 180.932 | 31,64 % |
| note | 165.410 | 28,92 % |
| issue | 144.132 | 25,20 % |
| editor | 122.274 | 21,38 % |
| tech[d] | 90.850 | 15,89 % |

*a* Die Zeichenkette, die die komplette Eingabe der Referenz darstellt

*b* Z.B. „[1]“

*c* 200 Zeichen vor und nach dem Zitat im Text

*d* Daten zu *technical reports* (Technische Berichte)

Tabelle 7: GROBID Metadaten zu Publikationen bei 571.897 Datensätzen

| Schlüssel | Anzahl | Anteilig |
|---|---|---|
| title | 321.386 | 56,20 % |
| surname | 286.440 | 50,09 % |
| forename | 284.710 | 49,78 % |
| affiliation | 182.364 | 31,89 % |
| orgName | 177.806 | 31,09 % |
| note | 176.071 | 30,79 % |
| date | 167.098 | 29,22 % |
| biblScope | 161.040 | 28,16 % |
| address | 158.593 | 27,73 % |
| settlement | 134.278 | 23,48 % |
| country | 122.539 | 21,43 % |
| idno[a] | 101.433 | 17,74 % |
| email | 82.232 | 14,38 % |
| postCode | 77.121 | 13,49 % |
| region | 61.422 | 10,74 % |
| publisher | 35.435 | 6,20 % |
| postBox | 6.147 | 1,07 % |

*a* Z.B. ein DOI-Eintrag

Tabelle 8: GROBID Metadaten zu Referenzen bei 571.897 Datensätzen

| Schlüssel | Anzahl | Anteilig |
|---|---|---|
| title | 134.279 | 23,48 % |
| date | 117.815 | 20,60 % |
| biblScope[a] | 116.071 | 20,30 % |
| surname | 106.936 | 18,70 % |
| forename | 106.308 | 18,59 % |
| publisher | 58.307 | 10,20 % |
| pubPlace | 55.199 | 9,65 % |
| editor | 34.049 | 5,95 % |
| note | 30.227 | 5,29 % |
| address | 28.503 | 4,98 % |

*a* Z.B. vom Typ *fpage* (Erste Seite) oder *lpage* (Letzte Seite)

Tabelle 9: ParCit Einzelangaben bei 11.023.889 Referenzen

| Schlüssel | Anzahl | Quotient |
|---|---|---|
| rawString | 11.023.889 | 1,00 |
| marker | 11.023.889 | 1,00 |
| context | 10.695.648 | 0,97 |
| author | 10.523.513 | 0,95 |
| date | 10.378.402 | 0,94 |
| title | 9.361.532 | 0,85 |
| pages | 7.765.207 | 0,70 |
| journal | 6.838.383 | 0,62 |
| volume | 6.776.944 | 0,61 |
| location | 2.319.741 | 0,21 |
| booktitle | 1.519.236 | 0,14 |
| publisher | 1.354.680 | 0,12 |
| issue | 888.605 | 0,08 |
| institution | 359.638 | 0,03 |
| note | 318.288 | 0,03 |
| editor | 311.625 | 0,03 |
| tech | 146.817 | 0,01 |

Tabelle 10: GROBID Einzelangaben bei 2.110.480 Referenzen

| Schlüssel | Anzahl | Quotient |
|---|---|---|
| title | 2.110.480 | 1,00 |
| surname | 1.897.603 | 0,90 |
| date | 1.894.521 | 0,90 |
| forename | 1.703.017 | 0,81 |
| biblScope | 1.683.384 | 0,80 |
| publisher | 209.928 | 0,10 |
| pubPlace | 143.760 | 0,07 |
| address | 90.667 | 0,04 |
| editor | 63.894 | 0,03 |
| note | 45.233 | 0,02 |

das Titel-Element zumindest einmalig vorhanden ist. Über einzelne Referenzen wird hier keine Aussage getroffen. Daher wurde die Datenerhebung so modifiziert, dass die Anzahl der jeweiligen Schlüssel im Verhältnis zur Gesamtanzahl der gefundenen Referenzen stehen. Dies betrifft 11.023.889 Referenzen aus ParsCit-Daten und 2.110.480 Referenzen aus GROBID-Daten. Die Ergebnisse sind in den Tabellen 9 und 10 (auf Seite 28) abgebildet.

### 3.1.3 *Auswahl von Metadaten zur Identifizierung*

Für eine Identifizierung von Publikationen soll in diesem Abschnitt eine Auswahl von geeigneten Metadaten-Elementen getroffen werden. Als Bedingung für die Auswahl eines Elements muss dieses sowohl in einer Tabelle für Metadaten zu Publikationen, als auch in einer Tabelle für Metadaten zu Referenzen vorkommen. Dies ist not-

wendig, da im Laufe der Arbeit Einträge zu Publikationen und zu Referenzen miteinander vereint werden sollen.

Für die Verwendung der extrahierten Metadaten ist als erstes Ergebnis festzuhalten, dass für die Eingaben (571.897) mehr ParsCit Ausgaben (498.904, 87,24 %) erstellt werden konnten, als Ausgaben von GROBID Metadaten zu den Eingabedokumenten (321.386, 56,20 %) und GROBID Metadaten zu den Referenzen (134.279, 23,48 %).

Das erste Element, dass zur Identifikation gewählt wird, ist der Titel. Titel wissenschaftlicher Publikation unterscheiden sich stärker voneinander als z.B. Autoren, die gemeinsam mehrere Publikationen veröffentlichen können und deren Namen potenziell für mehrere unterschiedliche Publikationen verwendet werden. Der Titel ist für 74,95 % der Metadaten zu Publikationen (Tabelle 5) und rund 85 % aller ParsCit-Referenzen (Tabelle 9) verfügbar. Insgesamt kann der verfügbare Anteil von Titeln in extrahierten Referenzen größer als 85 % sein. Diese Möglichkeit besteht, falls mit der Verwendung von GROBID Titel von Referenzen extrahiert werden, für die ParsCit keine Ausgabe liefert.

Als zweites Element sollen die Nachnamen der Autoren verwendet werden. Angaben zu Namen von Autoren sind für 72,25 % der Publikationen (Tabelle 5) und für rund 95 % der ParsCit-Referenzen (Tabelle 9), sowie rund 90 % der GROBID-Referenzen (Tabelle 10) verfügbar. Nachnamen von Autoren sollen als Zusatzinformation zu Titeln bei der Identifikation verwendet werden.

Angaben zum Veröffentlichungsjahr sind für rund 94 % der ParsCit-Referenzen (Tabelle 9) und rund 90 % der GROBID-Referenzen (Tabelle 10) verfügbar. Für Publikationen ist diese Information für 29,22 % der Daten vorhanden (Tabelle 7). Begründet durch das hohe Aufkommen in Metadaten zu Referenzen wird diese Angabe auch zur Identifizierung von Publikationen genutzt.

Als letztes Element wird die DOI gewählt. Diese stellt als Identifikator ein Element zur unmittelbaren Bestimmung von Publikationen dar. Bedingt durch das geringe Vorkommen in Metadaten sollen DOIs zunächst nicht für Referenzmatchings genutzt werden. Es ergibt sich jedoch die Möglichkeit, die Datenqualität der Metadaten zu verbessern. Beispielsweise kann durch Angabe einer DOI über die Anbindung des crossref.org Webservice auf Metadaten zu Publikationen zugegriffen werden [CROSS].

### 3.1.4 *Einschätzung der Datenqualität*

Die Unvollständigkeit der Metadaten lässt vermuten, dass die Qualität der extrahierten Daten nicht ausreichend ist. Für eine Einschätzung, wie viele der in Publikationen verfügbaren Daten genutzt wer-

den können, bietet sich ein Vergleich der extrahierten Daten mit den Eingabedaten an. Ursprünglich war eine Klassifizierung von Datensätzen angedacht, um darauf aufbauend die extrahierten Daten mit der höchsten Qualität zu nutzen. Dabei war eine Klassifizierung nach verwendeten Templates, z.B. LaTeX-Vorlagen verschiedener Verlage, geplant. Eine Klassifizierung der Datenqualität nach verwendeten Templates zur Generierung von PDF-Dateien erwies sich als schwierig. Die Qualität gleicher Templates variiert stark und für einen Teil der vorliegenden Dateien steht nicht unmittelbar fest, welches Template genutzt wird. Daher bietet sich eine Integration von redundant vorliegenden Daten an, anstatt zwischen diesen zu wählen.

Für eine Einschätzung der Datenqualität soll zunächst betrachtet werden, wie groß der Anteil extrahierter Referenzen ist. Dazu wurde aus einer Teilmenge der zur Verfügung stehenden Daten eine zufällige Stichprobe von 50 Dokumenten gewählt. Die Stichprobe stammt aus 973 Publikationen, die als PDF-Dateien über Webseiten von Informatik-Fachgruppen der Universität Paderborn verfügbar sind [PubUPB]. Für diese Publikationen wurde einerseits manuell ermittelt, wie viele Referenzen sie jeweils enthalten. Andererseits wurde festgestellt, wie viele Referenzen in den extrahierten ParsCit- und GROBID-Daten vorhanden sind. Die Ergebnisse zeigt Tabelle 11. Es wurde verglichen, wie viele Referenzen gefunden wurden und die Ergebnisse in Intervallen mit einem Abstand von jeweils 5 % zusammengefasst. Ausnahmen dazu stellen die Zeilen für 0 % und 100 % dar, die kein Intervall darstellen, sondern bei denen eine absolute Gleichheit besteht. (Werte zwischen 95 % und 100 % wurden nicht gefunden.) Zusätzlich ist angegeben, wenn eine Extraktion nicht erfolgreich verlief und damit keine XML-Datei erstellt wurde (*Datei*). Die Fälle, in denen die Dokumente zur Eingabe selbst keine Referenzen enthalten wurden ebenfalls zusätzlich angegeben (*Leer*). Es ist zu erkennen, dass mit ParsCit bei etwas mehr als der Hälfte der Fälle (27 von 50) exakt so viele Referenzen gefunden wurden, wie in den Eingabedokumenten vorhanden sind. Für GROBID sind dies nur 5 von 50. Die Ergebnisse der Extraktion mit ParsCit waren nie leer. In 13 Fällen wurden mehr Referenzen erkannt, als eigentlich existieren. GROBID erkannte bei 14 von 50 Fällen keine der Referenzen. Insgesamt kann davon ausgegangen werden, dass die automatisch extrahierten Metadaten zu Referenzen stark lückenhaft sind und die Qualität damit eher gering ist.

Die Verwendung der extrahierten Daten und Verfahren zum Umgang mit ihrer Datenqualität sind in Abschnitt 4.5 beschrieben.

Tabelle 11: Anzahl extrahierter Referenzen bei 50 Publikationen. Prozentwerte aus Intervall $(x-5, x]$, wenn nicht anders angegeben

| Einträge | ParsCit | GROBID | Einträge | ParsCit | GROBID |
|---|---|---|---|---|---|
| 100 %[a] | 27 | 5 | 60 % | 0 | 1 |
| Leer[b] | 4 | 4 | 70 % | 0 | 2 |
| Datei[c] | 1 | 0 | 75 % | 0 | 3 |
| 0 % | 0 | 14 | 80 % | 0 | 4 |
| 5 % | 0 | 2 | 85 % | 0 | 5 |
| 15 % | 1 | 0 | 90 % | 0 | 2 |
| 25 % | 1 | 0 | 95 % | 0 | 4 |
| 30 % | 0 | 2 | 105 % | 7 | 0 |
| 45 % | 2 | 1 | 110 % | 3 | 0 |
| 50 % | 1 | 1 | 115 % | 3 | 0 |

*a* Exakt 100 % Übereinstimmung der Anzahl der Referenzen
*b* Keine Referenzen in der Publikation verwendet
*c* Keine XML-Datei vorhanden

## 3.2 DISTANZ VON ZITATIONEN

In diesem Abschnitt wird untersucht, wie die vorliegenden Daten genutzt werden können, um Distanzen von Zitationen zu ermitteln. Diese wird dazu verwendet, gemeinsame Vorkommen von Referenzpaaren zu gewichten. Für eine geringe Distanz wird dabei eine starke inhaltliche Verbundenheit vermutet. Ist für eine große Anzahl von Publikationen die Distanz zu anderen bekannt, kann diese für Empfehlungen verwandter Literatur genutzt werden.

Zur Berechnung von Distanzen wird ein Text nach speziellen Zeichenketten durchsucht und die gefundenen Positionen festgehalten. Es handelt sich dabei um Markierungen von Zitaten, beispielsweise die Markierung „[1]“. Zitate werden im Text mit Markierungen versehen, die im Literaturverzeichnis gemeinsam mit Metadaten zu den entsprechenden Dokumenten erneut aufgeführt werden. Dadurch wird eine Zuordnung ermöglicht.

Wird ein Dokument mehrfach zitiert, können zu einer Markierung im Literaturverzeichnis mehrere Markierungen im Text aufgeführt sein. Um Distanzen verschiedener Paare zu ermitteln, sollten daher alle Positionen von Markierungen bekannt sein und genutzt werden. Die Markierungen zu einer Publikation im Text können sich zudem ebenfalls unterscheiden. Beispielsweise kann das erste aufgeführte Doku-

ment in einem Literaturverzeichnis durch eine einfache Markierung „[1]“ und an einer anderen Stelle gemeinsam mit einem weiteren Dokument durch „[1,3]“ referenziert werden.

Die Distanz der Markierungen kann ermittelt werden, indem ein Text in seine Bestandteile (z.B. Abschnitte, Absätze oder Sätze) zerlegt wird und für jeden der Bestandteile untersucht wird, ob Zitations-Paare darin vorhanden sind.

Um die Möglichkeiten der Nutzung von extrahierten Metadaten zur Ermittlung der Distanz von Zitationen zu ermitteln, wurde im Rahmen dieser Arbeit die Struktur der extrahierten XML-Dateien untersucht. In Dateien aus der Extraktion durch GROBID konnten keine Daten zu Positionen von Zitationen gefunden werden.

In ParsCit XML-Dateien konnten zwei Felder ausgemacht werden, die Angaben zu Markierungen enthalten. Die Existenz dieser Felder ist nicht zwingend, in einem Teil der untersuchten Dateien fehlen sie. Zum einen gibt es das Feld *marker*. Werte dieses Feldes beinhalten gefundene Markierungen für eine Zitation. Beispiele hierfür sind „ARUNACHALAM, 1979“ und „3.“ Das zweite Feld ist *context*. Werte dieses Feldes stellen einen Kontext einer Zitation innerhalb des Textes zur Verfügung. Dies geschieht in Form des gefundenen Markierung und jeweils 200 zusätzlichen Zeichen vor und nach der Markierung. Zusätzlich beinhaltet das Feld context die Attribute *citStr*, *position*, *startWordPosition* und *endWordPosition*. Diese beinhalten Angaben zur gefundenen Markierung (z.B. „[9]“) und zur Position der Markierung. Die Angaben zur Position wurden in einem dafür erstellten Java-Programm getestet. Die Marker konnten nicht an den angegebenen Positionen gefunden werden. Dies ist vermutlich auf die Textkodierung oder auf eine andere Behandlung von Zeichenketten in der Programmiersprache Perl, in der ParsCit geschrieben ist, zurückzuführen. In weiteren Untersuchungen konnte festgestellt werden, dass die Angaben im context-Feld nicht vollständig sind. Beim Durchlaufen des Textes und einer Extraktion von bekannten Markierungen konnten im Java-Programm Vorkommen von Markierungen gefunden werden, die nicht im ParsCit Resultat vorhanden sind. Aus diesen Gründen wird in dieser Arbeit auf eine eigene Implementierung zur Suche und Bestimmung von Positionen der Markierungen zurückgegriffen.

## 3.3 DATENHALTUNG

In diesem Abschnitt werden drei Lösungen für die Datenhaltung von Graphen untersucht. Dies geschieht durch einen Benchmark, bei dem zunächst ein Dokumentennetzwerk für jede der drei Datenhaltungs-

lösungen generiert wird. Anschließend werden die zur Berechnung von Bibliometriken benötigten Zeiten miteinander verglichen.

*Benchmark zur Auswahl einer Datenhaltungslösung für Dokumentengraphen*

Im Fokus dieser Arbeit steht die Analyse und Bewertung von Referenzierungen zwischen wissenschaftlichen Publikationen. Damit stellen diese Beziehungen den wichtigsten Teil der Datenbasis für die durchzuführenden Berechnungen dar. Eine Menge solcher Relationen zwischen Publikationen kann auch als Dokumentennetzwerk oder Dokumentengraph bezeichnet werden.

Für den Einsatz der Datenhaltung von Dokumentennetzwerken wird angenommen, dass die Extraktion von Metadaten durchgeführt wurde. Damit stehen Daten verschiedener Art zur Verfügung, die abgelegt und abgerufen werden sollen. Dies sind zum einen eindeutige Bezeichner zur Identifizierung von Publikationen, z.B. Zahlen oder Zeichenketten. Zum anderen stehen Informationen über Referenzierungen zwischen Publikationen zur Verfügung.

Mit Kenntnis der Struktur der benötigten Daten können Anforderungen an eine Komponente zur Datenhaltung von Dokumentengraphen formuliert werden. Zunächst muss eine Möglichkeit geben sein, Identifikatoren und einfache Relationen zwischen diesen speichern zu können. Basierend auf den gespeicherten Daten sollen die so entstehenden Dokumentengraphen durchlaufen werden, um damit Bibliometriken zu berechnen. Grundlegende Operationen hierbei sind einerseits die Abfrage von allen referenzierten Dokumenten für eine vorher festgelegte Publikation. Außerdem muss es möglich sein, für eine Publikation alle weiteren Publikationen zu bestimmen, die diese referenzieren. Eine zusätzliche Anforderung besteht in der performanten Durchführung von nebenläufigen oder sogar parallelen Durchführungen dieser Berechnungen. In einem Produktivsystem sollen also gleichzeitige Anfragen unterstützt werden.

Für die Datenhaltung der Dokumentengraphen werden drei Alternativen verglichen:

1. MySQL, als Vertreter relationaler Datenbanken [MySQL]
2. Neo4j, eine Not only SQL (NoSQL) Graphdatenbank [Neo4j]
3. Titan, eine verteilte Graphdatenbank mit Verwendung von HBase zu Datenhaltung [Titan]

#### 3.3.0.1 *Konzeption und Architektur des Benchmarks*

Der Vergleich der drei Alternativen zur Datenhaltung wurde pragmatisch konzipiert. Die Grundidee des Benchmarks ist die, jeweils ein Netzwerk von Pseudo-Publikationen aufzubauen und die benötigten Zeiten für die umzusetzenden Berechnungen der Bibliometriken zu messen. Es werden an dieser Stelle keine echten Metadaten von Publikationen verwendet, sondern stattdessen Identifikatoren in Form von natürlichen Zahlen generiert. Tatsächliche Netzwerke, die auch in dieser Arbeit behandelt werden, standen während der Durchführung des Benchmarks noch nicht zur Verfügung.

Um eine nach Möglichkeit hohe Vergleichbarkeit zwischen den drei Kandidaten zu erzielen, soll ein möglichst großer Teil der Implementierung für alle Alternativen identisch sein. Dies wird durch eine abstrakte Klasse umgesetzt, in der der eigentliche Ablauf des Benchmarks, sowie der Algorithmus zur Generierung des Netzwerks umgesetzt ist. Diejenigen Teile, die Framework-spezifisch sind, müssen individuell für jede der drei alternativen Datenhaltungen implementiert werden.

Die Struktur der verwendeten Klassen zeigt Abb. 8. In der abstrakten Klasse *Benchmark* sind Methoden zum Benchmark-Ablauf *benchmark()* und dem Festhalten der Ergebnisse *log()* implementiert. Außerdem wird sichergestellt, dass die Struktur der Graphen für alle Kandidaten zur Datenhaltung identisch ist. Dies wurde realisiert, indem beim Generieren eines Graphen durch *createGraph()* alle ausgehenden Kanten mittels *computeTargets()* gleichermaßen berechnet werden. Im Gegensatz zu diesen gemeinsam genutzten Methoden sind die spezifischen Methoden zur Generierung von Knoten und Kanten eines Graphs für jede Alternative zur Datenhaltung individuell implementiert (*createEdges()* und *createNodes()*). Ebenso verhält es sich mit den Methoden *bibliographicCoupling()* und *coCitation()* zur Berechnung der Bibliometriken. Bei der durchgeführten Implementierung wurden zudem Transaktionen verwendet, um den Aufbau der großen Netzwerke zu beschleunigen. Die Transaktionen wurden ebenfalls für jeden Kandidaten zur Datenhaltung implementiert und eine experimentell ermittelte Größe von 5.000 Operationen je Transaktion verwendet.

Der Aufbau der Publikationsgraphen wurde als Kompromiss zwischen realitätsnahen Publikationsnetzwerken und geeigneten Eigenschaften für einen Benchmark gestaltet. Während auf die Beschreibung der Implementierung weiter Teile des Benchmarks hier verzichtet werden soll, wird der Aufbau der Publikationsnetzwerke am Beispiel von Quellcode aus Listing 6 im Folgenden erläutert.

In dieser Arbeit wird bei der Betrachtung von Publikationsdaten zwischen zwei Typen von Dokumenten unterschieden: Solche, die kom-

Listing 1: Generierung von Identifikatoren referenzierter Knoten

```
int FULLTEXT_RATIO = 10
int EDGES_MIN = 20
int EDGES_MAX = 30

Set<Integer> computeTargets(int id) {
 Set<Integer> targets = new HashSet<Integer>();

 if (id % FULLTEXT_RATIO != 0)
  return targets;

 if (id <= EDGES_MAX) {
  for (int refId = id - 1; refId > 0; refId--)
   targets.add(refId);
  return targets;
 }

 int numberOfEdges = EDGES_MAX -
  (id % (1 + EDGES_MAX - EDGES_MIN));

 for (int j = 1; j < FULLTEXT_RATIO; j++)
  targets.add(id - j);

 int power = 1;
 int difference = 10;
 while (id > difference + (id % 13)) {
  targets.add(id - difference - (id % 13));
  difference = (int) (Math.pow(10, ++power));
 }

 power = 3;
 difference = 8;
 while (id - difference > FULLTEXT_RATIO) {
  int nextFulltext = (id - difference) -
   ((id - difference) % FULLTEXT_RATIO);
  targets.add(nextFulltext);
  difference = (int) (Math.pow(2, ++power));
 }

 int j = 1;
 while (targets.size() < numberOfEdges) {
  targets.add(id - j++);
 }

 return targets;
}
```

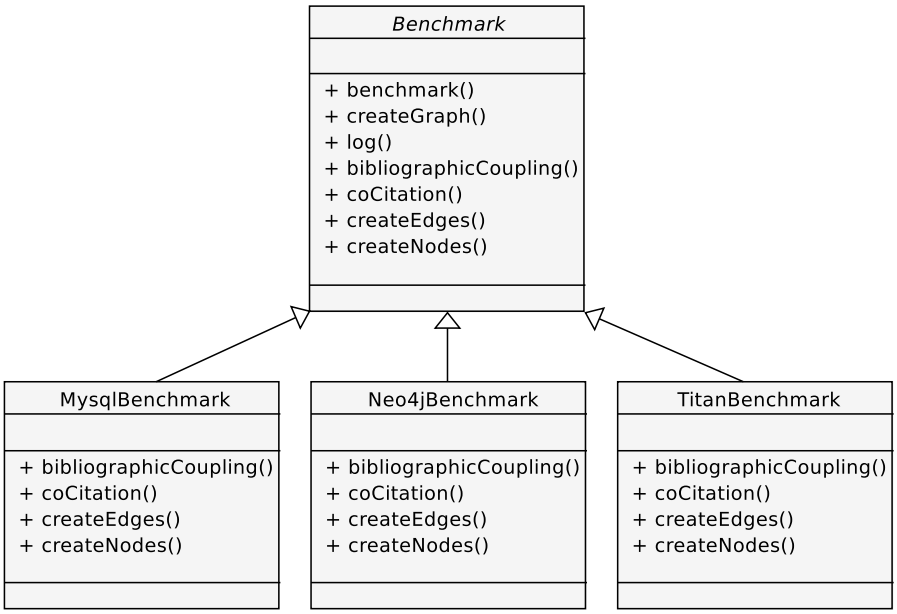



Abbildung 8: Klassendiagramm Benchmark

plett als Volltext vorliegen, und solche, über die nur Metadaten aus Referenzierungen bekannt sind. Das Verhältnis dieser Typen ist im Benchmark 1 : 9 und wird in der Konstante *FULLTEXT_RATIO* (Listing 6, Zeile 1) festgelegt. Publikationen, die als Volltext vorliegen, beinhalten zwischen *EDGES_MIN* und *EDGES_MAX* viele Referenzen (Z. 2-3). Die recht hohen Werte von 20 bis 30 ausgehenden Kanten werden benötigt, damit der Benchmark-Graph einen hohen Durchschnittsgrad aufweist. Damit entsteht ein Aufwand für die abschließende Berechnung der Bibliometriken.

Die Funktion *computeTargets()* (Zeile 5) wird für jeden generierten Knoten aufgerufen. Dabei wird der aktuelle Knoten-Identifikator *id* übergeben und eine Menge von Identifikatoren zurückgegeben. Dies sind Bezeichner für diejenigen Knoten, die im Netzwerk vom Knoten *id* referenziert werden. Die Knoten dieser Menge sind aus dem Intervall $[1, id)$. Dies legt eine zeitliche Ordnung fest, Publikationen des Benchmarks können nur Publikationen mit niedriger ID, also zeitlich vorhergehende Publikationen, referenzieren.

Die folgenden Fallunterscheidungen sind Sonderfälle. Publikationen, für die keine Volltexte vorliegen, haben keine Referenzen, daher ist die zurückgegebene Menge ihrer Referenzen leer (Z. 8-9). Publikationen mit sehr kleiner Identifikationsbezeichnung (ID) aus dem Intervall $[1, EDGES\_MAX]$ sind im hier verwendeten Graph Volltexte. Da sie nur sehr wenige zeitliche Vorgänger haben, referenzieren sie alle Publikationen mit niedriger ID (Z. 11-15).

Die Anzahl von Referenzen einer Publikation stammt aus dem Intervall $[EDGES\_MIN, EDGES\_MAX]$ und hängt von der eigenen ID ab

(Z. 17-18). Dadurch entsteht eine Gleichverteilung der Anzahl von Referenzen. Jeder Volltext referenziert diejenigen Publikationen, deren IDs zwischen der des aktuellen und des zeitlich vorhergehenden Volltexts befinden. Dies stellt sicher, dass sich kein Referenzeintrag im Datenbestand befindet, der niemals referenziert wird (Z.20-21).

Damit Publikationen zwar eher zeitlich nähere, aber insgesamt auch alte Publikationen referenzieren, wächst ein Teil der Abstände referenzierter Publikationen exponentiell. Dies wird dadurch verwirklicht, dass von der aktuellen ID Potenzen der Zahl 10 abgezogen werden. Zusätzlich wird durch das Einbeziehen der aktuellen ID die Verteilung etwas gestreut. Von der ID 13000 würden z.B. die IDs 12990, 12900, 12000 und 3000 referenziert (Z. 23-28).

Ein ähnliches Verfahren wird nochmals angewendet, allerdings mit zwei Unterschieden. Erstens wird mit Potenzen der Zahl 2 gearbeitet, um die Abstände kleiner zu halten. Zweitens werden in diesem Fall nur Referenzen zu Volltexten generiert, um eine dichtere Vernetzung zwischen diesen zu erreichen (Z. 30-37). Abschließend werden der zurückzugebenden Menge weitere zeitlich nahe IDs hinzugefügt, um die Gleichverteilung der Anzahl von Referenzen aufrechtzuerhalten (Z. 39-44).

#### 3.3.0.2 *Durchführung und Ergebnisse*

Die Durchführung des Benchmarks besteht aus zwei Teilen. Im ersten Teil wird ein Graph mit aus den zugehörigen Knoten und Kanten generiert. Während dieses Prozesses werden in regelmäßigen Abständen Bibliometriken berechnet und die Berechnung benötigten Zeiten festgehalten. Bei der Durchführung des zweiten Benchmarkteils steht ein vollständig aufgebauter Graph zur Verfügung. Auf diesem wird eine Menge von parallelen Berechnungen gestartet und die benötigte Zeit hierfür festgehalten. Die beiden Teile des Benchmarks werden für die drei möglichen Lösungen zur Datenhaltung durchgeführt und anschließend die benötigten Berechnungszeiten miteinander verglichen.

ERSTER TEIL DES BENCHMARKS Zum Zeitpunkt der Durchführung des Benchmarks standen im verwendeten Dokumentenkorpus etwa eine Million wissenschaftliche Publikationen zur Verfügung. Da die Anzahl der vorhandenen Dokumente zu späteren Zeitpunkten potenziell größer sein kann, wurde die Anzahl von Volltexten für den Benchmark auf zwei Millionen Knoten festgesetzt. Mit dem verwendeten Verhältnis von 1 : 9 zwischen Volltexten und referenzierten Dokumenten ergibt sich insgesamt eine Anzahl von 20 Millionen Knoten.

Die Generierung der Knoten und Kanten und die Berechnung der Bibliometriken im Benchmark erfolgen wechselweise. Zunächst wer-

Tabelle 12: Laufzeiten der Graphgenerierung (in h:min:s)

| MySQL | Neo4j | Titan |
|---|---|---|
| 7:11:21 | 6:55:44 | 41:14:31 |

den 10.000 Knoten generiert. Anschließend werden für den Knoten mit der ID 10000 die Ergebnisse für die bibliografische Kopplung und die Kozitation berechnet. Die dafür benötigten Rechenzeiten werden für eine spätere Auswertung festgehalten. Als Nächstes werden die Knoten und Kanten bis zur ID 20000 generiert. Für diese ID werden erneut die Bibliometriken berechnet, in diesem Fall für die IDs 10000 und 20000. Mit der Berechnung für die ID 10000 stehen nun also zwei gemessene Berechnungszeiten zur Verfügung, die miteinander verglichen werden können. Die Generierung und Berechnung wird in der beschriebenen Weise für das Intervall $(0, 100.000]$ fortgeführt. Für das Intervall $(100.000, 20.000.000]$ geschieht dies ebenso, die Schrittlänge wird hier allerdings von 10.000 Knoten auf 100.000 Knoten heraufgesetzt. Mit der wachsenden Anzahl der IDs steigt auch die Anzahl der durchgeführten Berechnungen für die einzelnen Schritte. Dadurch entsteht bei einer Größe von insgesamt 20 Millionen Knoten eine Menge von 43.890 festgehaltenen Berechnungszeiten. Diese ergeben sich jeweils für die einzelnen Bibliometriken und für jede der Datenhaltungslösungen.

Der Benchmark wurde für die Datenhaltungslösungen MySQL, Neo4j und die Titan/HBase Kombination ausgeführt. Die dafür benötigten Laufzeiten sind in Tabelle 12 aufgeführt. Da die Generierung der Graphen mit den gleichzeitig durchgeführten Berechnungen künstlich und damit nicht praxisrelevant sind, fließen die Gesamtlaufzeiten nicht in die Entscheidung für die zu verwendende Datenhaltung ein.

Im Interesse der Durchführung dieses Benchmarks steht, wie groß die einzelnen Berechnungszeiten sind. Die Berechnungen im Produktivsystem sollen in Echtzeit ausgeführt werden. Das bedeutet, dass ein Benutzer interessiert an einem bestimmten Dokument ist und für dieses Berechnungen von Bibliometriken angefordert werden. Diese Berechnungen sollen performant ausgeführt werden. Die benötigte Laufzeit für alle auszuführenden Berechnungen muss im Akzeptanzbereich der Nutzer liegen.

Für die Auswertung der Laufzeiten wurde die Anzahl der Knoten im Netzwerk als einzelne zu betrachtende Messpunkte gewählt. Für diese Messpunkte wurde das arithmetische Mittel aller gemessenen Laufzeiten berechnet. Die Ergebnisse der insgesamt 263.340 ursprünglichen Einzelwerte wurden anschließend grafisch aufbereitet und sind in Abb. 9 dargestellt. Es sind zwei Ergebnisse festzuhalten:

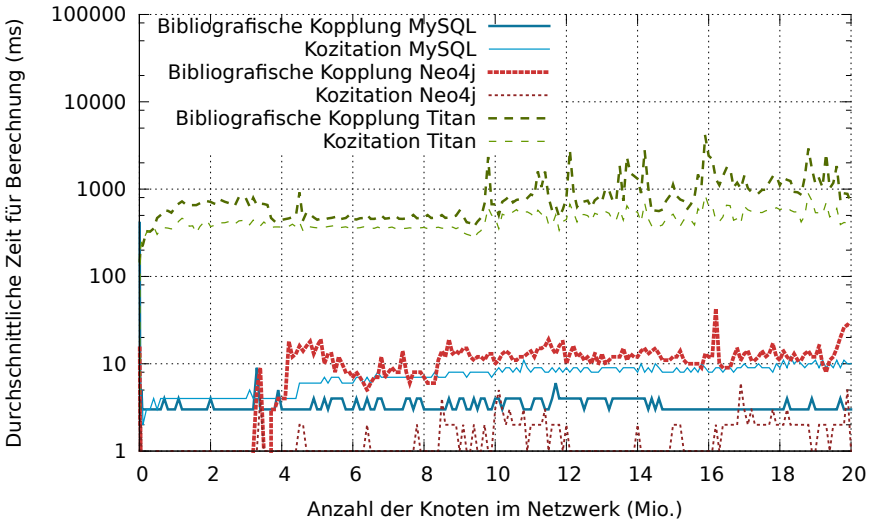


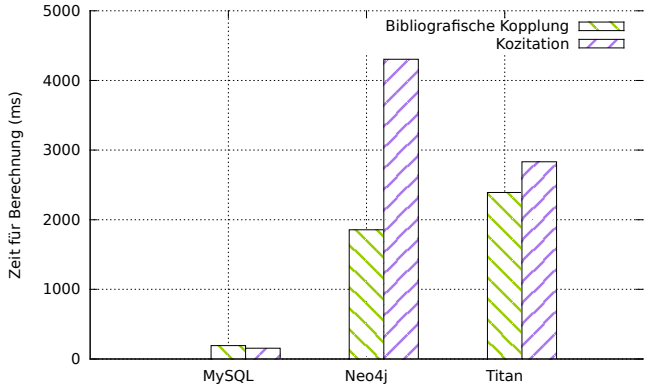

Abbildung 9: Benchmark Datenhaltung: Berechnung von Bibliometriken

1. Die Anzahl der Knoten im Netzwerk hat keine maßgebliche Auswirkung auf die Laufzeit der Berechnungen. Dies ist daran zu erkennen, dass für den betrachteten Bereich die Laufzeiten mit wachsender Netzwerkgröße nicht oder nur sehr leicht ansteigen. Vermutlich ist die Anzahl der genutzten Datensätze zu gering, um das MySQL System mittels der Berechnungen auszulasten. Es ist für diesen Punkt keine Datenhaltungslösung auszuschließen.
2. Während die maximalen Laufzeiten von MySQL[3] (12 ms) und Neo4j (43 ms) eine Schwelle von 43 ms nicht überschreiten, liegt die maximale Ausführungszeit von Titan bei 4.065 ms und der Durchschnitt bei 435 ms (Kozitation) bzw. 813 ms (Bibliografische Kopplung). Damit ist Titan im Schnitt um den Faktor 10 langsamer und wird deshalb als Datenhaltungslösung für die Dokumentengraphen ausgeschlossen.

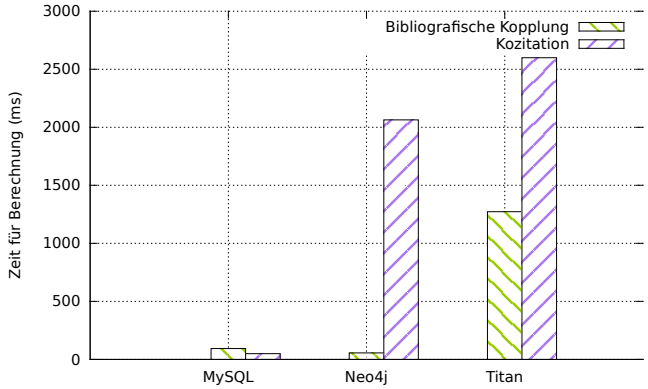
ZWEITER TEIL DES BENCHMARKS Im vorangegangen Teil des Benchmarks wurde ein Dokumentennetzwerk erstellt und zeitgleich Bibliometriken berechnet. Damit konnte das Verhalten der Datenhaltungslösungen bei wachsender Größe der Dokumentennetzwerke untersucht werden.

Der zweite Teil des Benchmarks nutzt die bereits erstellen Dokumentennetzwerke. Dabei werden benötigte Laufzeiten für parallele Anfragen gemessen. Zur Ausführung wird ein Java ThreadPool genutzt, und ausgehend von verschiedenen Threads Anfragen zu Bibliometriken gesendet. Der Benchmark besteht aus vier Teilen. Innerhalb eines Teils werden für jede Lösung zur Datenhaltung Anfragen zur

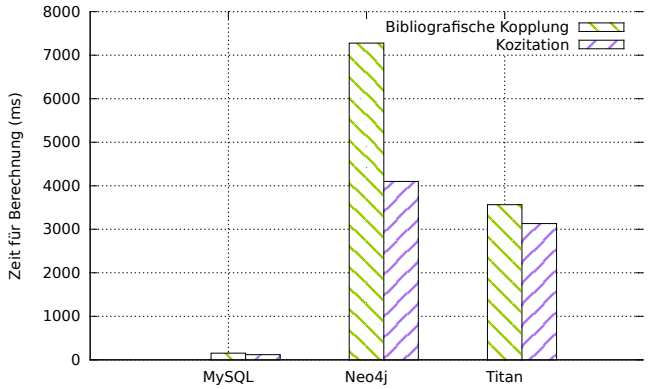
3 Ein einmaliger, initialer Wert von 410 ms ist hier außer Acht gelassen



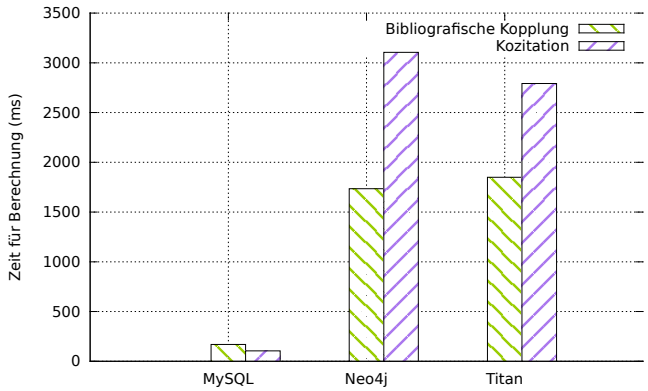

(a) 10 Anfragen aus 10 Mio. Knoten




(b) 10 Anfragen aus 20 Mio. Knoten




(c) 20 Anfragen aus 10 Mio. Knoten




(d) 20 Anfragen aus 20 Mio. Knoten

Abbildung 10: Performance Test durch parallele Anfragen

Kozitation und Bibliografischen Kopplung gesendet. Die vier Teile entstehen durch folgende Unterteilung: In zwei Fällen werden als Ausgangspunkt Knoten-IDs aus den ersten 10 Mio. Knoten gewählt. In den zwei übrigen Fällen werden als Eingabe Koten-IDs aus 20 Mio. Knoten gewählt. Um vergleichbare Laufzeiten zu erhalten, werden für jede Lösung zur Datenhaltung identische Knoten-IDs genutzt. Die einzelnen Fall-Paare werden nochmals unterteilt. In einem Teilfall werden 10 parallele Anfragen gestellt, im anderen sind es 20 parallele Anfragen.

Die Ergebnisse des zweiten Benchmarks zeigt Abb. 10. Für den Vergleich der Fälle (a) bis (d) sei hiermit auf die unterschiedliche Skalierung der Achse zu den Laufzeiten zugunsten genauerer Einzeldarstellungen hingewiesen. Die maximale MySQL-Laufzeit aus allen Testfällen ist 193 ms. Die Neo4j-Laufzeiten für die Berechnung der Bibliografischen Kopplung sind im Bereich zwischen 56 ms und 7.278 ms. Die Laufzeit von 56 ms ist durch ein Caching einer Anfrage, die während der Tests mehrfach gesendet wurde, zu erklären. Die maximale Neo4j-Laufzeit für die Berechnung der Kozitation liegt bei 4.304 ms. Die maximale Titan-Laufzeit liegt bei 3.565 ms für die Berechnung der Bibliografischen Kopplung und bei 3.131 ms für die Berechnung der Kozitation. Insgesamt liefert MySQL die kürzesten Laufzeiten.

# 4 KONZEPTION

Dieses Kapitel enthält den Entwurf für das System zur Berechnung von Literaturempfehlungen. Die Beschreibung der Konzepte orientiert sich dabei an der durchzuführenden Implementierung.

In Abschnitt 4.1 wird zunächst ein 5-Phasen-Modell vorgestellt. Dies repräsentiert eine Einteilung der durchzuführenden Berechnungen in aufeinander folgende Phasen. Anschließend wird der Entwurf zur verteilten Berechnung in Abschnitt 4.2 vorgestellt. Die folgenden Abschnitte bilden den größten Teil der Konzeption. Sie enthalten detaillierte Beschreibungen der Entwürfe der jeweiligen Phasen.

In den einzelnen Abschnitten dieses Kapitels werden Komponenten beschrieben, die später in Java-Code umgesetzt werden. Diese Komponenten werden als Maven [Maven] Softwareprojekte umgesetzt. Die Schnittstellen, die in den Komponetendiagrammen der Abschnitte abgebildet sind, stellen die Abhängigkeiten der Maven Projekte dar.

## 4.1 MODELLIERUNG DER PHASEN

Aufbauend auf den Ergebnissen der Analyse wurde der grundlegende Gesamtablauf der Datenaufbereitung modelliert. Das Ergebnis ist ein 5-Phasen-Modell. Das Modell setzt sich aus den Phasen Konvertierung, Extraktion, Integration, Deduplikation und Präsentation zusammen. Die fünf Phasen, der Datenfluss zwischen ihnen, die Datenhaltung und die verwendeten Datenformate sind in Abb. 11 dargestellt. Eine zentrale Rolle im Modell nimmt die Datenhaltung ein. Diese soll mittels der verteilten Datenbank HBase umgesetzt werden und unterstützt die Möglichkeit der parallelen Verarbeitung innerhalb der Phasen. Im Folgenden sind die Ziele und Berechnungen der Phasen, sowie der Datenfluss beschrieben.

### 4.1.1 *Konvertierung*

Die für diese Arbeit verwendeten wissenschaftlichen Publikationen liegen im Ausgangsformat PDF vor. Für die Extraktion von Metadaten in der zweiten Phase werden neben den PDF-Eingabedaten zusätzlich Textdateien benötigt. Es ist also notwendig, die PDF-Dateien zu konvertieren. Eine Komponente für diesen Arbeitsschritt stand mit Beginn dieser Arbeit bereits im HCPA zur Verfügung. Als Teil der Mas-

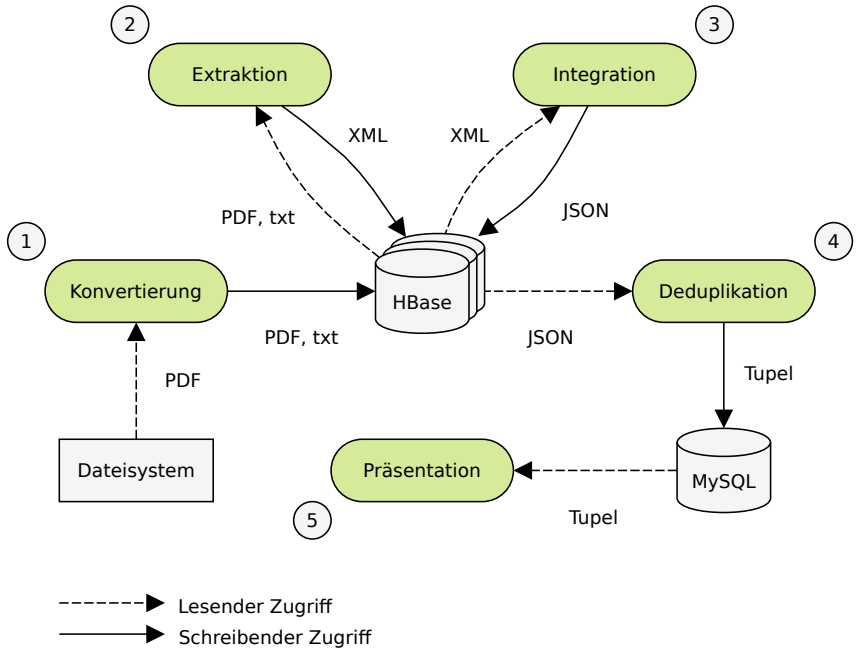



Abbildung 11: Übersicht des 5-Phasen-Modells: Phasen, Datenfluss, Datenhaltung und verwendete Datenformate

terarbeit von Tobias Varlemann [Var13] wurde die Komponente *PDF-Converter* entwickelt. Diese liest PDF-Dateien aus dem Dateisystem ein und extrahiert entsprechende Volltexte.

### 4.1.2 *Extraktion*

In der Extraktionsphase wird für jede Publikation eine Extraktion von Metadaten angestoßen. Dies umfasst Metadaten zur Publikation selbst, sowie Metadaten zu referenzierten Dokumenten. Der Vorgang wird dabei durch zwei Einzelkomponenten umgesetzt. Dies erzeugt für einen Teil der Publikationen Redundanz, da Metadaten mehrfach extrahiert werden. Diese Redundanz wird in der folgenden Integrations-Phase zur Unterstützung der Datenqualität genutzt. Die Datenhaltung der Extraktion stützt sich für Eingaben sowie Ausgaben auf HBase. Als Eingabe werden – je nach verwendeter Extraktions-Komponente – extrahierte Volltexte oder PDF-Dateien genutzt. Das Ausgabeformat der Extraktionsvorgänge ist XML, da dieses von den eingebundenen Softwarekomponenten verwendet wird.

### 4.1.3 *Integration*

Die redundanten Ergebnisse der Extraktionsphase werden in der Integrationsphase miteinander kombiniert und in ein einheitliches Format überführt. Als Eingabe werden hierfür die extrahierten XML-Datensätze genutzt. Nach Beendigung dieser Phase steht für jede Datei exakt ein Datensatz zur weiteren Bearbeitung zur Verfügung. Die Integration der Datensätze setzt sich aus regelbasierten Heuristiken zusammen. Die Integration wird in zwei separaten Teilen durchgeführt:

1. Integration der Metadaten der Publikation
2. Integration der Metadaten des Referenzteils

Bei der Integration der Metadaten der Publikation werden die Metadaten-Felder miteinander verglichen und ggf. ergänzt. Ein Beispiel hierfür ist der Vergleich der gefundenen Autoren. Ist in zwei XML-Datensätzen jeweils ein Autor aufgeführt, deren Namen sich jedoch unterscheiden, so resultiert die Integration in einer Liste von zwei Autoren. Die Integration der Metadaten des Referenzteils ist komplexer. Sie besteht aus einem paarweisen Vergleich der gefundenen Referenzen. Dabei werden zunächst zwei Listen gebildet und identische Datensätze identifiziert. Innerhalb der übrigen Listeneinträge wird nach Überschneidungen innerhalb der Datensätze gesucht und die Einträge darauf basierend miteinander kombiniert. Als Ausgabeformat der Integration wurde das Dateiformat JSON gewählt. Dadurch wird das Datenvolumen im Vergleich zur Verwendung von XML verkleinert.

### 4.1.4 *Deduplikation*

In der Deduplikationsphase werden die vorbereiteten JSON-Datensätze in eine Netzwerkstruktur überführt. Diese Netzwerkstruktur hat die Form eines Graphen, in dem Publikation und Einträge aus den Referenzteilen Knoten darstellen. Die Kanten des Graphen ergeben sich aus Publikationen als Quellen und den Einträgen der Referenzteile als Senken. Innerhalb Deduplikation sollen Duplikate erkannt und vereinigt werden. Duplikate sind verschiedene Datensätze, die dieselbe Publikation repräsentieren. Die berechneten Netzwerkdaten werden als Tupel in einer MySQL Datenbank gespeichert.

### 4.1.5 *Präsentation*

In der Präsentationsphase werden Bibliometriken berechnet und die Ergebnisse dargestellt. Die Berechnungen basieren auf den Datensät-

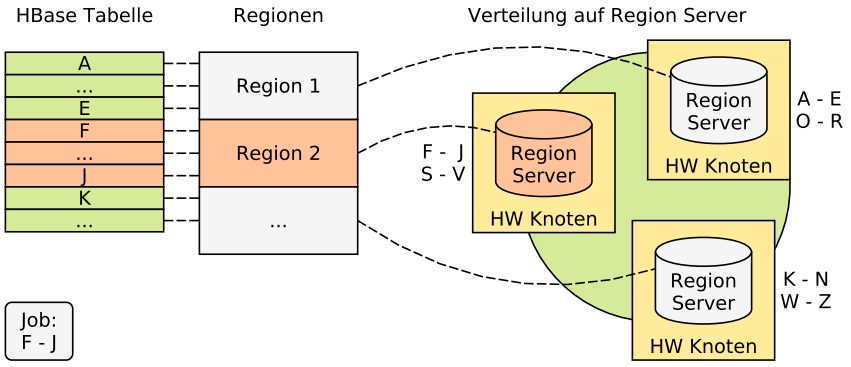



Abbildung 12: Ausführung vieler Berechnungen auf einem Knoten

zen der MySQL Datenbank. Zusätzlich werden Funktionen zur Navigation im Dokumentennetzwerk bereitgestellt.

## 4.2 ENTWURF ZUR VERTEILTEN BERECHNUNG

Die verteilten Berechnungen in dieser Arbeit werden mit Hadoop vorgenommen. Dabei wird die verteilte Datenhaltung HBase genutzt. Ausnahmen davon sind die lokale Speicherung für Zwischenergebnisse und die Verwendung einer rationalen Datenbank zur Berechnung von Bibliometriken.

In HBase werden Zeilen über die Angabe von Rowkeys (Schlüsselwerten) adressiert. Innerhalb des HCPA werden Pfadangaben zur Adressierung verwendet. Ein Beispiel für einen verwendeten Rowkey ist „/upload/upb-cs/ag-bloemer/crypto03.pdf“. Diese Darstellung wird für eine wechselseitige Zuordnung von HBase Zeilen und Dateien in einem hierarchischen Dateisystem genutzt.

Zeilen, die über Rowkeys adressiert sind, werden in Regionen zusammengefasst und diese auf RegionServer im Cluster verteilt. (Siehe dazu das Beispiel in Abschnitt 2.4.3 auf Seite 21.) Von der Replizierung abgesehen, ist ein Datensatz genau einem RegionServer zugeordnet. Die Kenntnis dieser physikalischen Abhängigkeit wird innerhalb von Hadoop Berechnungen berücksichtigt. Die Datenlokalität wird genutzt, indem ein Eingabe-Datensatz einer Mapper-Instanz zugeordnet wird, die auf der gleichen Hardware ausgeführt wird. Damit wird unnötiger Datenverkehr vermieden (vgl. [Sam12, S. 33]). Dies kann in speziellen Fällen von Nachteil sein.

Abb. 12 zeigt ein Beispiel, bei dem Daten des Rowkey-Intervalls F-J für Berechnungen verwendet werden sollen. Diese Berechnungen werden im Beispiel durch Mapper-Komponenten ausgeführt. Da die gegebene Datenlokalität vom Framework genutzt wird, werden alle

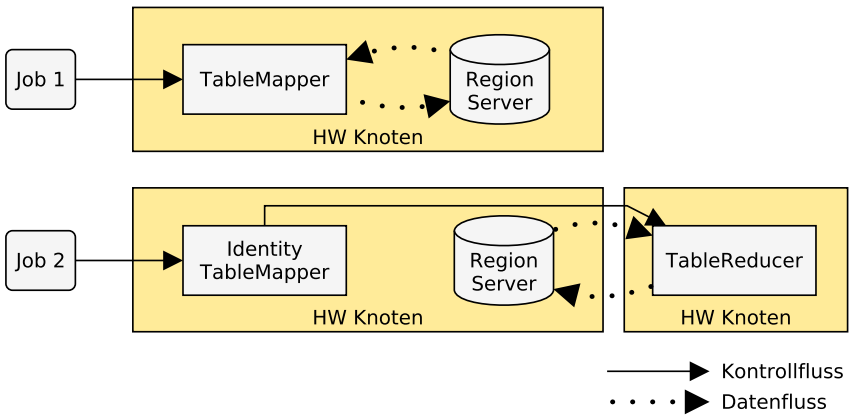



Abbildung 13: Daten- und Kontrollfluß bei der Nutzung von Mapper- und Reducer-Komponenten

Berechnungen auf dem für das Intervall zuständigen RegionServer ausgeführt. Die Rechenkapazitäten der übrigen Knoten werden nicht genutzt.

Für Fälle, bei denen Berechnungen mit kleinen Intervallen von Rowkeys durchgeführt werden sollen, kann eine Verteilung der Berechnungen von Vorteil sein. In diesen Fällen wird ein höherer Datenverkehr zugunsten der Gesamtlaufzeit in Kauf genommen. Dazu werden Eingaben für Mapper-Komponenten an Reducer-Komponenten weitergeleitet.

Abb. 13 zeigt eine Gegenüberstellung der beiden möglichen Ausführungstypen. Für *Job 1* werden alle Berechnungen auf der gleichen Hardware von *TableMapper*-Instanzen ausgeführt. Für *Job 2* leiten *IdentityTableMapper*-Instanzen die Eingabeparameter an *TableReducer*-Instanzen weiter. Diese werden auf weiteren Netzwerk-Knoten ausge-

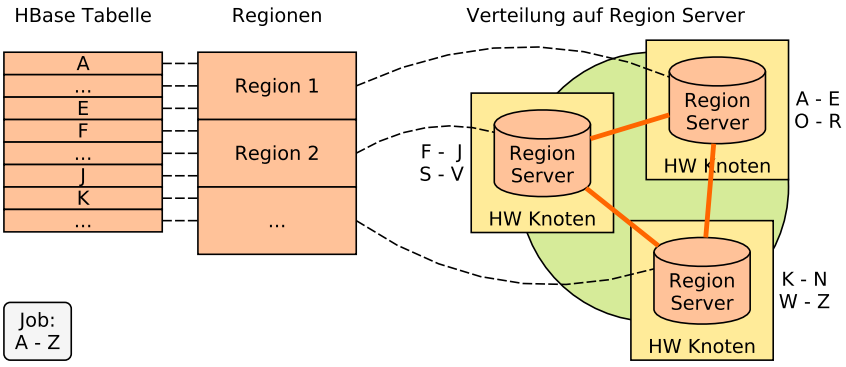



Abbildung 14: Verteilte Ausführung mit hohem Aufkommen von Datenverker

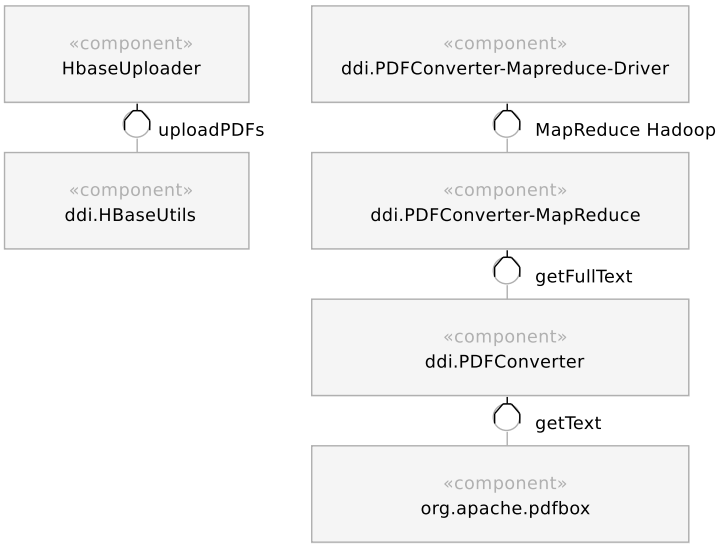



Abbildung 15: Komponenten der Phase Konvertierung

führt. Am Datenfluss ist zu erkennen, dass in diesem Fall ein höherer Datenverkehr entsteht.

Auch dieser Verarbeitungstyp sollte nicht für alle ausschließlich verwendet werden. In Fällen, in denen die zur Eingabe verwendeten Rowkeys über viele Knoten verteilt sind, ist eine Ausnutzung der Datenlokalität und eine Vermeidung von Datenverkehr von Vorteil. Abb. 14 zeigt ein Beispiel, in dem alle verfügbaren Rowkeys als Eingabe genutzt werden. Dabei werden Reducer-Komponenten zur Berechnung verwendet und die Eingabe- und Ausgabedaten über das Netzwerk gesendet.

In der zu entwickelnden Implementierung sollen beide Lösungen umgesetzt werden. Damit kann jeweils gewählt werden, welcher Verarbeitungstyp sich für eine Berechnung besser eignet.

## 4.3 DATENERFASSUNG UND -KONVERTIERUNG

Die erste Phase zur Vorbereitung der Berechnung von Bibliometriken ist die Konvertierung von Eingabedaten. Diese beinhaltet die Erfassung von PDF-Dateien als Eingabedaten und eine anschließende Überführung der PDF-Dateien in ein Reintextformat. Die für diese Phase verwendeten Komponenten standen zu Beginn dieser Arbeit größtenteils zur Verfügung und konnten genutzt bzw. eingebunden werden. Abb. 15 zeigt die zur Konvertierung genutzten Komponenten. Dabei sind die in [Var13] entwickelten und bereits verfügbaren Komponenten mit dem Präfix „ddi“ gekennzeichnet.

### 4.3.1 *Erfassung von PDF-Dateien*

Die Erfassung von PDF-Dateien wird über die Komponente *Hbase-Uploader* vorgenommen. Diese stellt eine Schnittstelle zur eigentlichen Funktionalität in der Komponente *ddi.HBaseUtils* dar. Aufgabe der Komponente ist die Erfassung von PDF-Dateien in einem hierarchischen Dateisystem und die anschließende Ablage in einer Tabelle der verteilten Datenbank HBase. Die Schnittstelle kann durch die Eingabe von vier Argumenten genutzt werden:

- *inputfolder* Der Pfad, über den die hinzuzufügenden Dateien erreichbar sind
- *outputtableinstance* Die zu nutzende HBase Tabelle
- *prefix* Das Präfix, das den Dateien zur Identifizierung vorangestellt wird
- *server* Die Adresse des zu nutzenden Servers

### 4.3.2 *Konvertierung der Datenformate*

Die Konvertierung der hochgeladenen Dateien geschieht über die Komponente *ddi.PDFConverter*. Dabei wird die Java-Bibliothek Apache PDFBox [PDFBox] verwendet. Da die Eingabedateien in diesem Schritt nicht mehr in einem hierarchischen Dateisystem liegen, so wie sie bei der Erfassung der PDF-Dateien vorlagen, sondern verteilt in der Datenbank HBase, besteht die Möglichkeit die Dateien parallel zu verarbeiten. Hierfür werden Hadoop und Instanzen der Komponente *ddi.PDFConverter-MapReduce* verwendet. Die parallele Verarbeitung wird über die Komponente *ddi.PDFConverter-Mapreduce-Driver* angestoßen.

Nach dem Erfassen und der Konvertierung der Eingabedaten stehen die Binärdaten der eingelesenen PDF-Dateien sowie die Volltexte in der verteilten Datenbank zur Verfügung. Diese können zur Extraktion von Metadaten genutzt werden.

## 4.4 EXTRAKTION VON METADATEN

Die Extraktion von Metadaten umfasst die verteilte Ausführung von Instanzen der Softwarepakete GROBID [GRO-Git] und ParsCit [ParsCit]. Diese wurden aufgrund der Ergebnisse, die in der zugehörigen Literatur genannt werden, ausgewählt (vgl. Abschnitt 2.2.1). Hierfür werden Eingabedaten in den Formaten PDF und Reintext aus HBase gelesen und extrahierte XML-Daten in HBase gespeichert. Nach dem Durchlaufen der Extraktionsphase stehen damit Metada-

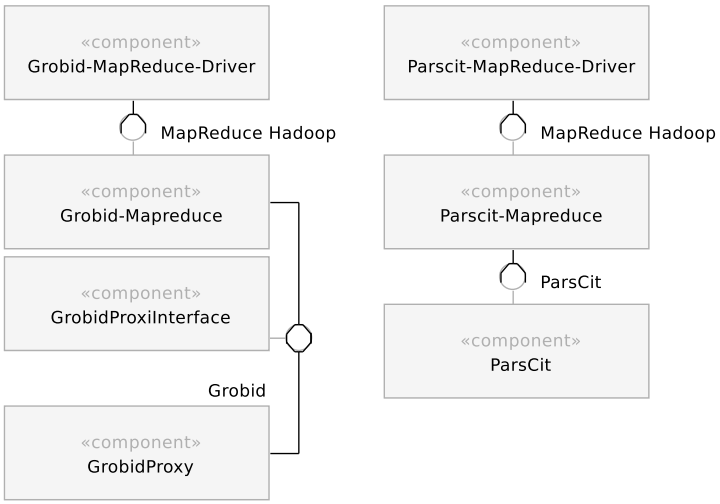



Abbildung 16: Komponenten der Phase Extraktion

ten zu den Publikationen und ihren Referenzen zur weiteren Verarbeitung zur Verfügung. Eine Einordnung in den Gesamtablauf des Systems zeigt Abb. 11 auf Seite 44.

Für die Extraktion wurden zusätzlich zu den beiden Softwarepaketen sieben weitere Komponenten entworfen. Abb. 16 zeigt die vier Grobid- und die drei Parscit-Komponenten.

Im Fokus des Entwurfs standen die Wiederverwendbarkeit der einzelnen Komponenten und die verteilte Ausführung im Rechnercluster. Die Extraktionstools werden lokal auf den Rechnern im Cluster installiert und können sowohl im Kontext einer Hadoop-Ausführung als auch von weiteren externen Anwendungen genutzt werden. Dabei stellen die Komponenten *Grobid-MapReduce-Driver* und *Parscit-MapReduce-Driver* Schnittstellen zur Konfiguration der verteilten Ausführung zur Verfügung. So kann festgelegt werden, welches Intervall von HBase-Schlüsseln genutzt werden soll und ob die Extraktion lokal vorgehaltener Daten oder eine Verteilung der Datensätze im Vordergrund steht. Über die Komponenten *Grobid-Mapreduce* und *Parscit-MapReduce* geschieht die Anbindung der Komponenten zur Extraktion an Mapper- bzw. Reducer-Klassen des Hadoop-Frameworks.

Die eigentliche Extraktion wird zum einen über die Komponente *ParsCit* vorgenommen. Diese stellt die benötigte Umgebung zur lokalen Ausführung von der Extraktionssoftware ParsCit zur Verfügung. Dies umschließt das speichern von XML Daten in lokalen Dateien, die zur Eingabe der Extraktion genutzt werden, die Ausführung in einem eigenen Prozess und den Lesevorgang der Ausgabe des Prozesses. Zur Verwendung in der Deduplikations-Phase werden zusätzlich Distanzen von Zitationen basierend auf Positionsangaben berechnet, die im ParsCit XML aufgeführt sind. Dies wird an dieser Stelle durchgeführt, da sowohl in der Extraktion als auch bei der Berechnung von Distan-

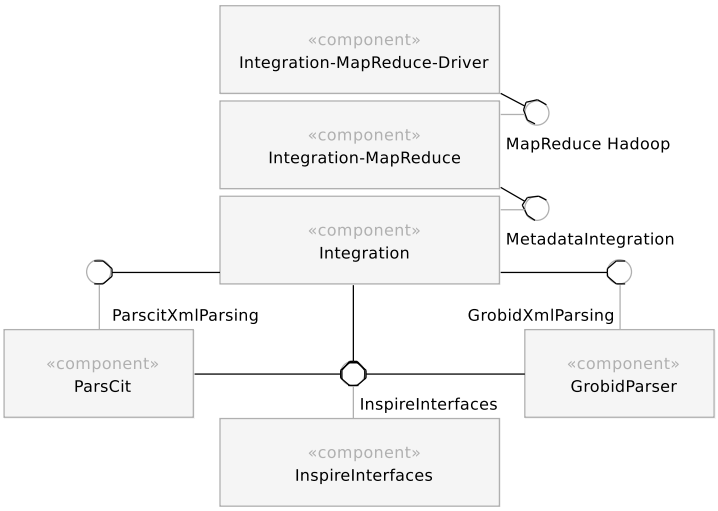



Abbildung 17: Komponenten der Phase Integration

zen auf Volltexte zugegriffen wird. Die Berechnung der Distanzen wird innerhalb der Deduplikation-Phase in Abschnitt 4.6 beschrieben. Zum anderen wird eine Extraktion mit GROBID vorgenommen. Auch diese ist lokal auf den Rechnern im Cluster installiert. Der Zugriff auf die benötigten Java-Klassen wird durch das Proxy Entwurfsmuster umgesetzt. Abschnitt 5.1 beschreibt Details der Implementierung.

Nach dem Durchlaufen der Phase Extraktion stehen Daten in den verschiedenen XML-Ausgabeformaten der verwendeten Anwendungen zur weiteren Verarbeitung bereit.

## 4.5 INTEGRATION

In der Integrations-Phase werden Metadaten aus der Extraktions-Phase zusammengeführt und gespeichert. Dabei kann für die Publikationen auf drei verschiedene XML-Formate zugegriffen werden:

- ParsCit XML mit Metadaten zur Publikation und Metadaten zur Liste der Referenzen
- GROBID XML mit Metadaten zur Publikation
- GROBID XML mit Metadaten zur Liste der Referenzen

Zur Integration sollen die Komponenten aus Abb. 17 umgesetzt werden. Neben den Komponenten zur Hadoop Anbindung sind dies vier geplante Komponenten. Die Komponente *InspireInterfaces* enthält Definitionen zu Schnittstellen, mit denen ein Komponenten übergreifender, konsistenter Zugriff auf Datenstrukturen realisiert wird. Diese werden zum einen von der Hauptkomponente *Integration* und zum anderen von den Komponenten *ParsCit* und *GrobidParser* genutzt, mit denen auf XML-Daten zugegriffen werden kann.

Zur Verarbeitung der vorliegenden Daten ergeben sich die folgenden Designfragen, die anschließend beantwortet werden.

### 4.5.1 *Design-Fragen*

I.1 Welche Metadaten werden zur Integration verwendet?

I.2 Wie soll der Zugriff auf die verfügbaren Metadaten geschehen?

I.3 In welcher Weise kann eine konsistente Verwendung der Metadaten sichergestellt werden?

I.4 Wie soll verfahren werden, falls Daten fehlen?

I.5 Welche Datenformate sollen verwendet werden?

I.6 Wie werden Metadaten zu Publikationen integriert?

I.7 Wie werden Metadaten zu Listen mit Referenzen integriert?

### 4.5.2 *Design-Entscheidungen*

#### *I.1 Verwendung von Metadaten*

Im Hinblick auf eine Identifikation und Deduplikation von Publikationen muss eine Menge von Metadaten-Feldern festgelegt werden, die durchgängig für eine Zuordnung von Publikationen verwendet werden kann. Basierend auf der Analyse aus Abschnitt 3.1 werden die Felder *Titel*, *Nachnamen von Autoren*, *Publikationsjahr* und *DOI* dafür festgelegt. Die Kombination aus Titel und Autorennamen eignet sich, um eine Publikation zu identifizieren. Da in vielen Fällen die Vornamen von Autoren nicht bekannt sind, und auch bei der Extraktion Anfangsbuchstaben von Vornamen Inkonsistenzen vorkommen, ist die alleinige Verwendung von Nachnamen vorteilhaft. Zusätzlich dazu sollen Angaben zum Jahr einer Veröffentlichung eine Identifizierung unterstützen. Mit der Verwendung von DOIs steht ein Konstrukt zur Verfügung, mit dem eine eindeutige Identifizierung möglich ist. Diese Angabe liegt nach der Analyse aus Abschnitt 3.1 lediglich bei 17,74 % der Eingabedokumente vor. Die DOI kann daher nicht als alleiniger Identifikator ohne die drei vorher genannten Felder genutzt werden.

#### *I.2 Zugriff auf die extrahierten Daten*

Die Daten, die aus der Extraktions-Phase zur Verfügung stehen, liegen in XML-Formaten vor. Um einen performanten Zugriff zu ermöglichen, werden die Daten mittels SAX-Parsern analysiert. Dies hat verschiedene Vorteile. SAX-Parser sind schneller als die Alternative der DOM-Parser und beanspruchen weniger Speicher. Die durchlau-

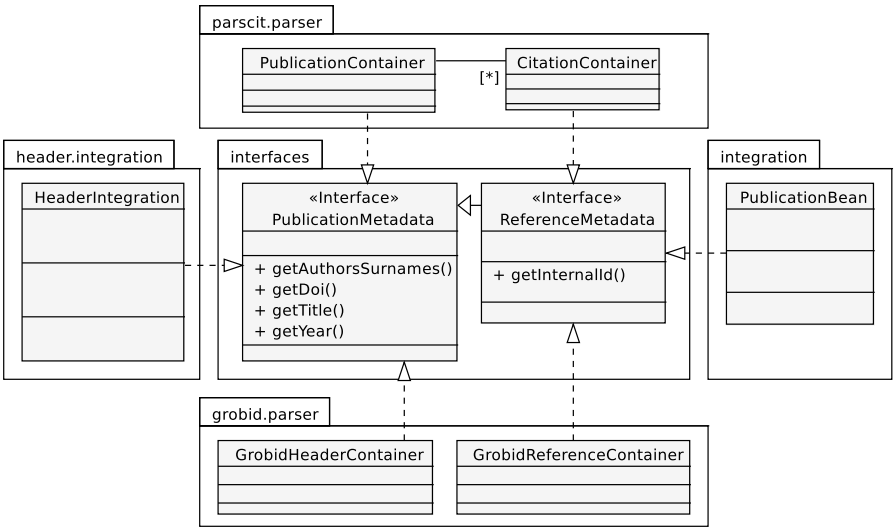



Abbildung 18: Nutzung der Schnittstellen zum Datentransfer zwischen Komponenten

fene XML-Struktur wird bei SAX-Parsern nicht komplett im Speicher gehalten. Für die zu implementierenden SAX-Parser sollen zudem Container-Klassen definiert werden, die zur Datenhaltung extrahierter Elemente dienen. Sowohl die SAX-Parser als auch die Container-Klassen sind Bestandteile der Komponenten *ParsCit* und *GrobidParser* aus Abb. 17. Zum Datenzugriff werden die nachfolgend definierten Schnittstellen implementiert.

### *I.3 Schnittstellen zur konsistenten Verwendung*

Um einen konsistenten Zugriff auf zu verarbeitende Daten zu gewährleisten, werden Schnittstellen der Komponente *Interfaces* verwendet. Das Java-Paket *interfaces*, das im Folgenden genannt wird, ist Bestandteil der *Interfaces*-Komponente.

Abbildung 18 zeigt eine Übersicht der Verwendung der Schnittstellen *PublicationMetadata* und *ReferenceMetadata* im Paket *interfaces*. Daneben sind Klassen aus anderen Paketen der Integrations-Phase abgebildet, die jeweils eine der Schnittstellen realisieren. An dieser Stelle steht die Verwendung der Schnittstellen im Vordergrund, daher sind ausschließlich Methoden aus diesen aufgeführt.

Die abgebildete Schnittstelle *PublicationMetadata* definiert Methoden, mit denen auf die Attribute zugegriffen werden kann, die zur Identifizierung von Publikationen genutzt werden sollen: Titel, Nachnamen von Autoren, Publikationsjahr und DOI (siehe Abschnitt 4.5). Diese stellen die wichtigsten Daten dar, daher werden die entsprechenden Methoden zum Zugriff sowohl im Paket *parscit.parser*, als auch in *grobid.parser* bereitgestellt und können für die Integration genutzt werden. Nach der durchgeführten Integration sollen die zusammenge-

führten Daten über die gleiche Schnittstelle zugegriffen werden. Diese Bereitstellung geschieht über die Klasse *HeaderIntegration* im Paket *header.integration*.

Die Schnittstelle *ReferenceMetadata* erweitert PublicationMetadata um die Methode *gerInternalId()*. Diese Schnittstelle wird für Listen von Referenzen genutzt. Die zusätzliche Methode *gerInternalId()* gibt eine ID zurück, welche zur Zuordnung von genutzten Referenzen in dieser und der Deduplikations-Phase dient.

Mit der beschriebenen Verwendung der Schnittstellen wird eine konsistente Behandlung der Metadaten unterstützt.

### *I.4 Verfahren bei nicht verfügbaren Daten*

Abhängig davon, wie in den Phasen Konvertierung und Extraktion Daten verarbeitet werden konnten, stehen für eine ursprüngliche Eingabedatei unterschiedlich viele Daten zur Verfügung. Es kann zwischen drei Fällen unterschieden werden: (I.4.1) Es konnten keine Daten extrahiert werden, (I.4.2) Es konnten nur durch eine Software Daten extrahiert werden, (I.4.3) Es stehen zwei unterschiedliche Datentypen zur Verfügung. Jeder dieser Fälle kann dabei jeweils für Metadaten zu einer Publikation und Metadaten zu den Referenzen der Publikation auftreten.

In der Integration soll, je nach Fall, folgendermaßen mit den Daten verfahren werden: (I.4.1) Es wird keine weitere Verarbeitung durchgeführt, (I.4.2) Der verfügbare Datensatz wird verwendet, ohne dass eine Integration notwendig ist, (I.4.3) Die beiden Datentypen werden in ein integriertes Format zusammengeführt.

### *I.5 Verwendete Datenformate*

Dieser Abschnitt soll einen Überblick der verwendeten Datenformate während der Integration geben. Wie eingangs erwähnt, liegen die bereits extrahierten Daten im XML-Format vor. In I.2 wurde zudem festgelegt, dass die Ergebnisse der zu erstellenden SAX-Parser zunächst als Instanzen von Container-Klassen vorliegen.

Abbildung 19 zeigt die verwendeten Formate und den Datenfluss zwischen den Klassen der Integration. Zunächst werden im *Integration-Mapper* Daten der drei XML-Formate aus HBase gelesen und an die *IntegrationApi* übergeben. Die XML-Daten werden anschließend von den jeweiligen Parsern ausgewertet und Container-Objekte zur eigentlichen Integration zurückgegeben. Nach der Durchführung der Integration werden die Ergebnisse als JSON Daten in HBase gespeichert.

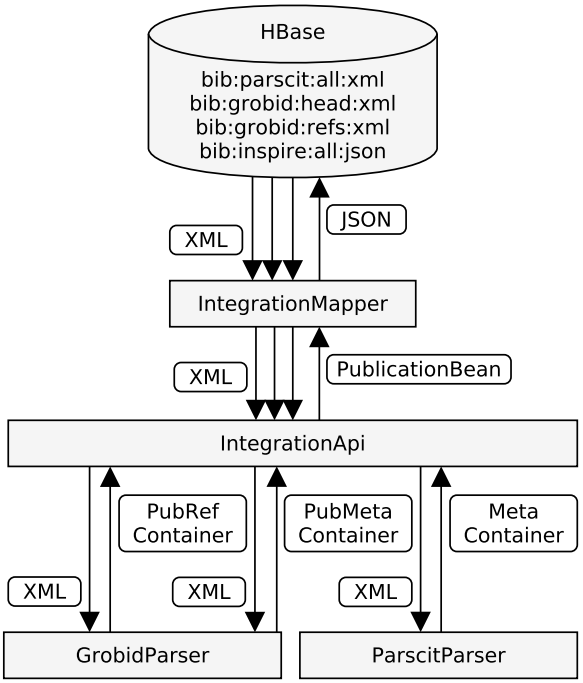



Abbildung 19: Datenfluss und Datenformate der Integration

Die JSON-Daten werden von einem *PublicationBean*-Objekt erzeugt. Dies ist als JavaBean realisiert, mit dem die enthaltenen Daten serialisiert werden können. Es dient also dazu, die zur Laufzeit erzeugten Variablen in einen String zu überführen, der abgespeichert werden kann. Die zu entwickelnde *PublicationBean*-Klasse wird zudem einen Konstruktor enthalten, der als Parameter einen JSON String erwartet. Dadurch wird ein Objekt zur Weiterverwendung der Metadaten erzeugt. Damit wird eine komfortable Verwendung derselben Metadaten in der Integrations- und Deduplikations-Phase ermöglicht.

*I.6 Wie werden Metadaten zu Publikationen integriert?*

Die Datenqualität der extrahierten Metadaten variiert stark. Der wichtigste Schritt zur einheitlichen Verwendung der Daten stellt daher eine Normalisierung dar. Dadurch wird die Menge der erlaubten Zeichen reduziert und die Wahrscheinlichkeit eines Matchings identischer Daten erhöht. Im Folgenden ist sind die verwendeten Schritte zur Normalisierung der verwendeten Zeichenketten aufgeführt. Die zu ersetzenden Zeichenketten wurden durch eine nicht-automatische, explorative Untersuchung vorhandener Datensätze gefunden. In den folgenden Listen werden Tupel genutzt, in denen der erste Bestandteil durch den zweiten Bestandteil ersetzt wird.

1. Falls der Titel mit „http://“ startet, verwende Titel ohne Normalisierung
2. Ersetze die Umlaute (Ä, Ae), (ä, ae), (Ö, Oe), (ö, oe), (Ü, Ue) und (ü, ue)
3. Ersetze die Zeichenketten (A¨, Ae), (a¨, ae), (O¨, Oe), (o¨, oe), (U¨, Ue) und (u¨, ue)
4. Entferne die Satzzeichen „.“, „!“, „?“ und „-“
5. Entferne alle diakritischen Zeichen[1]
6. Ersetze das Zeichen (ß, ss)
7. Entferne alle Zeichen außer Buchstanben, Zahlen und dem Leerzeichen
8. Entferne alle Leerzeichen am Anfang und am Ende der Zeichenkette

#### NACHNAMEN VON AUTOREN

1. Verfahre wie bei einem Titel
2. Entferne alle Zahlen
3. Ersetze das erste Zeichen durch das erste Zeichen als Großbuchstanbe
4. Ersetze alle weiteren Zeichen durch entsprechende Kleinbuchstaben

#### PUBLIKATIONSJAHR

1. Entferne alle Zeichen außer Zahlen
2. Verwerfe Publikationsjahr, wenn es kleiner als 1000 ist
3. Verwerfe Publikationsjahr, wenn es größer als das Folgejahr das aktuellen Jahres ist

#### DOI

1. Verwende DOI, falls sie nicht dem Nullwert entspricht

Bei der Integration stehen jeweils bis zu zwei Ausprägungen der Metadaten zur Verfügung. Es können also beispielsweise zwei Titel vorliegen, aus denen der Titel abgeleitet wird, der letztendlich genutzt werden soll. Falls innerhalb der Extraktion für Datensätze einzelne Felder nicht gefunden werden konnten, stehen möglicherweise weniger als zwei Ausprägungen zur Verfügung. Liegt nur eine Ausprägung vor, wird diese genutzt. Bei Fällen, in denen kein Titel oder kein Autor zur Verfügung steht, kann der zugehörige Datensatz nicht verwendet werden.

1 Nutzung von *java.text.Normalizer* mit kanonischer Dekomposition (NFD)

Bei der Integration der Metadaten zu Publikationen stehen keine Zusatzinformationen darüber zur Verfügung, wie hoch die Wahrscheinlichkeit dafür ist, dass die vorliegenden Daten korrekt sind. Daher wird auf Heuristiken zurückgegriffen, mit denen gute Ergebnisse erzielt wurden.

Im Fall von extrahierten Titeln kommt es vor, dass Zeichenketten, die korrekten Titeln entsprechen, weitere Zeichen angehängt werden. Daher wurden zur Integration von Titeln in der ersten entwickelten Version die ersten überlappenden Zeichen genutzt. In Einzelfällen entstand durch dieses Verfahren ein Fehler. Zeichenketten, in denen nur der erste Buchstabe übereinstimmt, wurden auf genau dieses Zeichen gekürzt. Dieses Verfahren wurde durch eine Heuristik ersetzt, bei der dieser Fehler nicht auftritt und die gute Ergebnisse liefert. Für die Integration von zwei vorliegenden Titeln wird die kürzere von zwei vorliegenden Zeichenketten gewählt.

Für die Integration von Autoren liegen zwei Listen mit Nachnamen vor. Diese werden in eine gemeinsame Liste überführt. Dazu werden zunächst alle Nachnamen der zwei Eingabelisten der gemeinsamen Liste hinzugefügt, falls der jeweilige Nachname nicht bereits in der Ausgabeliste vorhanden ist. Anschließend wird geprüft, ob in den einzelnen Eingabelisten Nachnamen mehrfach auftreten. Tritt ein solcher Fall auf, werden die mehrfach auftretenden Nachnamen hinzugefügt. Durch dieses Verfahren wird erstens eine Generierung von Duplikaten vermieden. Beispielsweise wird aus zwei Listen (schmidt) und (mueller, schmidt) die Liste (mueller, schmidt) generiert. Zweitens werden doppelt vorkommene Nachnamen berücksichtigt. Aus den Listen (schmidt, schmidt) und (mueller, mueller) wird die Liste (mueller, mueller, schmidt, schmidt) generiert. Die Reihenfolge der in Publikationen aufgeführten Autoren wird hierbei nicht berücksichtigt. Dies ist eine bewusste Designentscheidung. Die Verwendung der Nachnamen von Autoren dient der Identifizierung von Publikationen durch ihre Metadaten. Die Reihenfolge der Autorennamen kann für eine Identifizierung als zweitrangig eingestuft werden. Dies ist insbesondere der Fall, da auch zusätzliche Metadaten zur Identifizierung genutzt werden. Da außerdem bekannt ist, dass die verwendeten Eingabedaten fehlerhaft sein können, dient dieses Verfahren im Hinblick auf die Identifizierung in einem Dokumentennetzwerk zur Verbesserung der Datenqualität.

Zur Integration des Publikationsjahres und der DOI werden die verfügbaren Zeichenketten normalisiert. Wird in einem der Fälle ein falscher Wert festgestellt (z.B. eine Angabe zu einem Publikationsjahr, die aus Buchstaben zusammengesetzt ist), so wird der jeweils andere Wert genutzt.

*I.7 Wie werden Metadaten zu Listen mit Referenzen integriert?*

Für die Integration von Referenzen aus Literaturverzeichnissen von Publikationen stehen Listen mit Metadaten-Einträgen zur Verfügung. Diese Listen stimmen im optimalen Fall überein und enthalten exakt diejenigen Metadaten, die in der zugehörigen Publikation verwendet werden.

Die zur Verfügung stehenden Listen weichen sowohl in der Anzahl ihrer Einträge als auch in den enthaltenen Werten voneinander ab. Auch hier werden regelbasierte Heuristiken verwendet, um Daten für die Verwendung miteinander zu modifizieren und somit die Datenqualität einer Identifizierung von Publikationen über Metadaten zu verbessern. Dazu werden die Listeneinträge zunächst wie im vorhergehenden Abschnitt normalisiert und anschließend paarweise miteinander verglichen.

Der Vergleich und die Identifizierung von Einträgen, die eine gleiche Publikation beschreiben, verwendet in erster Linie die Levenshtein-Distanz [Lev66] zwischen Titeln. Mit der Levenshtein-Distanz wird das Minimum von Operationen (Einfügen, Ersetzen oder Löschen) ermittelt, die benötigt werden, um aus einer Zeichenkette eine zweite zu formen. Es werden jeweils Titel aus Metadaten-Einträgen der zwei unterschiedlichen Listen verwendet. Die Integration von Listen setzt sich aus drei Schritten zusammen:

1. Eine absolute Übereinstimmung der Titel entspricht einer berechneten Levenshtein-Distanz von 0. Bei einem solchen Fall wird angenommen, dass die zugehörigen Metadaten-Einträge die gleiche Publikation beschreiben. Die beiden Einträge werden wie im vorigen Abschnitt zusammengeführt und einer dritten Liste hinzugefügt. Die Einträge werden nicht mehr für weitere Vergleiche verwendet.
2. Eine starke Übereinstimmung der Titel liegt vor, wenn die Levenshtein-Distanz einen Schwellenwert nicht übersteigt. Es wurden gute Ergebnisse mit einem festen Schwellenwert von 5 erzielt. Weitere Möglichkeiten bestehen in der Verwendung von dynamisch gewählten Werten. Beispielsweise einem prozentualen Anteil des Maximums oder Minimums der Länge der Zeichenketten. Einträge, die eine starke Übereinstimmung aufweisen, werden der dritten Liste hinzugefügt und nicht mehr für weitere Vergleiche verwendet.
3. Für verbleibende Einträge konnte keine Übereinstimmung der Titel festgestellt werden. Es handelt sich also um Einträge der beiden Eingabelisten, zu denen kein entsprechender Eintrag der anderen Liste finden lies. Es wird angenommen, dass kein solcher Eintrag existiert und die verbleibenden Einträge in die dritte Liste übernommen.

List 1

| # | 1-1 |
|---|---|
| Titel | aco routing survey |
| Authors | mueller |
| Year | 2012 |
| DOI | - |

| # | 1-2 |
|---|---|
| Titel | visualizing bigdata |
| Authors | mueller |
| | schmidt |
| Year | 2013 |
| DOI | - |

| # | 1-3 |
|---|---|
| Titel | trends in compression th |
| Authors | fischer |
| | jones |
| Year | - |
| DOI | - |

List 2

| # | 2-1 |
|---|---|
| Titel | visualizing bigdata |
| Authors | mueller |
| | schmidt |
| Year | - |
| DOI | - |

| # | 2-2 |
|---|---|
| Titel | trends in compression |
| Authors | fischer |
| | schneider |
| | jones |
| Year | 2013 |
| DOI | - |

Result list

| # | 1 |
|---|---|
| Titel | visualizing bigdata |
| Authors | mueller |
| | schmidt |
| Year | 2013 |
| DOI | - |

| # | 2 |
|---|---|
| Titel | trends in compression |
| Authors | fischer |
| | schneider |
| | jones |
| Year | 2013 |
| DOI | - |

| # | 3 |
|---|---|
| Titel | aco routing survey |
| Authors | mueller |
| Year | 2012 |
| DOI | - |

Abbildung 20: Beispiel zur Integration von Referenzen

Die neu erstellte Liste wird als integrierte Version der beiden Listen der Eingabe verwendet. Abb. 20 zeigt ein Beispiel dazu. Bei einem Vergleich der Levenshtein-Distanzen der Einträge *1-2* aus *List 1* und *2-1* aus *List 2* wird dabei eine Distanz von 0 für die Titel festgestellt. Sie werden zusammengeführt, der entstehende Eintrag zu *Result list* hinzugefügt und aus den ursprünglichen Listen entfernt. Für die Einträge *1-3* und *2-2* wird eine Levenshtein-Distanz von 3 festgestellt und die Einträge ebenfalls integriert. Zuletzt wird der Eintrag *1-1* unverändert der Ergebnisliste hinzugefügt.

## 4.6 DEDUPLIKATION

In der Deduplikations-Phase werden einzelne Datensätze von Publikationen zu einem Netzwerk zusammengesetzt. Dieses kann anschließend analysiert und für die Berechnungen von Literaturempfehlungen genutzt werden.

In dieser Phase kann für die Publikationen, die als PDF-Dateien vorliegen, auf extrahierte Metadaten zugegriffen werden. Diese liegen im integrierten Format vor, es wird also nicht mehr zwischen verschiedenen Formaten der Extraktionstools unterschieden. Die Metadaten beschreiben zum einen die Publikation selbst und zum anderen die Referenzen aus dem Literaturverzeichnis der Publikation.

Bei der bevorstehenden Zusammensetzung eines Netzwerks sollen Metadaten, die dieselbe Publikation beschreiben, auf denselben Datensatz im Netzwerk abgebildet werden. Dabei soll vermieden werden, dass eine Publikation mehrfach als verschiedene Datensätze im

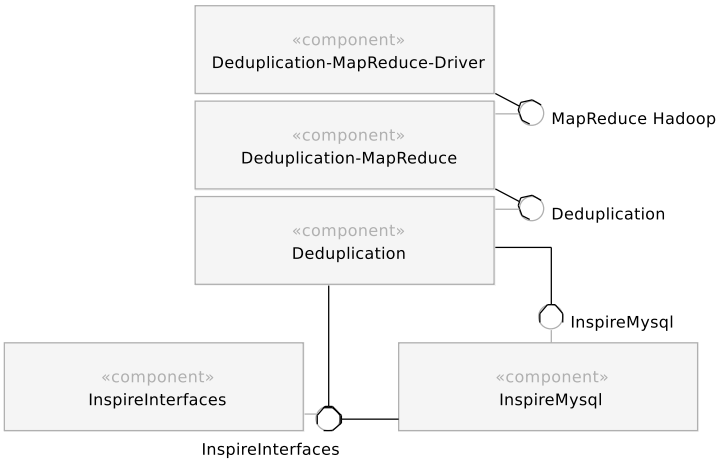



Abbildung 21: Komponenten der Phase Deduplikation

Netzwerk vorkommt. Die Vermeidung der Generierung solcher Duplikate ist die Deduplikation.

Abbildung 21 zeigt die in dieser Phase verwendeten Komponenten. Für die parallele Verarbeitung werden die Komponenten *Deduplication-MapReduce-Driver* und *Deduplication-Mapreduce* genutzt. In der Komponente *Deduplication* werden Daten gelesen, die in der Extraktions-Phase erstellt wurden. Um in der gleichen Weise auf die Daten zuzugreifen, findet die Komponente *InspireInterfaces* erneut Verwendung. Für die Datenhaltung des zu generierenden Dokumentennetzwerks wurde die Datenbank MySQL gewählt. Der Zugriff darauf geschieht in der Komponente *InspireMysql*. Vor einem möglichen Zugriff wird zunächst ein Entity-Relationship-Modell erstellt und damit die zugreifbaren Elemente definiert.

### *Entity-Relationship-Modell der MySQL Datenhaltung*

Das zur Datenablage verwendete MySQL-Datenbankschema ist im Entity-Relationship-Modell in Abb. 22 dargestellt.

Mittelpunkt des Modells bildet die Tabelle *publication*, in der Metadaten zu einer Publikation abgelegt werden können. Ein zusätzliches und wichtiges Feld ist *id*. Dieses wird verwendet, um Einträge innerhalb der Datenbank eindeutig zuordnen zu können. Dabei werden Angaben zum Publikationsjahr in der Tabelle *year* gespeichert.

Autoren stellen ebenfalls zur Deduplikation genutzte Metadaten dar werden durch ihren Nachnamen in der Tabelle *author* gespeichert. Die Zuordnung geschieht über die Tabelle *publication_has_author*. Datensätze zu Autoren werden nicht dedupliziert, so dass für Autoren mit gleichem Nachnamen der gleiche Datensatz verwendet werden kann.

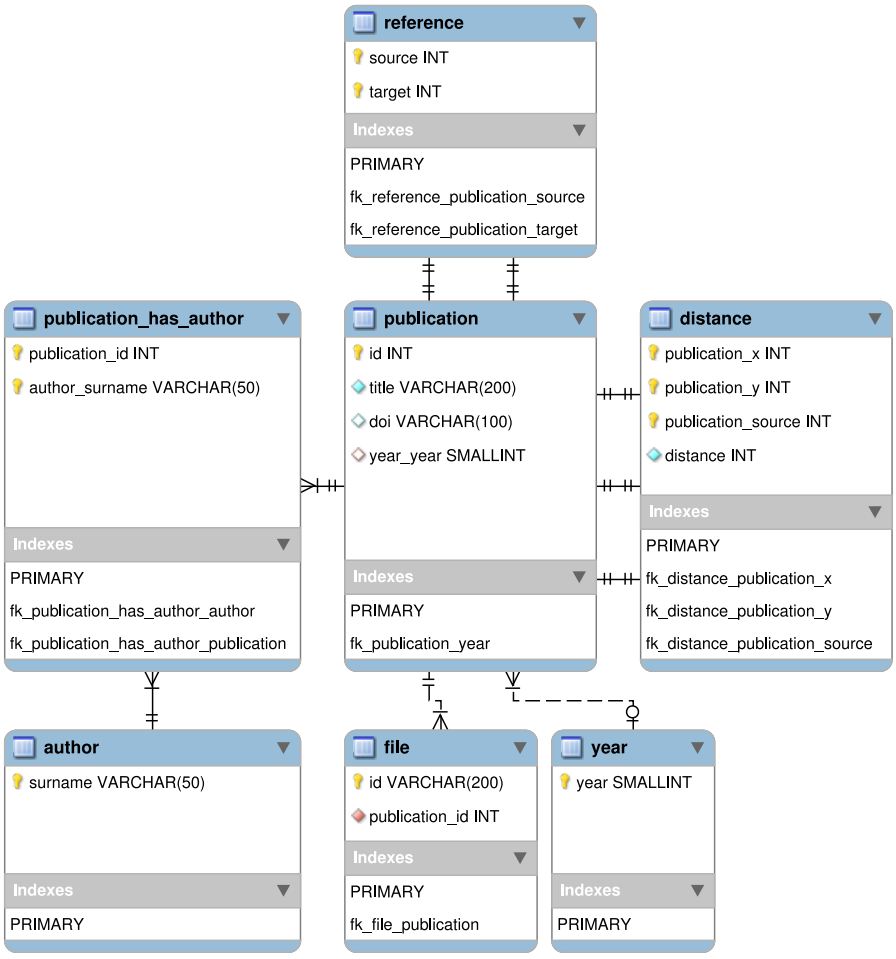



Abbildung 22: Entity-Relationship-Modell

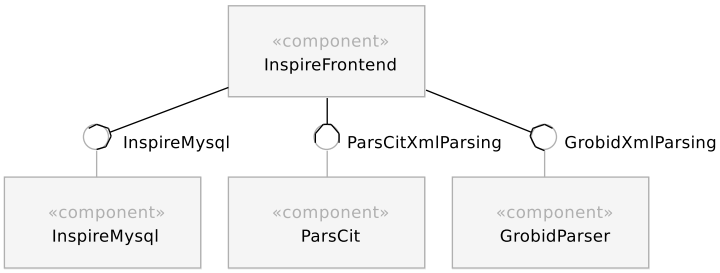



Abbildung 23: Komponenten der Phase Präsentation

Für Publikationen, die als PDF-Dateien vorliegen, wird die Tabelle file genutzt, um den zugehörigen Pfad zu speichern. Dadurch entsteht die Möglichkeit, Daten nach Zusammenhängen, die sich durch den Pfad ergeben, zu filtern. Zum Beispiel werden für alle Publikationen, die über die Webseite der Universität Paderborn gefunden wurden [PubUPB], das gleiche Verzeichnis genutzt. Diese können damit über einen Präfix-Vergleich des Pfades abgerufen werden.

Für Referenzen wird die Tabelle *references* genutzt. In dieser wird die ID einer referenzierenden Publikation *source* und den IDs der referenzierten Publikationen *target* angegeben.

Distanzen werden in der Tabelle *distance* gespeichert. Sie stellen eine Gewichtung des Zusammenhangs zweier Publikationen mit den IDs *publication_x* und *publication_y* dar. Die Gewichtung wird dabei als ganze Zahl angegeben. Zusätzlich wird das Feld *publication_source* genutzt, um die ID für diejenige Publikation abzulegen, in der die beiden übrigen Publikationen referenziert wurden.

Die zurückgegebenen IDs werden anschließend verwendet, um Relationen zwischen Publikationen und ihren Refrenzen zu erstellen. Abschließend werden die ermittelten IDs genutzt, um die Distanzen der Publikationen zu speichern.

## 4.7 PRÄSENTATION

Die Präsentations-Phase ist weitgehend unabhängig von den vorhergehenden Phasen. Das generierte Dokumentennetzwerk in der MySQL-Datenbank wird genutzt, um die Bibliometriken zu berechnen. Die Ergebnisse werden über die Benutzerschnittstelle präsentiert. Dabei besteht für Benutzer die Möglichkeit, im durch das Netzwerk zu navigieren und die Berechnung der Bibliometriken für weitere Publikationen anzufordern. Außerdem besteht für Benutzer die Möglichkeit, auf alle Metadaten zuzugreifen, die von den Anwendungen der Extraktion in HBase abgelegt wurden.

Abbildung 23 zeigt eine Übersicht der genutzten Komponenten. In der Komponente InspireFrontend wird der Großteil der benötigten

Funktionalitäten umgesetzt. Zusätzlich werden drei Komponenten angebunden und damit von der Wiederverwendbarkeit Gebrauch gemacht. Durch die Komponente InspireMysql werden die Berechnungen der Bibliometriken umgesetzt. Die Komponenten ParsCit und GrobidParser dienen dazu, XML-Daten der Extraktions-Phase zu parsen. Die geparsten Daten dienen als zusätzliche Informationen, die von Benutzern des Systems abgerufen werden können.

Im Fokus der Präsentations-Phase stehen die berechneten Bibliometriken, die als Literaturempfehlungen ausgegeben werden. Zusätzlich zu den einzelnen Bibliometriken

### 4.7.1 *Synthese der Bibliometriken*

Die Synthese der Bibliometriken ist die Berechnung finaler Literaturempfehlungen, basierend auf Ergebnissen der Bibliometrie und Metadaten zu Publikationen. Dazu werden die Variablen aus Tabelle 13 verwendet. Die Synthese beruht auf Einschätzungen der Semantik von Zusammenhängen im Publikations-Netzwerk. Die darauf aufbauenden Gewichtungen der ursprünglichen Werte können als einfache Variablen angesehen werden, die bei Bedarf angepasst werden können.

Zur Berechnung von Literaturempfehlungen werden alle im System bekannten Publikationen $\mathcal{P}$ betrachtet. Dabei dient eine Publikation $p \in \mathcal{P}$ als Eingabe und ist der alleinige Ausgangspunkt der Berechnungen. Hier wird vorausgesetzt, dass auf die Ergebnisse der Berechnung von Bibliometriken als einzelne, vollständige Listen zugegriffen werden kann. Werte aus diesen Listen betreffen Publikationspaare $(p, x)$, wobei $x$ eine weitere beliebige, aber fest gewählte Publikation aus $\mathcal{P}$ darstellt. Die vollständige Synthese ist in Formel 3 dargestellt; sie wird im Folgenden beschrieben.

$$\begin{aligned} \mathcal{S}_p(x) = [\, & 2 \cdot \mathcal{R}_p(x) + \widehat{\mathcal{R}}_p(x) + \mathcal{B}_p(x) + 2 \cdot \mathcal{C}_p(x) \\ & + \mathcal{D}_p(x) + \mathcal{J}_1(x) \,] \cdot \mathcal{J}_2(x) \end{aligned} \tag{3}$$

Zur Synthese wird ermittelt, ob $p$ eine Publikation $x$ referenziert. Existiert eine solche Referenz, wird für $\mathcal{R}_p(x)$ der Wert 1 gesetzt, andernfalls 0. Der umgekehrte Zusammenhang besteht, falls $x$ die Publikation $p$ referenziert. Auch in diesem Fall wird der Wert 1 genutzt, falls die Referenz existiert und falls nicht, dann 0. Diese Richtung der Referenzierung wird als $\widehat{\mathcal{R}}_p(x)$ bezeichnet. Besteht in keiner Richtung eine direkte Referenzierung zwischen $p$ und $x$, ist $\mathcal{R}_p(x) = \widehat{\mathcal{R}}_p(x) = 0$. Existiert genau eine dieser Referenzen, beträgt einer der beiden Werte 1, der andere 0. In seltenen Fällen kommt es vor, dass sich Publikationen gegenseitig referenzieren. Diese Möglichkeit entsteht, wenn

Tabelle 13: Variablen zur Synthese der Bibliometriken

| Var. | Beschreibung |
|---|---|
| $\mathcal{P}$ | Menge der bekannten Publikationen |
| $p$ | Publikation $p \in \mathcal{P}$ |
| $x$ | Publikation $x \in \mathcal{P}$, $x \neq p$ |
| $\mathcal{R}_p$ | Menge der Publikationen, die von $p$ referenziert werden |
| $\widehat{\mathcal{R}_p}$ | Menge der Publikationen, die $p$ referenzieren |
| $\mathcal{B}_p$ | Menge der Publikationen, die Referenzen mit $p$ teilen |
| $\mathcal{C}_p$ | Menge der Publikationen, die gemeinsam mit $p$ referenziert werden |
| $\mathcal{D}_p$ | Menge der Publikationen, die gemeinsam mit $p$ referenziert werden und der zugehörigen Distanzen |
| $\mathcal{R}_p(x)$ | 1 bei Existenz von $x$ in $\mathcal{R}_p$, sonst 0 |
| $\widehat{\mathcal{R}}_p(x)$ | 1 bei Existenz von $x$ in $\widehat{\mathcal{R}}_p$, sonst 0 |
| $\mathcal{B}_p(x)$ | Anzahl der gemeinsamen Referenzen von $p$ und $x$ |
| $\mathcal{C}_p(x)$ | Anzahl der Publikationen, die sowohl $p$ als auch $x$ referenzieren |
| $\mathcal{D}_p(x)$ | Summe der Distanz-Werte zu $p$ und $x$ |
| $j$ | Aktuelles Jahr |
| j($x$) | Publikationsjahr von $x$ |
| $\mathcal{J}_1(x)$ | 1, falls j($x$) = $j-2$<br>0, sonst |
| $\mathcal{J}_2(x)$ | 2, falls j($x$) >= $j-1$<br>1, sonst |

zwei Publikationen zeitgleich entstehen und die Autoren sich im Wissen der bevorstehenden Veröffentlichung gegenseitig referenzieren. In diesem Fall sind beide Einzelwerte 1. Obwohl es sich bei beiden Zusammenhängen um direkte Referenzierungen handelt, soll bei der Gewichtung ein Unterschied gemacht werden. Eine Referenzierung $\mathcal{R}_p(x)$ kann als unmittelbare Literaturempfehlung der Autoren von Publikation $p$ gedeutet werden. Diese Sichtweise ist ausschlaggebend dafür, dass Werte von $\mathcal{R}_p(x)$ mit einem Faktor 2 gewichtet werden. Im Gegensatz dazu werden eingehende Referenzen $\widehat{\mathcal{R}}_p(x)$ einfach gewichtet. Die Überlegung, eingehende Referenzen ebenfalls stärker zu gewichten, weil sie im Regelfall aktueller sind, wurde im Laufe der Entwicklung durch die Einführung von $\mathcal{J}_1(x)$ und $\mathcal{J}_2(x)$ hinfällig.

Der Wert der Bibliografischen Kopplung $\mathcal{B}_p(x)$ ist die Anzahl der Referenzen, die sowohl in $p$, als auch in $x$ verwendet werden. Damit kann dieser Wert als ein möglicher Indikator gemeinsamer Grund-

lagen der Publikationen angesehen werden. Da der Wert der Bibliografischen Kopplung höher ausfallen kann als der der direkten Referenzierungen, kann dieser potenziell auch bei einfacher Gewichtung einen größeren Teil des Gesamtwertes der Synthese ausmachen. Das entstehende Verhältnis der drei bisher behandelten Bibliometriken wird als angemessen bewertet, und daher wird für $\mathcal{B}_p(x)$ eine einfache Gewichtung festgesetzt.

Mit der Kozitation $\mathcal{C}_p(x)$ wird die Anzahl der Publikationen bestimmt, die Referenzen zu beiden Publikationen, $p$ und $x$, aufweisen. Da Beobachtungen für die Kozitation als besseren Indikator für thematische Ähnlichkeit als die Bibliografische Kopplung sprechen (vgl. Abschnitt 2.1.3), soll $\mathcal{C}_p(x)$ mit einem Faktor 2 gewichtet werden.

Als eine Erweiterung der Kozitation kann die Distanz von Zitationen $\mathcal{D}_p(x)$ betrachtet werden. Hierbei wird zusätzlich zur Anzahl der gemeinsamen Vorkommen zweier Publikationen die Distanz der Vorkommen einbezogen. Für die Distanz werden innerhalb der Synthese die Distanz-Klassen und -Gewichtungen aus Tabelle 14 (Seite 78) genutzt. Die ursprünglich festgesetzten Gewichtungen wurden im Hinblick auf die praktisch verfügbaren Datensätze angepasst. Innerhalb der Implementierung besteht die Möglichkeit, dass für Publikation Daten über Referenzierungen, nicht aber über Distanzen vorliegen. Die Ursache hierfür ist die Verwendung verschiedener Software-Komponenten zur Extraktion. Mit Verwendung der ursprünglich festgesetzten Distanz-Gewichtungen entstünde die Möglichkeit, dass für eine Publikation ohne vorliegende Distanz kein zusätzlicher Wert einbezogen würde; für eine Publikation der Distanz-Klasse „Dokument“ jedoch schon. Dies hätte eine unausgeglichene Bewertung zur Folge. Daher wurden alle die Distanzwerte um den Wert 1 dekrementiert. Für Distanzen der Klasse „Dokument“ hat der Distanzwert keine Auswirkung auf das Gesamtergebnis. Gemeinsam zitierte Dokumente werden durch die Kozitation trotz dessen berücksichtigt. Da der Wert von Distanzen vergleichsweise hoch ausfallen kann, wird $\mathcal{D}_p(x)$ einfach gewichtet.

Die letzten beiden Werte, die in die Synthese einfließen, sind unabhängig von der Art der Referenzierungen zwischen den Publikationen. Es handelt sich um Verwendungen des Erscheinungsjahres einer Publikation $x$, das mit dem aktuellen Jahr vergleichen wird. Stammt eine Publikation $x$ aus dem aktuellen Jahr oder dem Vorjahr, wird die Summe der bisher verwendeten Werte mit einem Faktor $\mathcal{J}_2(x) = 2$ multipliziert. Dies soll aktuelle und womöglich eher unbekannte Literatur in den berechneten Empfehlungen bevorzugen. Bisherige Werte von Publikationen, die bereits zwei Jahre alt sind, werden zumindest noch um $\mathcal{J}_1(x) = 1$ erhöht.

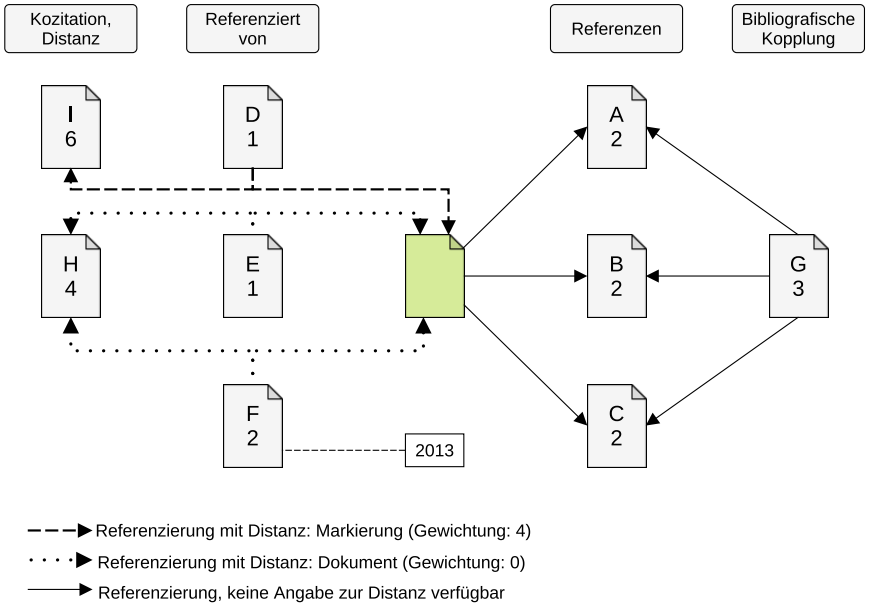



Abbildung 24: Beispiel zur finalen Gewichtung der Bibliometriken

*Beispiel zur Gewichtung der Bibliometriken*

Zur Anschauung zeigt Abb. 24 ein Beispiel zu den Gewichtungen in Formel 3 und den damit entstehenden Punkten für eine Literaturempfehlung. Die Publikation ohne Beschriftung sei dabei als Eingabe gewählt worden. Die Publikationen A, B und C werden referenziert und erhalten durch die doppelte Gewichtung für Referenzierungen 2 Punkte. Die Publikationen D und E referenzieren die Eingabepublikation und erhalten damit 1 Punkt. Die Publikation F referenziert die Eingabepublikation ebenfalls, ihre Punkte werden aufgrund des Jahres der Veröffentlichung verdoppelt. Publikation G steht durch die Bibliografische Kopplung über A, B und C mit der Eingabepublikation in Verbindung. Daraus ergeben sich 3 Punkte. Die Publikation H weist durch E und F zwei Kozitationen mit der Eingabepublikation auf. Durch die doppelte Gewichtung ergeben sich 4 Punkte. Für I ergeben sich durch die von D ausgehende Kozitation zunächst 2 Punkte. Da innerhalb von D eine Zitationsmarkierung gefunden wurde, die die Eingabepublikation gemeinsam mit I enthält, werden 4 Punkte addiert, daraus ergibt sich die Gesamtpunktzahl 6.

### 4.7.2 *Framework zur Benutzerschnittstelle*

Als Benutzerschnittstelle soll eine Webanwendung genutzt werden. Im Gegensatz zu einer Desktopanwendung wird damit von Benutzern keine zusätzliche Installation von Software verlangt. Lediglich ein Webbrowser wird benötigt.

Als Webframework soll das Play Framework 2.1 [Play] genutzt werden. Diese Designentscheidung wurde aufgrund von zwei Hauptmerkmalen getroffen.

Zum einen wurde das Play Framework entwicklerfreundlich gestaltet. Es ist nach dem Paradigma *convention over configuration* gestaltet und bietet unter anderem Unterstützung für die integrierte Entwicklungsumgebung Eclipse, die auch für die Entwicklung mit Hadoop bei dieser Masterarbeit eingesetzt wird. Das Play Framework ist quelloffen und folgt dem praxisbewährten Model-View-Controller Architekturmuster.

Zum anderen ist das Play Framework auf schnelle, skalierbare Webanwendungen ausgerichtet. Dies wird durch eine zustandslose Implementierung ermöglicht. Das Play Framework nutzt das Netty Framework [Nty], mit dem eine asynchrone Ein-/Ausgabe umgesetzt wird. Dadurch ist ein schneller Datendurchsatz möglich und benötigte Ressourcen können reduziert werden.

Das Play Framework wird u.a. für die Webseite LinkedIn [LIn] verwendet und kann damit als praxiserprobt angesehen werden. Insgesamt wird der Einsatz des Play Frameworks für die zu entwickelnde Benutzerschnittstelle als passend eingeschätzt.

# 5 IMPLEMENTIERUNG

Dieses Kapitel enthält Details zur Implementierung der konzipierten Komponenten. Dabei werden Abläufe der realisierten Klassen beschrieben.

Das Kapitel ist nach den Phasen unterteilt, die in der Konzeption definiert wurden. Dabei wird die Phase Konvertierung nicht behandelt. Für diese wird die Implementierung aus der Masterarbeit von Tobias Varlemann [Var13] verwendet.

Abschnitt 5.1 beschreibt die Implementierung der Extraktion. Der Ablauf der Integration wird in Abschnitt 5.2 behandelt. Abschnitt 5.3 enthält die Realisierung der Deduplikation. Abschließend wird die Umsetzung der Präsentations-Phase in Abschnitt 5.4 beschrieben.

## 5.1 EXTRAKTION VON METADATEN

In diesem Abschnitt wird die Umsetzung der Extraktion beschrieben. Dabei steht bei der Software ParsCit der Ablauf der Extraktion im Vordergrund. Für die Software GROBID wurde das Proxy Entwurfsmuster angewendet. In diesem Fall stellt die Architektur den interessanteren Teil der Implementierung dar und wird erläutert.

### 5.1.1 *ParsCit-Ausführung*

ParsCit ist eine Software, die größtenteils mit der Programmiersprache Perl realisiert wurde. Daher ist in der Java-Implementierung kein direkter Zugriff auf Objekte möglich, die während der Ausführung von ParsCit verwendet werden.

Für die lokale Ausführung von ParsCit soll daher ein neuer Prozess erzeugt werden. Abb. 25 zeigt den geplanten Ablauf einer *Parscit* Komponente von Aktionen als Aktivitätsdiagramm. In diesem Prozess wird ein Befehl ausgeführt, der zunächst aus Argumenten einer entsprechenden Java-Methode zusammengesetzt wird. Um eine Blockierung der verteilten Ausführung zu verhindern, kann eine maximale Laufzeit angegeben werden. Bei einer Überschreitung dieser wird der Prozess beendet.

Diese lokale Ausführung wird bei einer verteilten Berechnung in die Ausführung einer *Parscit-Mapreduce* Komponente eingebettet. Der Ab-

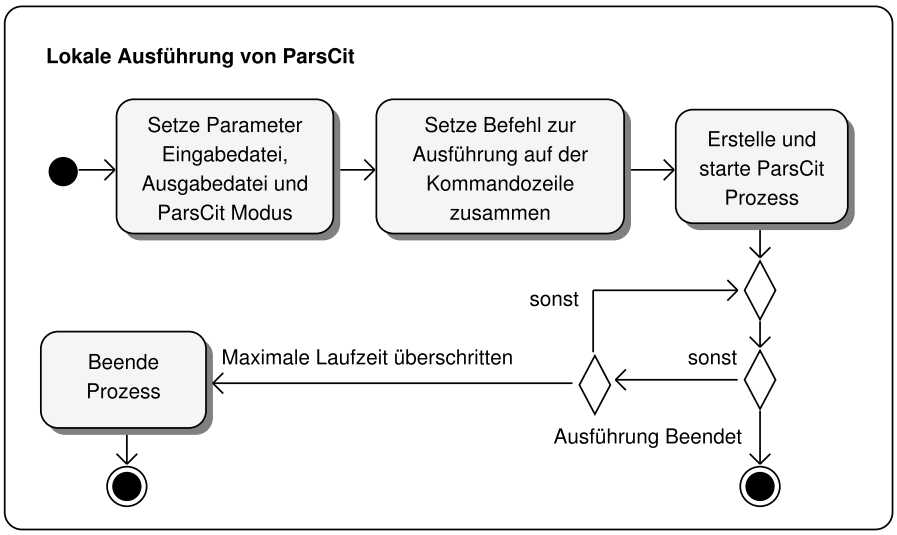



Abbildung 25: Ausführung eines ParsCit Prozesses

lauf zur Einbindung zeigt Abb. 26. Da der ParsCit Prozess als Parameter eine Eingabe- und eine Ausgabedatei erwartet, werden diese vor der Ausführung erstellt und danach entfernt. Die Daten zur Eingabe werden zunächst aus HBase gelesen und abschließend in HBase gespeichert.

### 5.1.2 *GROBID-Komponente*

Mit der GROBID-Komponente[1] werden Publikations-Metadaten extrahiert. Im Folgenden wird die Einbindung der GROBID-Software durch die GROBID-Komponente erläutert. Das Ziel der Einbindung ist die benötigten Methoden der GROBID-Software zu nutzen und dabei die Datenmenge, die über das Netzwerk geschickt werden muss, gering zu halten.

Als Eingabe für die Extraktion wird jeweils eine PDF-Datei verwendet. Die anschließende Durchführung der Extraktion geschieht durch Aufrufe zweier Methoden der GROBID-Software: *processHeaders()* zur Extraktion von Metadaten einer Publikation und *processReferences()* zur Extraktion von Metadaten der in der Publikation aufgeführten Referenzen.

Bei der Extraktion der Metadaten durch die GROBID-Software werden Conditional Random Fields (CRFs) verwendet. Während der Verarbeitung von Publikationsdaten wird dafür auf CRF-Modelldateien zugegriffen, die Bestandteil der GROBID-Software sind. Diese Modelldateien nehmen den Großteil der Gesamtgröße der Software ein.

1 Die *GROBID-Komponente* wird im Rahmen dieser Arbeit entwickelt, während die Software aus [Lopo9] hier zur Abgrenzung als *GROBID-Software* bezeichnet wird.

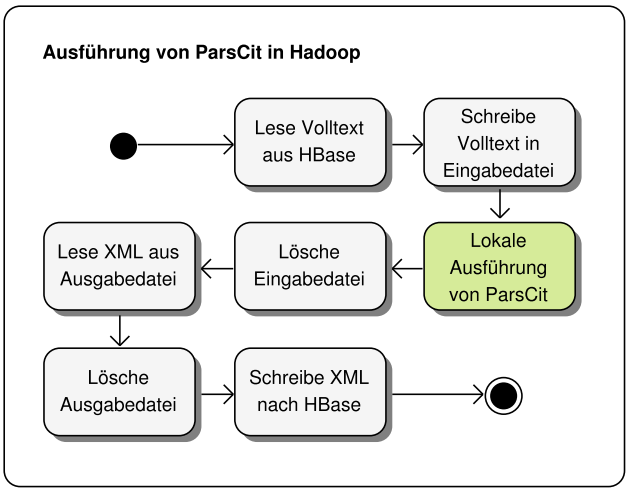



Abbildung 26: Ausführung von ParsCit in Hadoop

Insgesamt hat die GROBID-Software[2] eine Größe von etwa 2,1 Gigabyte; davon sind etwa 1,5 Gigabyte Modelldateien.

Die GROBID-Komponente soll verwendet werden, um Metadaten der Publikationen im HCPA Dokumentenkorpus zugreifbar zu machen. Da dieser mehrere Hunderttausend Dokumente umfasst, soll die Berechnung verteilt in einem Rechnercluster geschehen. Dies setzt voraus, dass die für die Extraktion benötigten Einzelkomponenten auf jedem Rechner im Cluster zugreifbar sind. Für die Verteilung der Software sind zwei verschiedene Lösungsansätze möglich. Grundsätzlich können Komponenten entweder während der Initialisierung eines Rechenauftrags verteilt werden oder sie werden bereits lokal vorgehalten. Im vorliegenden Fall soll die GROBID-Software aufgrund ihrer Größe lokal auf den einzelnen Rechnern vorgehalten werden.

Die GROBID-Software bietet einen Mechanismus an, mit dem die CRF-Modelldateien unabhängig von der Software im Dateisystem gespeichert werden können. Dafür kann die Methode *set_GROBID_HOME_PATH()* der Klasse *GrobidProperties* verwendet werden. Mit dieser wird zur Laufzeit festgelegt, an welcher Stelle im Dateisystem die benötigten Modelle zu finden sind. Damit können Code und Modelldateien unabhängig voneinander verwendet werden.

Mit der Auslagerung der Modelldateien verringert sich die Größe der Datenmenge, die noch über das Netzwerk verteilt werden muss, erheblich. Die innerhalb dieser Arbeit verwendeten Teile der GROBID-Software haben in Form einer Java-Archivdatei eine Größe von etwa 16 Megabyte. Um zusätzlich den Datenverkehr innerhalb des Clusters weiter zu verringern und kürzere Laufzeiten für externe Anbindungen zu ermöglichen wurde innerhalb dieser Arbeit eine Schnittstelle

2 GROBID in der Revision 629 des Softwareverwaltungssystems Subversion [GRO-SVN]

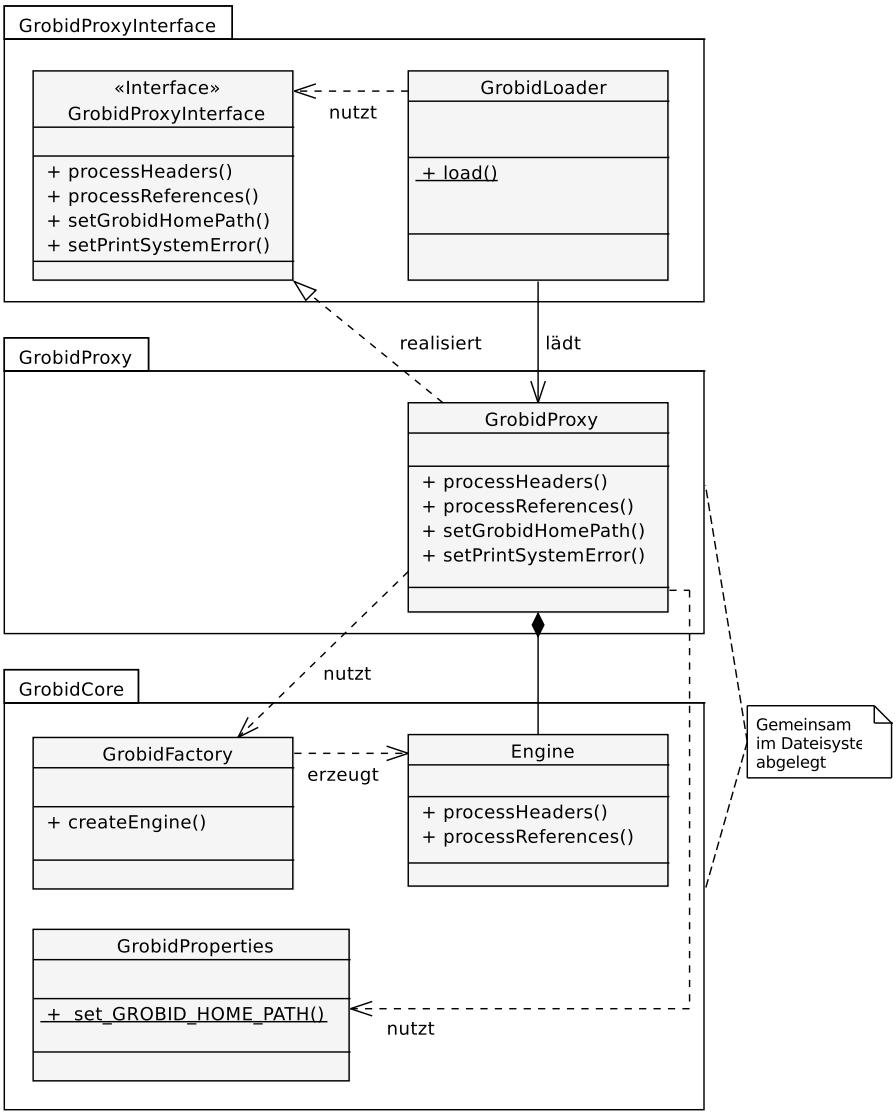



Abbildung 27: Pakete und Klassen zur Realisierung des Proxy Entwurfsmusters innerhalb der GROBID-Komponente

entwickelt, mit der auch diese Codeteile lokal im Cluster abgelegt werden können. Mit dieser Lösung beträgt die über das Netzwerk zu übertragene Datenmenge für die GROBID-Komponente noch 3,8 Kilobyte.

Die GROBID-Komponente wurde mit dem Proxy Entwurfsmuster (auch: Proxy Design Pattern) umgesetzt. Beim Proxy Entwurfsmuster wird der „[...] Zugriff auf ein Objekt mithilfe eines vorgelagerten Stellvertreterobjekts [kontrolliert]“ [GHJV11, S. 254]. Bei der GROBID-Komponente dient der Einschub des Proxy-Objekts zusätzlich der Auslagerung der GROBID-Software.

Das Klassendiagramm in Abbildung 27 zeigt die Zusammensetzung der Pakete *GrobidProxyInterface*, *GrobidProxy* und *GrobidCore*.

Das Paket *GrobidCore* ist Bestandteil der GROBID-Software. Durch Verwendung der statischen Methode *set_GROBID_HOME_PATH()* der Klasse *GrobidProperties* wird das Ressourcen-Verzeichnis festgelegt. Dies schließt insbesondere die Modelldateien ein. Über Aufrufe der Methoden *processHeaders()* und *processReferences()* der Klasse *Engine* wird die Extraktion der Metadaten angestoßen. Eine Instanz der Klasse *Engine* kann über einen Aufruf der Methode *createEngine()* der Klasse *GrobidFactory* bezogen werden.

Für das Proxy Entwurfsmuster wurde im Rahmen dieser Arbeit zunächst das Interface *GrobidProxyInterface* im gleichnamigen Paket definiert. Es umfasst die Methoden *processHeaders()* und *processReferences()* für die Extraktion, die Methode *setGrobidHomePath()* zur Konfiguration der Ressourcen und die Methode *setPrintSystemError()*.

Das Interface wird vom Stellvertreterobjekt *GrobidProxy* im gleichnamigen Paket realisiert. Aufrufe der beiden Methoden zur Extraktion werden an eine Instanz der Klasse *Engine* aus dem Paket *GrobidCore* weitergeleitet. Um diese Methoden verwenden zu können, muss zunächst einmalig die Methode *setGrobidHomePath()* aufgerufen werden, um damit das Verzeichnis der Modelldateien anzugeben. Dies geschieht durch eine Weiterleitung des Aufrufs an die Klasse *GrobidPoperties*. Für etwaige Fehlermeldungen kann die Methode *setPrintSystemError()* genutzt werden, über die Ausgabe von Hinweisen zur Entwicklung aktiviert wird.

Die beiden Pakete *GrobidProxy* und *GrobidCore* werden inklusive aller benötigten Abhängigkeiten in ein Java-Archiv gepackt und gemeinsam im Dateisystem abgelegt.

Um im finalen Schritt Zugriffe auf die gewünschten Methoden zu definieren, können externe Codeteile Objekte des Typs *GrobidProxyInterface* verwenden. Ein Objekt dieses Typs wird von der statischen Methode *load()* der Klasse *GrobidLoader* zurückgegeben. Dieses Objekt ist eine Instanz der Klasse *GrobidProxy*, die über den Java *URLClassLoader* nachgeladen wird. Insgesamt muss also lediglich die Implementierung der beiden Klassen des Pakets *GrobidProxyInterface* bekannt sein, um die Methoden zur Extraktion auszuführen. Das Java-Archiv mit dem Paket *GrobidProxyInterface* hat eine Gesamtgröße von 3,8 Kilobyte. Dies entspricht ca. $\frac{1}{4000}$ der 16 Megabyte bei Verwendung des ursprünglichen Codes und verringert somit den Datenverkehr.

## 5.2 INTEGRATION

In diesem Abschnitt wird der sequenzielle Ablauf der Integration beschrieben. Die Implementierung der realisierten Klassen repräsentiert ein Caching, das innerhalb eines Integrationsvorgangs Verwendung findet.

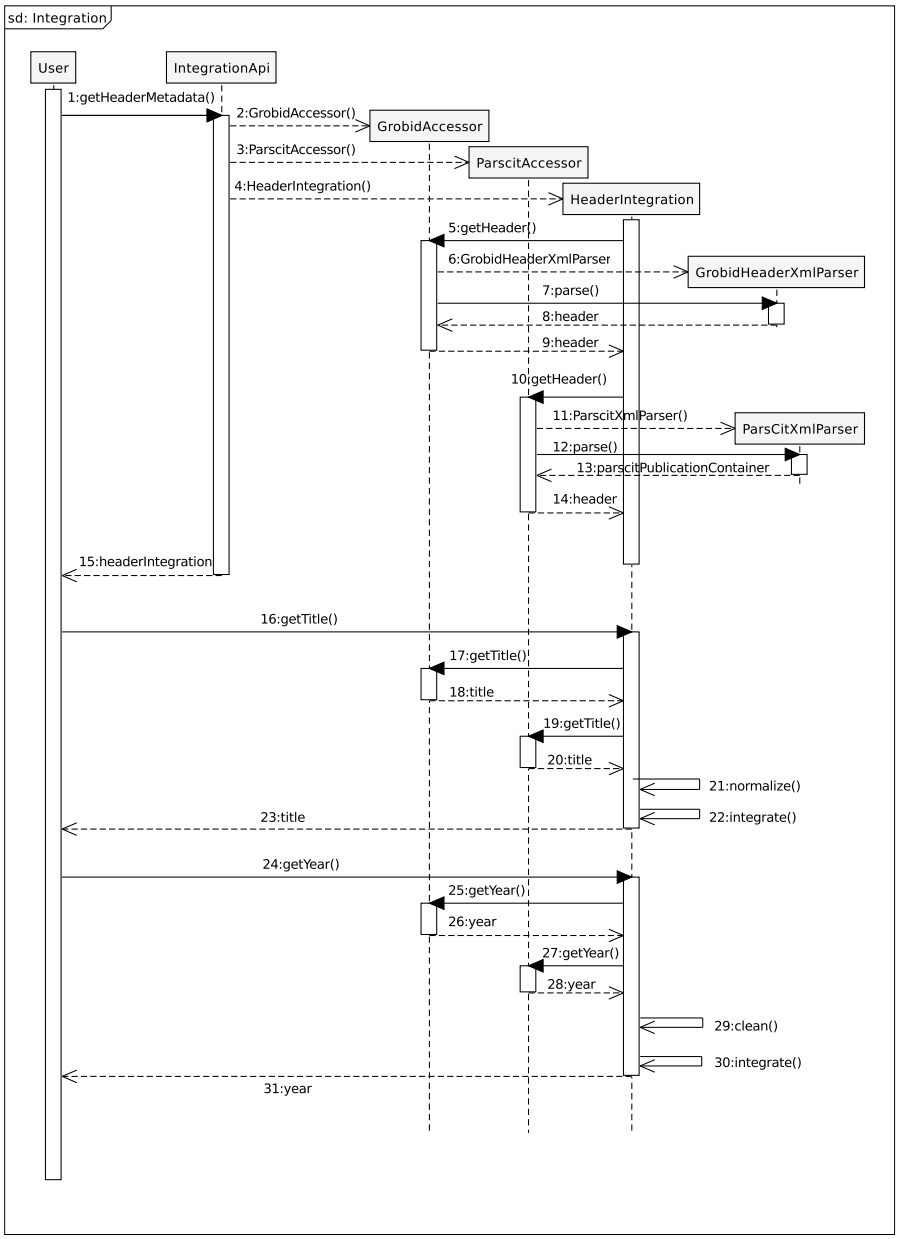



Abbildung 28: Interaktionen der Integration von GROBID und ParsCit Metadaten einer Publikation

Für die Integration von Metadaten stehen drei verschiedene Typen von XML-Daten aus der Extraktions-Phase zur Verfügung. Diese sind GROBID XML zu einer Publikation, GROBID XML zu Referenzen und ParsCit XML, das sowohl Daten zur Publikation als auch zu ihren Referenzen enthält. Um diese Daten zu integrieren, müssen die entsprechenden XML-Daten zunächst eingelesen werden. Hierfür wurden die drei Klassen *GrobidHeaderXmlParser*, *GrobidReferencesXmlParser* und *ParscitXmlParser* entwickelt. Diese Klassen sind SAX-Parser, die XML-Dokumente durchlaufen und die aus der Analyse (siehe Abschnitt 3.1) bekannten Felder extrahieren. Die extrahierten Werte werden Variablen aus Container-Klassen zugewiesen. Damit kann bei der weiteren Verarbeitung komfortabel auf gewünschte Werte zugegriffen werden.

Den Ablauf und die Interaktionen beim Datenzugriff zeigt Abb. 28. Zunächst wird die API-Methode zum Zugriff auf Metadaten zur Publikation aufgerufen (1). Es werden Objekte erstellt, die den Zugriff auf ParsCit- und GROBID-Daten steuern (2,3) und diese dem Konstruktor einer Instanz von *HeaderIntegration* übergeben (4). Diese ist zuständig für die Integration der einzelnen Datenfelder. Nachdem der Zugriff auf die GROBID-Metadaten eingeleitet wurde (5), wird ein Parser erstellt (6), der die entsprechenden XML Daten einließt (7). Die zurückgegebenen Daten verwenden die im vorigen Abschnitt vorgestellte Schnittstelle *PublicationMetadata* (8, 9). Für ParsCit-Daten wird ebenso verfahren (10-14). Damit wurden alle zur Verfügung stehenden Daten eingelesen und ein *HeaderIntegration* Objekt wird zurückgegeben (15). Bei einem Methodenaufruf zur Integration des Titels (16) wird dieser an die Objekte zum Zugriff auf GROBID- und ParsCit-Daten weitergereicht (17-20). Die zurückgegebenen Titel werden zunächst normalisiert (21). Das bedeutet unter anderem, dass Sonderzeichen ersetzt oder entfernt werden. Anschließend werden die Titel miteinander verglichen und zusammengeführt (22). Der ausgewählte oder generierte Titel wird zurückgegeben (23). Für die Anfrage des Jahres der Veröffentlichung wird ebenso verfahren (24-31). In diesem Fall werden Zeichen, die keine Ziffern darstellen, entfernt (29).

Technisch ist für diese Lösung des Zugriffs anzumerken, dass Instanzen der Klassen *GrobidAccessor* und *ParsCitAccessor* als Cache dienen und so jede XML-Datei nur einmalig gelesen werden muss. Auch diese Lesevorgänge sind durch die Verwendung von SAX-Parsern möglichst performant umgesetzt worden.

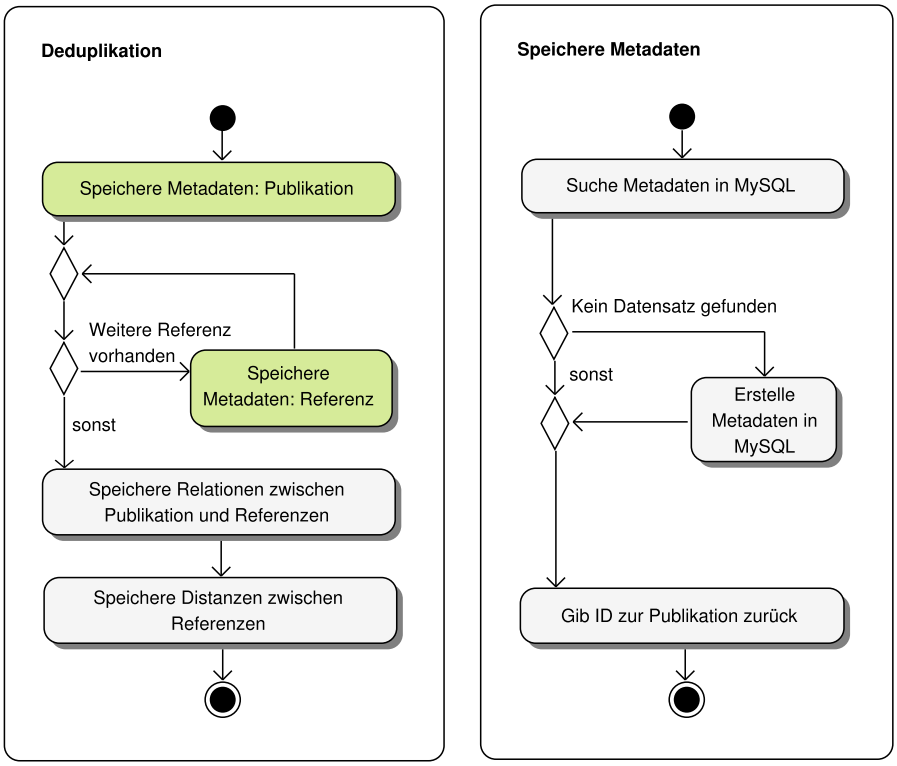



Abbildung 29: Deduplikations-Prozess

## 5.3 DEDUPLIKATION

In diesem Abschnitt wird der Ablauf der Deduplikation beschrieben. Die in Abschnitt 4.6 konzipierten Komponenten werden genutzt, um ein Dokumentennetzwerk zu generieren.

Im Deduplikations-Prozess wird abwechselnd geprüft, ob Metadaten bereits in der Datenbank existieren und weitere Metadaten hinzugefügt. Abb. 29 zeigt den Prozess der Deduplikation. Zunächst wird geprüft, ob die Metadaten zur vorliegenden Publikation in der Datenbank vorhanden sind. Wird kein passender Datensatz zu den Metadaten gefunden, so wird ein neuer Datensatz erstellt. In jedem Fall wird die zugehörige ID zurückgegeben. In dieser Weise werden auch die IDs von referenzierten Publikationen ermittelt. Die Suche und Verwendung von IDs bereits vorliegender Datensätze stellt die Deduplikation dar.

Die in Abb. 29 verwendeten Metadaten entsprechen den Entitäten aus des Entity-Relationship (ER) Modells. Sie sind in Abb. 22 auf Seite 22 aufgeführt.

Das in der Deduplikations-Phase erstellte Dokumentennetzwerk wird in der Präsentations-Phase zur Berechnung der Bibliometriken verwendet.

## 5.4 PRÄSENTATION

Dieser Abschnitt behandelt Details der Implementierung der Präsentations-Phase. Diese umfasst die Berechnung von Bibliometriken sowie die grafische Benutzerschnittstelle. Im Folgenden wird zunächst die Anwendung der Bibliometrik „Distanz von Zitationen“ erläutert. Anschließend ist die Umsetzung Benutzerschnittstelle durch ein Web-Framework beschrieben.

### 5.4.1 *Distanz von Zitationen*

Für die Berechnung der Distanz von Zitationen sollen zwei Komponenten entwickelt werden. Zum einen ist dies die Komponente *Offset-Parser*, die Positionen verschiedener Text-Bestandteile ermittelt. Die zweite Komponente *DistanceCalculator* baut darauf auf und ist für die eigentliche Berechnung der Distanzen zuständig.

Der Komponente *OffsetParser* werden als Eingabe der Volltext einer Publikation und die Ergebnisse aus einem ParsCit Aufruf bereitgestellt. Damit stehen Überschriften und die für Zitierungen verwendeten Markierungen zur Verfügung. Mit diesen Eingaben sollen als Ausgaben die Positionen von Abschnitten, Absätzen, Sätzen und Zitations-Markierungen ermittelt werden.

Durch Verwendung der ermittelten Positionen soll die Komponente *DistanceCalculator* die minimalen Distanzen zwischen allen Zitations-Paaren ermitteln. Die Positionsangaben werden dazu verwendet, einen gegebenen Text in seine Bestandteile zu zerlegen. Durch einen Vergleich der Positionen der Markierungen von Zitationen und Start- sowie Endpositionen der Bestandteile kann ermittelt werden, welche Zitationspaare sich in welchen Textbestandteilen befinden und darauf basierend die entsprechenden Distanzen ermittelt werden. Das Ergebnis ist ein kartesisches Produkt der Zitationen mit entsprechenden Distanzangaben und wird als Matrix zurückgegeben. Die geplanten Distanz-Klassen und entsprechenden Gewichtungen der Distanzen sind in Tabelle 14 aufgeführt.

Im Vergleich zu den gewählten Werten für den DSI (Tabelle 3, Seite 12) und dem CPI (Tabelle 4) stammen die hier gewählten Werte nicht aus dem Intervall $(0, 1]$, sondern sind ganze Zahlen. Dies hat den Vorteil, dass gefundene Zitations-Paare, für die keine Angaben zu Distanzen zur Verfügung stehen, mit der Wertigkeit 1 ausgezeichnet werden können. Diese einfache Wertung ist kompatibel mit der Bibliografischen Kopplung sowie der Kozitation und ermöglichst somit eine leicht verständliche Kombination der Metriken. Zudem werden in dieser Arbeit andere Abstände der Distanz-Klassen festgelegt. Anstelle von Abständen zwischen Klassen, die sich jeweils verdoppeln,

Tabelle 14: Verwendete Distanz-Klassen und ihre Gewichtung

| Distanz-Klasse | Gewichtung | Gewichtung für Synthese |
|---|---|---|
| Gleiches Dokument | 1 | 0 |
| Gleicher Abschnitt | 2 | 1 |
| Gleicher Absatz | 3 | 2 |
| Gleicher Satz | 4 | 3 |
| Gleiche Markierung | 5 | 4 |

soll hier ein gleicher Abstand zwischen zwei aufeinanderfolgenden Klassen genutzt werden. Dies gewichtet einfache Vorkommen von Zitations-Paaren in einem Dokument stärker, die Distanzen fallen dafür weniger stark ins Gewicht. Trotz dieser Abschwächung müssten zwei Dokumente fünf Mal gemeinsam in Dokumenten auftreten, um genauso stark gewichtet zu werden wie zwei andere Dokumente, die einmalig in einer gemeinsamen Markierung auftreten.

Für die Implementierung wurde die Gewichtung der Distanzen angepasst. Dies wurde aus folgendem Grund entschieden. Sowohl die Kozitation als auch die Distanz der Zitationen beschreiben einen Zusammenhang, der sich auf referenzierte Publikationen auswirkt. Im optimalen Fall steht nach einer Extraktion von Metadaten ein komplettes Ergebnis zur Verfügung, das Daten zu beiden erwähnten Bibliometriken enthält. Die realen Daten der Extraktion sind lückenhaft. Durch fehlende Angaben zu Volltexten stehen für einen Teil der Datensätze Daten zur Berechnung der Kozitation, nicht aber für die Berechnung von Distanzen zur Verfügung. Dadurch entsteht ein Ungleichgewicht. Für Publikationen, zu denen keine Distanz-Daten bekannt sind, ergeben sich bei der Synthese der Bibliometriken kleinere Werte. Dadurch bedingt wurde die Gewichtung von Distanz-Klassen angepasst. Für Publikationen, deren Marker der Klasse *Dokument* entsprechen, wird eine Gewichtung von 0 zugewiesen. Dies entspricht dem Fall, dass Daten zu Distanzen fehlen. Die gemeinsame Zitierung von zwei Dokumenten wird dennoch durch die Kozitation berücksichtigt. Tabelle 14 zeigt eine Gegenüberstellung der Werte.

### 5.4.2 *Benutzerschnittstelle*

Für die Realisierung der Benutzerschnittstelle wurde das Play Framework gewählt (siehe Abschnitt 4.7.2). Das Framework nutzt die Model-View-Controller Architekturmuster, mit der Datenhaltung, die Benutzerschnittstelle und die Programmlogik getrennt werden. Zur

Anbindung an die bestehende Datenhaltung werden Modell-Klassen verwendet, mit denen ein Zugriff auf HBase und MySQL möglich ist.

Die grafische Benutzerschnittstelle besteht aus View-Komponenten, die verschiedene HTML5-Technologien verwenden. Mit HTML5 Boilerplate [HBP] wurde eine Vorlage für HTML5 und CSS3 verwendet. Durch diese wird Unterschieden der Darstellung verschiedener Browser begegnet. Die Gestaltung von Web-Oberflächen kann damit weitgehend einheitlich geschehen. Zusätzlich werden Komponenten des Bootstrap Projektes [Boot] verwendet. Damit können Vorlagen für HTML5-Elemente zur Darstellung verwendet werden. Durch die Verwendung der beiden HTML5-Projekte wurde der Entwicklungsaufwand für die grafische Benutzerschnittstelle verringert. Die Wiederverwendung existierender Softwarekomponenten stellt ein wichtiges Konzept in der Softwareentwicklung dar, das hier Verwendung finden konnte.

Die Literaturempfehlungen, deren Berechnung den Mittelpunkt der Präsentations-Phase darstellt, benötigen als Eingabe die Angabe einer Publikation. Damit entsteht ein Kaltstart-Problem, da Benutzern zunächst nicht bekannt ist, welche Datensätze verfügbar sind. Ein Zugriff auf die vorhandenen Volltexte soll nicht möglich sein, da die Bedingungen zur Verwendung, Veröffentlichung und der Lizensierung bei großen Datenmengen für einzelne Dokumente unklar sein können. Dies schließt unter Umständen auch eine Volltext-Suche ein. Für den ersten Zugriff von Benutzern wurde eine Suche nach den wichtigsten Metadaten realisiert. Dies sind der Titel einer Publikation und die Autoren, die durch ihre Nachnamen repräsentiert werden. Ergebnisse von Suchvorgängen werden als Listen entsprechender Publikationen angezeigt. Diese werden als Kombination von Titel, Autoren und Publikationsjahr angezeigt. Die Einträge sind durch Hyperlinks mit Webseiten zur Präsentation einzelner Publikationen verbunden.

Die Präsentation von Publikationen besteht aus einer Kombination aus Metadaten und Literaturempfehlungen. Abbildung 30 zeigt einen Screenshot zur Darstellung einer Publikation. Die folgende Gruppierung fasst die dargestellten Elemente zusammen:

A Der Titel der ausgewählten Publikation als primärer Identifikator

B Weitere Metadaten zur Publikation, bestehend aus Nachnamen der Autoren, Publikationsjahr, DOI, ID der Publikation im System

C Weitere Möglichkeiten zur Recherche. Es wurden die Projekte aus Abschnitt 2.3 genutzt. Diese stehen als Hyperlinks zur Suche auf den Seiten Google Scholar [Scholar], Mendeley [Mendeley], ResearchGate [RGate], Microsoft Academic Search [MSAS] und CiteSeerX [Cite] zur Verfügung.

D Beginn der Literaturempfehlungen

E Titel der folgenden Liste von Literaturempfehlungen

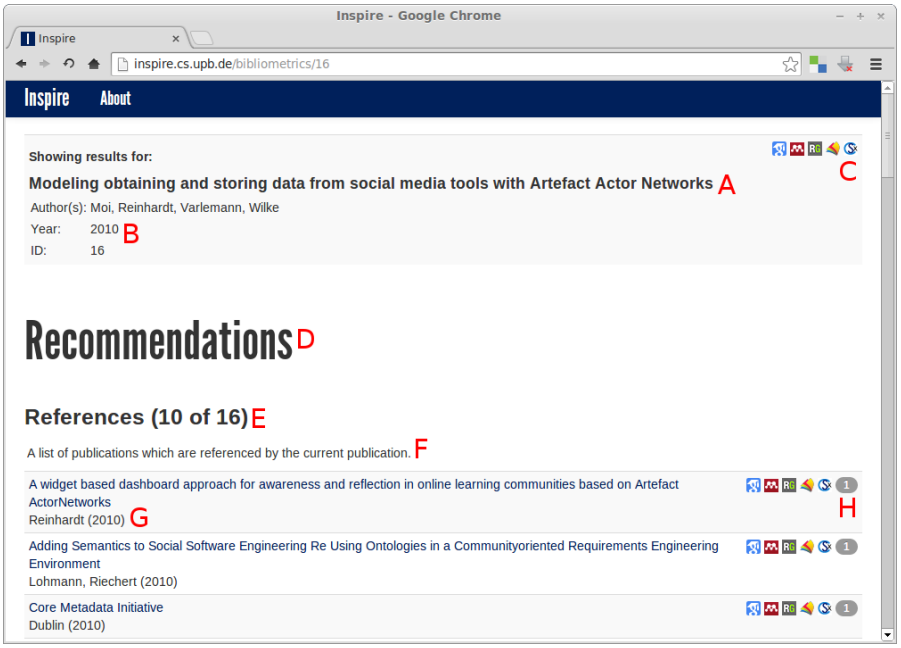



Abbildung 30: INSPIRE Benutzerschnittstelle

F Eine kurze Beschreibung zur Berechnung der folgenden Liste von Literaturempfehlungen

G Der Titel der empfohlenen Publikation. Durch einen Klick auf den Titel wird eine Seite mit Empfehlungen für den gewählten Titel geladen.

H Weitere Recherchemöglichkeiten. Diese entsprechen den Seiten aus (C). Außerdem wird die errechnete Punktezahl für den entsprechenden Listeneintrag angezeigt.

Die Punktezahl für einfache eingehende und ausgehende Referenzen ist jeweils 1. Punkte der Liste kombinierter Bibliometriken errechnen sich nach Formel 3. Die für die Bibliografische Kopplung und die Kozitation errechneten Punkte entsprechen der Anzahl der Publikationen, die gemeinsam verwendet werden bzw. die die aktuell betrachtete Publikation gemeinsam verwenden. Die Zusammensetzung dieser Werte entspricht den Erläuterungen aus Abschnitt 2.1. Um die Berechnung der Bibliometriken transparent zu gestalten, werden zusätzliche Ansichten verlinkt. Diese enthalten die Dokumente, auf denen die jeweiligen Berechnungen beruhen.

Die Berechnungen der Bibliometriken Referenzen, Zitiert-von, Bibliografischen Kopplung, Kozitation und Distanzen von Zitationen werden durch optimierte SQL-Anfragen durchgeführt. Die kombinierten Bibliometriken beruhen auf diesen und werden in Java berechnet. Eine Auflistung der genutzten Anfragen ist in Anhang A.2 aufgeführt.

# 6 ERGEBNISSE UND EVALUIERUNG

Dieses Kapitel enthält Ergebnisse zu den durchgeführten Berechnungen. Dabei werden in Abschnitt 6.1 Ergebnisse zur Extraktion von Metadaten präsentiert. Abschnitt 6.2 vergleicht die benötigten Ausführungszeiten der Phasen. Dabei wird die Phase der Präsentation gesondert in Abschnitt 6.3 behandelt. Das System wird durch die Durchführung einer Befragung evaluiert. Die Ergebnisse dazu sind in Abschnitt 6.4 aufgeführt.

## 6.1 ERGEBNISSE ZUR EXTRAKTION VON METADATEN

Dieser Abschnitt gibt einen Überblick über die Laufzeiten und Datenmengen der durchgeführten Extraktion von Metadaten zu Publikationen und ihren Referenzen. Die ermittelten Werte zur Anzahl von Elementen in XML-Dateien aus Abschnitt 3.1 stellen zusätzliche Ergebnisse der Extraktion dar. Sie werden an dieser Stelle nicht erneut aufgeführt.

Die Metadatenextraktion durch die Softwarepakete ParsCit und GROBID wird im entwickelten System mit der Hadoop MapReduce Implementierung verteilt ausgeführt. Die Ausführung der jeweiligen Extraktionssoftware findet lokal auf Cluster-Rechnern statt. Dabei werden PDF-Dateien (GROBID) und Volltexte (ParsCit) aus der verteilten Datenbank HBase gelesen und anschließend auf die lokale Festplatte geschrieben. Die jeweilige Extraktionssoftware nutzt die lokalen Dateien als Eingabe zur Extraktion. Dabei berechnete Ergebnisse sind XML-Dateien, die von den beiden Anwendungen auf die Festplatte geschrieben werden. Sie werden abschließend von der Festplatte gelesen und in HBase gespeichert.

Die im Folgenden aufgeführten Laufzeiten beziehen sich für die Ausführung von ParsCit auf die einzelnen Extraktionsvorgänge. Die Laufzeiten zur Ausführung von GROBID schließen zusätzlich den jeweiligen Schreib- und Lesevorgang der Eingabe- bzw. Ausgabedatei mit ein. Dies ist bedingt durch die unterschiedlichen Implementierungen der Anwendungen.

Die Werte der folgend aufgeführten Tabellen und Grafiken wurden durch die Verwendung von Hadoop-Zählern ermittelt. Laufzeiten bis zu zehn Sekunden wurden dabei in Sekunden-Schritten aufgerundet. Laufzeiten über zehn Sekunden wurden ebenfalls aufgerundet, in diesem Fall in 5-Sekunden-Schritten. Datengrößen bis 100 Kilobyte

Tabelle 15: Werte zur verteilten ParsCit Ausführung

| Zählvariable | Wert |
|---|---|
| Laufzeit | 18 h, 59 min, 16 s |
| Eingabe-Datensätze | 653.817 |
| Leere Eingabe | 2.413 |
| Keine Referenzen in Eingabe gefunden | 158.508 |
| Ausnahmefehler während der Ausführung | 4.803 |
| Leeres XML in der Ausgabe | 17 |
| Valide XML Länge der Ausgabe | 488.076 |

wurden in 10-Kilobyte-Schritten aufgerundet. Datengrößen über 100 Kilobyte wurden in 50-Kilobyte-Schritten aufgerundet. Für GROBID Metadaten zu Publikationen wurden zusätzliche Abstufungen vorgenommen, da diese Daten geringere Größen aufweisen. Es wurde zusätzlich bis zu einer Größe von 10 Kilobyte in Kilobyteschritten aufgerundet. Diese Rundungen beeinflussen die Ergebnisse von Berechnungen dieses Abschnitts für Laufzeiten und Datengrößen. Im Vergleich zu exakten Werten fallen die hier aufgeführten Werte geringfügig größer aus.

Durch die Verwendung des Hadoop-Frameworks entsteht ein Overhead. Dieser ist bedingt durch zusätzlichen Datenverkehr und Berechnungen zur Verteilung von Daten. Auch die Verwendung lokaler Hardware kann Verzögerungen von Berechnungen verursachen. Beispielsweise, wenn ein Rechner über mehrere Kerne verfügt, diese aber durch die Verwendung von nur einer Festplatte bei vielen Lese- und Schreiboperationen nicht voll ausgenutzt werden können. Abgesehen von der jeweiligen Gesamtlaufzeit werden die hier aufgeführten Ergebnisse davon nur in geringem Umfang beeinflusst.

### 6.1.1 *ParsCit*

Ein Teil der ermittelten Werte zur ParsCit Ausführung ist in Tabelle 15 aufgeführt. Ausgenommen sind Werte der einzelnen Ausführungszeiten und Datengrößen. Diese werden nachfolgend anhand von Grafiken beschrieben.

Die Laufzeit von knapp 19 Stunden entstand dadurch, dass für die hier aufgeführte Ausführung Mapper-Komponenten des Hadoop-Frameworks genutzt wurden. Dadurch findet keine Verteilung der Daten statt. Sind die Eingabedaten nicht gleichmäßig verteilt, kann es dazu kommen, dass auf einzelnen Rechnern im Cluster noch viele Daten verarbeitet werden müssen, während die Mehrzahl an Rechnern nicht

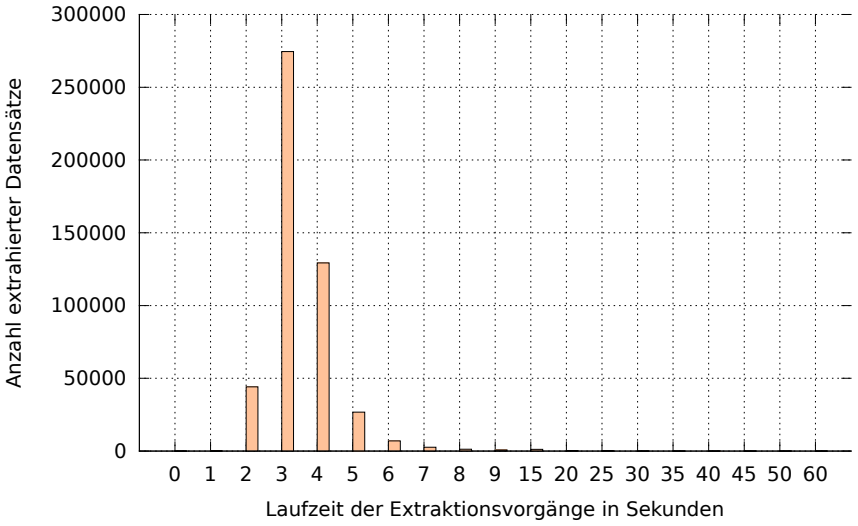



Abbildung 31: Anzahl der benötigten ParsCit-Ausführungszeiten
Verwendete Zeitintervalle zur Staffelung: $[0, 10]$, $(10, \infty)$

genutzt wird. Auf die Werte der einzelnen Messungen wirkt sich dies nicht aus.

Insgesamt konnten mit 488.076 XML Dateien rund 74% der Publikationen Metadaten extrahiert werden. Eine leere Eingabe deutet auf eine fehlerhafte Konvertierung vom PDF ins Textformat im vorangegangenen Schritt hin. In 158.508 Dateien wurde kein Referenzteil im Eingabetext gefunden werden. Dies ist beispielsweise der Fall, wenn im Eingabetext eine unbekannte Überschrift für den Referenzteil verwendet wird, etwa in einer nicht unterstützen Sprache. Die PDF Datei der Eingabe besteht in manchen Fällen aus Bilddateien, für die in der derzeitigen Implementierung keine Extraktionsmöglichkeit besteht. Es kann ebenfalls kein Referenzteil gefunden werden, wenn eine PDF-Eingabedatei keine Publikation, sondern beispielsweise eine Präsentation ist.

Während der Extraktion kam es in 4.803 Fällen zu einem Fehler während der Ausführung. Dies kann eintreten, wenn nicht zu verarbeitende Zeichenkodierungen auftreten. In 17 Fällen wurde die Extraktion abgeschlossen, es wurden jedoch keine Daten gefunden.

Die Ausführungszeiten der einzelnen Extraktionsvorgänge zeigt Abb. 31. Es sind 488.093 Zeiten aufgeführt, dies umschließt alle Ausführungen mit validem XML als Ausgabe. Das arithmetische Mittel der Ausführungszeiten ist 3,42 Sekunden. Der am häufigsten aufgezeichnete Wert ist 3 Sekunden; er tritt 274.540-mal auf.

Zusätzlich wurden die Größen der erstellten XML-Dateien festgehalten. Eine Übersicht zeigt Abb. 32. Der häufigste Wert ist 150 Kilobyte; er tritt 91.862-mal auf. Das arithmetische Mittel der Ausgabegrößen ist 88,21 Kilobyte. Die von ParsCit generierten XML-Dateien

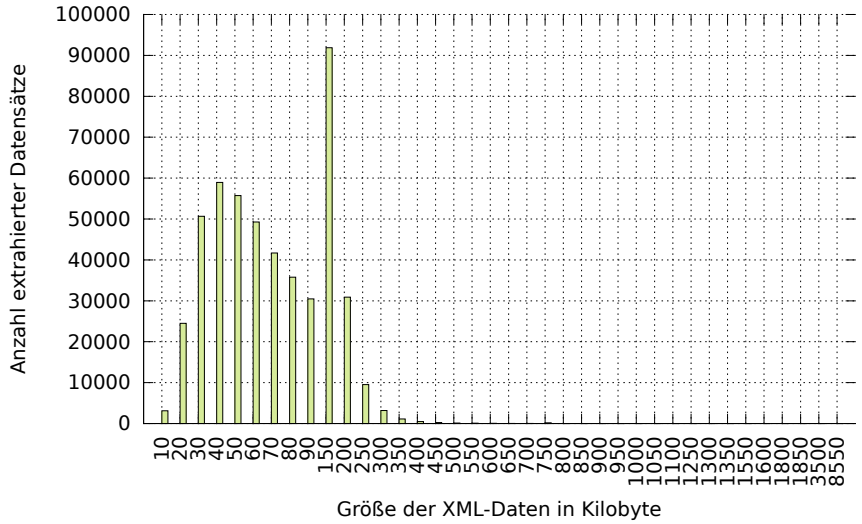



Abbildung 32: Anzahl der Größen von ParsCit XML Daten
Verwendete Intervalle zur Staffelung: $[0, 100], (100, \infty)$

Tabelle 16: Werte zur verteilten GROBID Ausführung

| Zählvariable | Wert |
|---|---|
| Laufzeit | 1 h, 19 min, 34 s |
| Eingabe-Datensätze | 170.677 |
| Ausnahmefehler während der Ausführung | 1 |
| Kopfdaten: Leeres XML in der Ausgabe | 340 |
| Kopfdaten: Valide XML Länge der Ausgabe | 170.336 |
| Referenzen: Leeres XML in der Ausgabe | 106.947 |
| Referenzen: Valide XML Länge der Ausgabe | 63.729 |

beinhalten neben den Metadaten auch den Volltext der Eingabe. Dessen Bestandteile werden mit ihrem jeweiligen Typ (z.B. Überschrift, Gleichung oder Abbildung) aufgeführt. In dieser Weise kommt die Größe der Dateien zustande.

### 6.1.2 *GROBID*

Die Extraktion mittels GROBID fand in mehreren Vorgängen statt. An dieser Stelle wird ein Extraktionsvorgang mit 170.677 verwendeten Eingabedokumenten präsentiert. Die wichtigsten Daten zur Extraktion mit GROBID zeigt Tabelle 16. In diesem Extraktionsvorgang konnten 99,80 % der Daten zu Publikationen und 37,34 % der Daten zu Referenzen extrahiert werden.

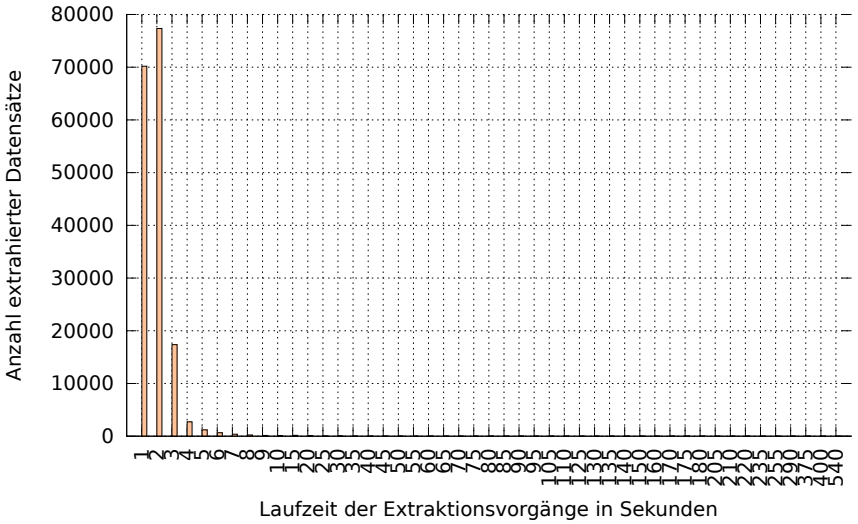



Abbildung 33: Anzahl der benötigten GROBID-Ausführungszeiten
Verwendete Zeitintervalle zur Staffelung: $[0, 10], (10, \infty)$

Die Ausführungszeiten zu den Extraktionsvorgängen zeigt Abb. 33. Es sind Zeiten von 170.676 Extraktionen aufgeführt. Das arithmetische Mittel der Zeiten ist 1,85 Sekunden. Die drei häufigsten Zeiten sind 2 Sekunden (Anzahl der Messungen: 77.345), 1 Sekunde (70.193) und 3 Sekunden (17.370).

Die Größen der XML-Daten zu den Publikationen zeigt Abb. 34. Es sind 170.336 Datensätze aufgeführt. Das arithmetische Mittel der Datengrößen ist 5,61 Kilobyte. Die drei häufigsten Werte sind 4 Kilobyte (39.078), 5 Kilobyte (32.221) und 3 Kilobyte (30.014).

Die Größen der XML-Daten zu den Referenzen der Publikationen zeigt Abb. 35. Es sind 63.729 Datensätze aufgeführt. Das arithmetische Mittel der Datengrößen ist 21,44 Kilobyte. Die drei häufigsten Werte sind 10 Kilobyte (22.766), 20 Kilobyte (19.667) und 30 Kilobyte (14.116).

Im Vergleich sind die durch ParsCit extrahierten Daten mit einer durchschnittlichen Größe von rund 88 KB größer als die GROBID Daten mit durchschnittlich rund 6 KB und 21 KB. Die ParsCit Daten enthalten neben Metadaten weitere Angaben zum Volltext.

Die durchschnittliche Ausführungszeit von ParsCit ist mit 3,42 Sekunden größer als die von GROBID mit 1,85 Sekunden.

Die Summe der einzelnen ParsCit Ausführungszeiten für 488.093 Berechnungen beträgt 1.670.160 Sekunden. Dies entspricht 19 Tagen, 7 Stunden, 56 Minuten und 0 Sekunden. Durch die parallele Verarbeitung im Rechnercluster wurden dafür 18 Stunden, 59 Minuten und 16 Sekunden benötigt.

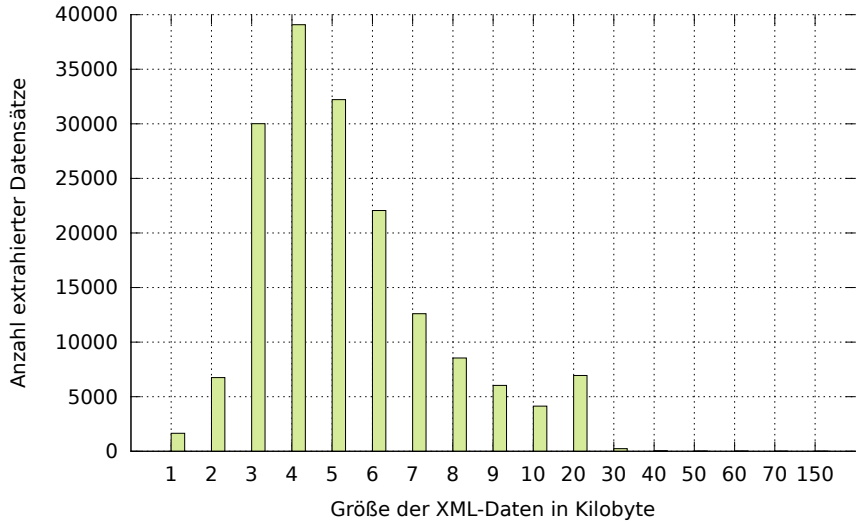



Abbildung 34: Anzahl der Größen von GROBID XML Daten zu Publikationen. Verwendete Intervalle zur Staffelung: $[0, 10]$, $(10, 100]$, $(100, \infty)$

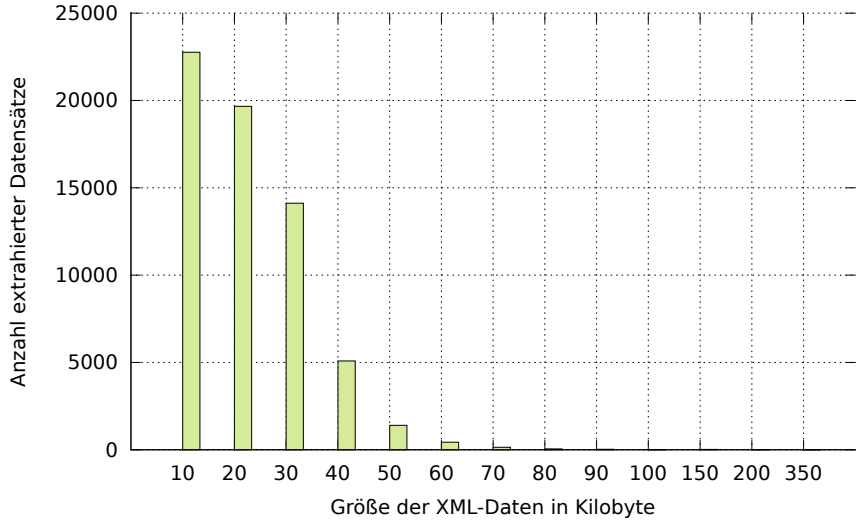



Abbildung 35: Anzahl der Größen von GROBID XML Daten zu Referenzen Verwendete Intervalle zur Staffelung: $[0, 100]$, $(100, \infty)$

Die Summe der einzelnen GROBID Ausführungszeiten für 170.676 Berechnungen beträgt 316.550 Sekunden. Diese Summe entspricht 3 Tagen, 15 Stunden, 55 Minuten und 50 Sekunden. Durch die parallele Verarbeitung nahm die Berechnung 1 Stunde, 19 Minuten und 34 Sekunden in Anspruch.

## 6.2 GEGENÜBERSTELLUNG DER BENÖTIGTEN AUSFÜHRUNGSZEITEN

Tabelle 17: Ausführungszeiten der Phasen

| Phase | Zeit | Je Eingabe | Prozentual |
|---|---|---|---|
| Upload | 7 min, 35 s | 49,07 ms | 7,99 % |
| Konvertierung | 6 min, 56 s | 44,86 ms | 7,30 % |
| Extraktion: ParsCit | 11 min, 23 s | 73,65 ms | 11,99 % |
| Extraktion: GROBID | 32 min, 24 s | 209,64 ms | 34,12 % |
| Integration | 0 min, 21 s | 2,26 ms | 0,37 % |
| Deduplikation | 36 min, 19 s | 234,98 ms | 38,24 % |
| Insgesamt | 94 min, 58 s | 614,47 ms | 100,00 % |

Für eine Gegenüberstellung der benötigten Ausführungszeiten wurde der Dokumentenkorpus der ED-MEDIA[1] genutzt. Dieser umfasst 9.273 Publikationen. Die Ausführungszeiten sind in Tabelle 17 aufgeführt. Für die Phasen sind die gemessene Ausführungszeit, die durchschnittliche Ausführungszeit der 9.273 Dokumente und der prozentuale Anteil aufgelistet.

Zusätzlich zu den fünf Phasen sind die Zeiten für den Upload angegeben. Die benötigten Zeiten für die Extraktion sind getrennt nach den beiden Extraktionsanwendungen ParsCit und GROBID aufgeführt. In der letzten Phase, der Präsentation, wird nicht jedes Dokument durchlaufen. Daher werden die Ausführungszeiten in Abschnitt 6.3 gesondert behandelt.

Während des Uploads werden neben der Übertragung und Verteilung von Daten keine weiteren Berechnungen durchgeführt. Der Anteil der Ausführungszeit von 7,99 % ist darauf zurückzuführen, dass die Daten aus einem nicht-verteilten Dateisystem gelesen werden und für den Lesevorgang keine Parallelisierung stattfindet.

Die Konvertierungs-Phase besteht aus der Extraktion von Text aus PDF-Dateien. In der verwendeten Implementierung werden Mapper-

1 ED-MEDIA: World Conference on Educational Multimedia, Hypermedia and Telecommunications

Instanzen verwendet und dadurch die Datenlokalität genutzt. Damit findet keine zusätzliche Verteilung der Daten statt. Dateien werden auf den Clusterrechnern konvertiert, auf denen sie gespeichert sind. Da die Pfade der verwendeten Eingabedokumente sich ähneln, und den verwendeten HBase-Schlüsseln entsprechen, wurden die Daten auf zwei Clusterrechnern verteilt. Diese Verteilung geschieht automatisch. Bei umfangreicheren Berechnungen würde durch eine stärkere Verteilung zusätzliche Clusterrechner genutzt. Bei der Betrachtung dieser Ausführungszeit muss also beachtet werden, dass die Konvertierung lediglich auf zwei Rechnern ausgeführt wurde.

Die Extraktion von Metadaten mit der Anwendung GROBID benötigt mit 34,12 % die meiste Zeit. Damit entsteht für die Extraktions-Phase insgesamt ein Ausführungszeitanteil von 46,11 %. Die Extraktion wurde durch Reducer-Instanzen ausgeführt und im gesamten Cluster verteilt.

Die Integrations-Phase nimmt mit 0,37 % einen geringen Teil der benötigten Zeit ein. In dieser Phase werden ausschließlich JAVA-Klassen genutzt, die im Rahmen dieser Arbeit implementiert wurden. Die vorgenommenen Berechnungen sind weniger komplex. Die kurze Laufzeit ist auch darauf zurückzuführen, dass SAX-Parser zur Analyse der zu integrierenden XML-Daten genutzt werden.

Mit 38,24 % nimmt die Deduplikation mehr als ein Drittel der Zeit in Anspruch. In dieser Phase werden im Cluster verteilte Daten in einer MySQL Instanz vereinigt. Da für jedes Eingabedokument Vergleichsoperationen zwischen dem Titel des Dokuments und (im Falle keiner Übereinstimmung) allen bereits existierenden Titeln durchgeführt werden, fällt die Laufzeit entsprechend hoch aus.

Insgesamt wird für jedes Dokument im Durchschnitt eine Laufzeit von 614,47 ms benötigt.

## 6.3 PERFORMANZ DER BENUTZERSCHNITTSTELLE

Ein Ziel für das entwickelte System ist die performante Echtzeitberechnung von Literaturempfehlungen. Im Vorfeld wurde dafür im Abschnitt 3.3 untersucht, welche Möglichkeit der Datenhaltung sich für die Echtzeitberechnung von Bibliometriken anbietet. Die relationale Datenbank MySQL erzielte die besten Ergebnisse. Basierend auf diesem Resultat wurde ein ER Modell für die Datenhaltung zur anschließenden Berechnung der Bibliometriken konzipiert. Die Berechnungen werden dabei abschließend durch optimierte SQL-Anfragen in Echtzeit durchgeführt. Die Ausgabe übernimmt das Play Framework.

In diesem Abschnitt soll stichprobenartig geprüft werden, ob das Ziel der performanten Berechnung erreicht wurde. Dazu wird das für die

Evaluierung erstellte System genutzt. Es werden Anfragen für die Berechnung von Literaturempfehlungen für zehn zufällig gewählte IDs gesendet und die Zeit gemessen, die zwischen der Anfrage und den empfangenen Ergebnissen vergeht. Die Anfragen wurden nicht parallel, aber direkt aufeinander folgend gestellt. Die Anfragen[2] wurden zur Vergleichbarkeit außerhalb des Universitätsnetzes gestellt.

Die durchschnittlich vergangene Zeit ist 250 ms. Dabei wurden durchschnittlich Daten der Größe 54.012 Byte übertragen[3]. Unter Betrachtung der Messungen wurde das Ziel einer performanten Echtzeitberechnung der Bibliometriken erreicht.

## 6.4 EVALUIERUNG

Zur Evaluierung des entwickelten Systems viel die Wahl auf eine quantitative Methode. Diese wurde durch eine Befragung in Form eines Online-Fragebogens umgesetzt. Dabei wurden Items verwendet, die in die folgenden Gruppen unterteilt werden können:

1. Angaben zur eigenen Person
2. Einschätzung der eigenen Expertise zu der/den gewählten Publikation(en)
3. Nutzen des Systems
4. Eignung der Listeneinträge als Literaturvorschläge
5. Zukünftige Nutzung des Systems

Die verwendeten Items können in Anhang A.1 eingesehen werden. Mit Wahl der Methode soll der pragmatische Stil dieser Arbeit fortgeführt und Einschätzungen zum Nutzen des Systems von dessen Benutzern ermittelt werden.

Die Grundlage zur Befragung stellen die in dieser Arbeit entwickelten Komponenten, mit denen ein Dokumentennetzwerk erstellt wurde, das für die Berechnung der Literaturempfehlungen genutzt wird. Für diese wurden wissenschaftliche Publikationen aus der Informatik als Datenbasis zur Generierung des Dokumentennetzwerks gewählt. Die erstellte Datenbank zur Umfrage umfasst 761.432 Datensätze zu Publikationen und 970.733 Datensätze zu Referenzierungen.

Die Befragung fand vom 12.04. bis zum 22.04.2013 statt. Potenzielle Probanden wurden durch die Verteilung von E-Mails innerhalb von drei Fachgruppen der Universität informiert. Zusätzlich wurde auf der Webseite der Fachschaft Mathematik/Informatik der Universität

2 Zur Messung wurde folgender Befehl verwendet:
*time wget http://inspire.cs.upb.de/bibliometrics/ID*

3 Zur Einordnung: Eine Anfrage zur Adresse *http://www.google.de/* dauerte 460 ms, dabei wurden 12.826 Byte übertragen. Dies nur zur Einschätzung und nicht als Vergleich – Routing-Daten wurden z.B. nicht betrachtet.

Paderborn auf die Umfrage hingewiesen [FSMI]. Außerdem wurden 15 weitere Studierende der Informatik durch E-Mails informiert.

In der Befragung wurden Items genutzt, bei denen eine Angabe zwischen 1 und 5 möglich war. Dabei wurde eine Zuordnung von 1: *Stimme überhaupt nicht zu* und 5: *Stimme vollständig zu* verwendet. Im Folgenden wird eine Zustimmung mit dem Zeichen „+“ und keine Zustimmung mit „-“ zur besseren Lesbarkeit gekennzeichnet.

*1. Angaben zur eigenen Person*

Es nahmen 11 Probanden an der Befragung Teil. Davon waren 81,82 % männlich und 18,18 % weiblich. Das Alter der Probanden lag zwischen 23 und 62 Jahren. 63,64 % der Probanden gaben an, an der Universität Paderborn zu arbeiten oder zu studieren. Für die Zugehörigkeit zu einer Hochschule wurden zusätzlich folgende Angaben gemacht:

18,18 % Ich bin Bachelor-Student an einer Hochschule

27,27 % Ich bin Master-Student an einer Hochschule

27,27 % Ich bin Mitarbeiter an einer Hochschule

27,27 % Sonstiges

Zum Fach-/Studiengebiet wurde Folgendes angegeben:

63,64 % Mein Fach-/Studiengebiet ist Informatik

00,00 % Mein Fach-/Studiengebiet ist Mathematik

36,36 % Sonstiges

Es ist anzumerken, dass durch die geringe Anzahl der Teilnehmer von den Ergebnissen dieser Evaluation nicht auf die Allgemeinheit zurück geschlossen werden kann. Für einen Eindruck der Bewertungen werden die vorliegenden Ergebnisse an dieser Stelle trotz dessen ausgewertet. Der Hauptteil der Probanden setzt sich aus Personen des Gebiets Informatik zusammen. Hauptsächlich haben Studierende an der Befragung teilgenommen, aber auch Mitarbeiter nahmen teil.

*2. Einschätzung der eigenen Expertise zu der/den gewählten Publikation(en)*

Die Einschätzung der eigenen Expertise zu der/den gewählten Publikation(en) wurde über die folgenden drei Items angegeben:

> Ich habe mich bereits intensiv mit den Inhalten der von mir gewählten Publikation auseinandergesetzt

| 1 (−) | 2 | 3 | 4 | 5 (+) |
|---|---|---|---|---|
| 18,18 % | 9,09 % | 9,09 % | 18,18 % | 45,45 % |

Ich habe bereits eine akademische Arbeit im Bereich der betrachteten Publikation verfasst (z.B. Bachelorarbeit)

| Ja | Unsicher | Nein |
|---|---|---|
| 54,55 % | 9,09 % | 36,36 % |

Ich habe bereits an einer wissenschaftlichen Publikation aus dem Bereich der betrachteten Publikation mitgewirkt

| Ja | Unsicher | Nein |
|---|---|---|
| 36,36 % | 9,09 % | 54,55 % |

Etwa 45 % der Teilnehmer haben sich intensiv mit den betrachteten Publikationen beschäftigt. Etwa die Hälfte hat eine akademische Arbeit im gewählten Bereich verfasst und ca. ein Drittel der Probanden war an einer wissenschaftlichen Publikation des Bereichs beteiligt. Diese Selbsteinschätzung deutet darauf hin, dass die Befragten Fachwissen vorweisen können, das zur Einschätzung der Literaturempfehlungen genutzt werden kann.

*3. Nutzen des Systems*

Eine Einschätzung zum Nutzen des Systems konnte über die zwei folgenden Items angegeben werden.

Die zur gewählten Publikation vorgeschlagenen Dokumente eigenen sich gut als Zusatzliteratur

| 1 (−) | 2 | 3 | 4 | 5 (+) |
|---|---|---|---|---|
| 00,00 % | 9,09 % | 36,36 % | 36,36 % | 18,18 % |

Durch die Nutzung Webseite konnte ich nützlich Dokumente finden, die mir bisher unbekannt waren

| 1 (−) | 2 | 3 | 4 | 5 (+) |
|---|---|---|---|---|
| 00,00 % | 18,18 % | 18,18 % | 27,27 % | 36,36 % |

Mehr als die Hälfte der Befragten würde einer Eignung der Empfehlungen als zusätzliche Literatur eher zustimmen. Auch unbekannte, nützliche Dokumente konnten von mehr als der Hälfte der Teilnehmer gefunden werden.

*4. Eignung der Listeneinträge als Literaturvorschläge*

Die Ergebnisse zur Eignung der Literaturvorschläge, die über die Verwendung verschiedener Bibliometriken ermittelt wurden, sind in Abb. 36 gegenübergestellt.

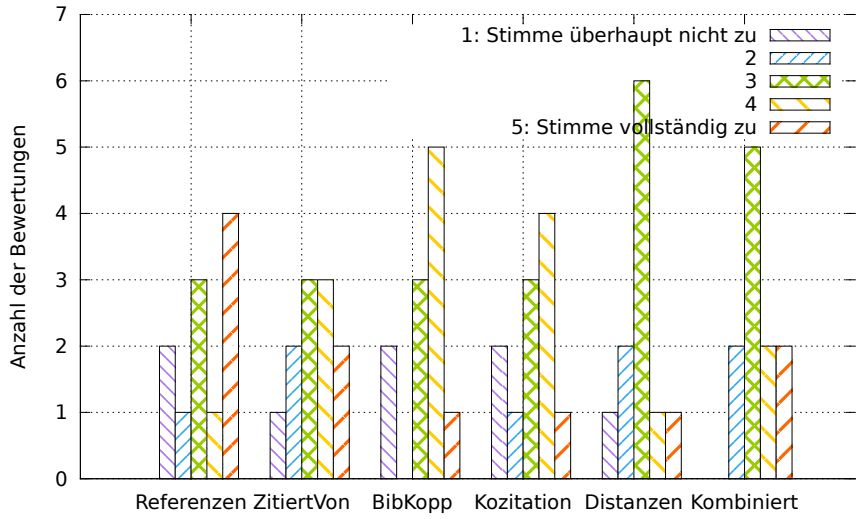



Abbildung 36: Bewertung der Bibliometriken über Zustimmung zu "Die Publikationen in der Liste eignen sich gut als Vorschläge für zusätzliche Literatur"

Es kann nicht festgestellt werden, dass sich eine der Alternativen besonders von den übrigen abhebt. Ein Ergebnis ist trotzdem interessant. Zur Einschätzung, ob sich die kombinierten Bibliometriken gut als Vorschläge eignen, wurde in keinem Fall überhaupt nicht zugestimmt. Dies kann möglicherweise durch den folgenden Zusammenhang erklärt werden. Eine Publikation, die als Volltext vorliegt, hat i.d.R mehrere Einträge für die Listen *Referenzen* und *Bibliografische Kopplung*. Publikationen, die nicht als Referenzen vorliegen, weisen dagegen Einträge in den Listen *Zitiert von*, *Kozitation* und *Distanzen* auf. In der Liste *Kombinierte Bibliometriken* werden Einträge aus allen Listen vereint. Daher ist diese Liste, im Gegensatz zu den anderen, selten leer. Dies könnte die Ursache darstellen, warum die Ergebnisse wie dargestellt bewertet wurden.

*5. Zukünftige Nutzung des Systems*

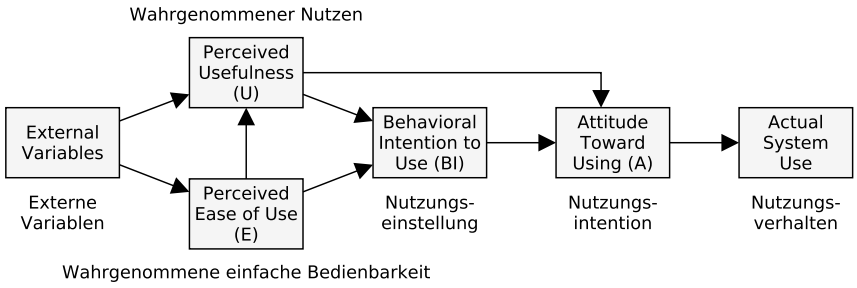



Abbildung 37: Technology Acceptance Model

Tabelle 18: Werte zum Technology Acceptance Model

| Konstrukt | Item | Mittel* | $\sigma$ | Minimum | Maximum | $\alpha$ |
|---|---|---|---|---|---|---|
| EOU | EOU1 | 3,91 | 1,38 | 2 | 5 | 0,69 |
| | EOU2 | 3,55 | 1,13 | 1 | 5 | |
| | EOU3 | 4,09 | 1,04 | 2 | 5 | |
| | EOU4 | 4,00 | 0,89 | 3 | 5 | |
| U | U1 | 3,91 | 1,14 | 2 | 5 | 0,97 |
| | U2 | 3,55 | 1,29 | 1 | 5 | |
| | U3 | 3,73 | 1,19 | 1 | 5 | |
| | U4 | 3,82 | 1,25 | 1 | 5 | |
| A | A1 | 4,18 | 1,33 | 1 | 5 | 0,79 |
| | A2$^{\pm}$ | 1,45 | 0,69 | 1 | 3 | |
| | A3 | 3,27 | 0,90 | 2 | 5 | |
| | A4 | 3,36 | 1,03 | 2 | 5 | |
| | A5 | 3,55 | 0,82 | 2 | 5 | |
| BI | BI1 | 3,82 | 1,17 | 2 | 5 | 0,95 |
| | BI2 | 3,64 | 1,50 | 1 | 5 | |
| | BI3 | 3,36 | 1,43 | 1 | 5 | |

* Arithmetisches Mittel — $\sigma$ Standardabweichung

$\alpha$ Cronbachs Alpha — $^{\pm}$ Polung bei Berechnung von $\alpha$ geändert

Um einen Indikator für eine zukünftige Nutzung des entwickelten Systems zu erhalten, wurden der Befragung Items zum Technology Acceptance Model (TAM) [DBW89] verwendet. Mit dem TAM wird der Einfluss zwischen verschiedenen Konstrukten modelliert und damit Bedingungen für eine Nutzung formuliert. Abb. 37 zeigt den Aufbau des TAM. Die deutschen Übersetzungen wurden aus [Joco9] übernommen. Wie in der Abbildung zu erkennen, hängt der wahrgenommene Nutzen (U) von der wahrgenommenen einfachen Bedienbarkeit (E) ab. Beide Konstrukte wirken sich auf die Nutzungseinstellung (BI) aus. Die Nutzungseinstellung und der wahrgenommene Nutzen üben wiederum Einfluss auf die Nutzungsintention (A) und damit indirekt auf das Nutzungsverhalten aus.

In der Befragung zu dieser Arbeit wurden 16 Items zu den vier Konstrukten EOU[4], U, BI und A verwendet. Tabelle 18 zeigt eine Auswertung der über die Befragung ermittelten Werte. Dabei wurde Cronbachs Alpha ermittelt, um die interne Konsistenz der verwendeten

4 Die Bezeichnung EOU wird für externe Variablen verwendet, mit denen E ermittelt wird

Items zu überprüfen. Ein Wert ab 0,7 kann dabei als akzeptabel angesehen werden. Der Alpha-Wert für EOU ist grenzwertig; für die übrigen Alpha-Werte kann davon ausgegangen werden, dass die interne Konsistenz der Item-Gruppen gegeben ist.

Zu den vier Konstrukten wurde das arithmetische Mittel unter Einbeziehung aller abgefragten Items zu den Konstrukten ermittelt. Die Ergebnisse sind in Tabelle 19 aufgeführt.

Tabelle 19: Arithmetisches Mittel der TAM Konstrukte

| EOU | U | BI | A |
|---|---|---|---|
| 3,89 | 3,75 | 3,61 | 3,58 |

Die Werte zeigen, dass sich das System eher einfach bedienen lässt (EOU) und ein auch Nutzen (U) wahrgenommen wurde. EOU und U haben Einfluss auf die Nutzungseinstellung (BI), deren Wert entsprechend ausfällt. Auch die darauf basierende Nutzungsintention fällt eher zustimmend aus. Insgesamt kann davon ausgegangen werden, dass das System akzeptiert und genutzt werden würde.

# 7 DISKUSSION UND AUSBLICK

In diesem Kapitel werden die Resultate der Arbeit diskutiert. Dazu werden die Ergebnisse der Arbeit in Abschnitt 7.1 zunächst resümiert und diskutiert. Überlegungen zu möglichen Erweiterungen in Abschnitt 7.2 bilden den Abschluss dieser Arbeit.

## 7.1 ZUSAMMENFASSUNG UND DISKUSSION DER ERGEBNISSE

Mit dieser Arbeit wurde ein System zur Berechnung von Literaturempfehlungen realisiert. In diesem Abschnitt werden die erarbeiteten Ergebnisse zusammengefasst und diskutiert.

Als erster Schritt wurde in Abschnitt 3.1 untersucht, welche Metadaten nach einer Extraktion durch die Anwendungen ParsCit und GROBID zur Verfügung stehen. Dazu wurden 571.897 PDF-Dateien als Eingabe verwendet und insgesamt 954.569 extrahierte XML-Datensätze untersucht. Diese Datensätze bestehen aus einzelnen Metadaten wie Titel oder den Namen von Autoren, die über XML-Elemente zugreifbar sind. Ziel der Analyse war die Feststellung, welche XML-Elemente extrahiert werden und in welcher Anzahl sie dadurch verfügbar sind. Die durchgeführte Analyse ist rein quantitativer Art. Von einer umfassenden qualitativen Untersuchung wurde abgesehen, da die Ergebnisse der Extraktion von trainierten Modell-Dateien abhängen. Eine Erweiterung der Modelle führt zu veränderten Ausgaben, wodurch ursprüngliche Ergebnisse ihre Aussagekraft verlieren. Durch die Ergebnisse der qualitativen Untersuchung von 954.569 Datensätzen wird eine Einschätzung der verfügbaren Daten ermöglicht. Die Ergebnisse sind in 6 Tabellen ab Seite 26 aufgeführt. Zusätzlich wurde eine Analyse der Anzahl extrahierter Referenzen durchgeführt. Für diese Untersuchung wurde die Anzahl der Referenzen von 50 zufällig gewählten Publikationen manuell ermittelt und mit den extrahierten Datensätzen abgeglichen. ParsCit fand in 27 der 50 Fälle exakt so viele Referenzen, wie in den Eingabedokumenten vorhanden sind. Im Fall von GROBID waren dies nur 5 der 50 Fälle. Dies lässt eine eher geringe Qualität der Extraktion vermuten. Die Ergebnisse sind in Tabelle 11 auf Seite 31 aufgeführt.

Eine weitere Untersuchung (Abschnitt 3.3) hatte die Auswahl einer Datenhaltungslösung zum Ziel. Dafür wurde ein Benchmark konzipiert, mit dem Berechnungszeiten von Bibliometriken bei wachsender Netzwerkgröße untersucht wurden. Zur Analyse standen dabei

MySQL, Neo4j und eine Titan/HBase Kombination. Es konnte festgestellt werden, dass die Berechnungszeiten bis zu einer Netzwerkgröße von 20 Mio. Knoten nicht maßgeblich ansteigen. Anschließend wurden die absoluten Zeiten zur Berechnung der Bibliometriken gemessen und verglichen. MySQL lieferte hier die mit Abstand besten Ergebnisse. Dies ist vermutlich auf interne Optimierungen der relationalen Datenbank zurückzuführen. Es wird weiter vermutet, dass die MySQL-Berechnungszeiten bei viel größeren Netzwerken die Zeiten einer verteilten, skalierbaren Lösung überschreiten (vgl. [Geo11, S. 5]. Dies ist für die Dokumentennetzwerke dieser Arbeit jedoch nicht von Relevanz.

Zur Konzeption des Gesamtablaufs wurde untersucht, welche grundsätzlichen Typen von Berechnungen unterschieden werden können. Das Ergebnis ist ein 5-Phasen-Modell, das als Grundgerüst zum weiteren Design und der Implementierung verwendet wurde. Eine Übersicht zum Modell zeit Abb. 11 auf Seite 44. Die resultierenden Phasen sind Konvertierung, Extraktion, Integration, Deduplikation und Präsentation. Die Phasen sind in sich abgeschlossen, sodass die zugehörigen Berechnungen durch unterschiedliche Frameworks und Implementierungen realisierbar sind.

Der Entwurf zur verteilten Berechnung (Abschnitt 4.2) nutzt die unterschiedlichen Eigenschaften von Hadoop Mapper- und Reducer-Komponenten. Durch Einbringung von Domänenwissen kann die verteilte Berechnung beschleunigt werden. Dies ist das Wissen darüber, wie die zu verarbeitenden Dateien im hierarchischen Dateisystem verteilt sind. Diese Informationen über die verwendeten Dateipfade werden im verwendeten System als HBase-Schlüssel genutzt. Damit kann ein Intervall von zu berechnenden Datensätzen angegeben werden. Ein Beispielintervall stellt [/upload/2011, /upload/2013/] dar. Bei umfangreichen Intervallen sollten Mapper-Komponenten zur Berechnung gewählt werden. Dadurch wird von der Datenlokalität Gebrauch gemacht und es entsteht weniger Netzwerkverkehr. Bei kleinen Intervallen sollten Reducer-Komponenten gewählt werden. In diesem Fall werden Eingabedateien zunächst im Rechnercluster verteilt und können anschließend parallel verarbeitet werden. Von dieser Unterscheidung konnte bei den Berechnungen dieser Arbeit umfassend Gebrauch gemacht werden.

Die Behandlung der mangelnden Datenqualität extrahierter Daten findet in der Integrationsphase statt. Durch Normalisierungen der Zeichenketten von Metadaten werden mögliche Unterschiede und Fehler aus der Extraktion vermindert. Dabei werden unter anderem Umlaute ersetzt und Satz- sowie Sonderzeichen entfernt. Die normalisierten Metadaten werden anschließend integriert. Dies verbessert die Datenqualität dadurch, dass Einträge gegenseitig ergänzt werden. Beispielsweise könnte ein Matching von normalisierten Titeln zweier

Datensätze auftreten. Angenommen in diesem Beispiel fehlt in genau einem Datensatz die Angabe zum Publikationsjahr und nur im anderen Datensatz die Angabe einer DOI, würde der resultierende, integrierte Datensatz beide Angaben enthalten.

Neben den eingangs beschriebenen Ergebnissen zu Häufigkeiten von XML-Elementen der Metadaten wurden weitere Daten zur Extraktion mittels ParsCit und GROBID untersucht. Abschnitt 6.1 enthält eine Übersicht der benötigten Ausführungszeiten und der Größe extrahierter Datensätze. Diese Daten können für mögliche Folgeprojekte genutzt werden.

Bei der Entwicklung der Komponenten dieser Arbeit wurde auf eine mögliche Wiederverwendbarkeit Wert gelegt. Dabei konnte die ParsCit-Komponente bereits in der zeitgleich entstehenden Masterarbeit von Nicolas Schelp [Schck] Verwendung finden. Im Rahmen dieser Arbeit wurde ein zusätzlicher SAX-Parser entwickelt, mit dem ParsCit XML-Dateien analysiert werden und Elemente, die keinen Text darstellen, entfernt werden. Dies sind zum Beispiel Angaben zu Grafiken oder Formeln. Diese werden entfernt, da sie bei Clusterings von Volltexten zur Feststellung der Ähnlichkeit von Dokumenten keinen Mehrwert darstellen und die Ergebnisse sogar verschlechtern können. Im Gegenzug konnten in dieser Arbeit Publikationen aus dem Dokumentenkorpus genutzt werden, der im Rahmen der Arbeiten von Nicolas Schelp und Tobias Varlemann [Var13] aufgebaut wurde. Daneben wurde im Rahmen ihrer Arbeiten die genutzte Hardware auf- und ausgebaut. Wie bereits erwähnt, konnten in dieser Arbeit auch Komponenten zur Konvertierung von PDF-Dateien von Tobias Varlemann wiederverwendet werden.

Der Benchmark der möglichen Lösungen zur Datenhaltung aus Abschnitt 3.3 war eins der ersten Resultate dieser Arbeit. Mit der Fertigstellung der Implementierung konnte abschließend getestet werden, ob sich die Bibliometriken im laufenden Betrieb des entstandenen Systems performant berechnen lassen. Mit dem Test der Performanz der Benutzerschnittstelle in Abschnitt 6.3 konnte dies bestätigt werden. Die Evaluierung zur Akzeptanz und dem Nutzen des Systems (Abschnitt 6.4) ergab zufriedenstellende Umfrageergebnisse.

## 7.2 MÖGLICHE ERWEITERUNGEN

Im Laufe dieser Arbeit stellte sich heraus, dass die größte Herausforderung der behandelten Bereiche in der Datenqualität liegt. Ein Zugriff auf vollständige, korrekte Metadaten würde die Qualität der Datenbasis enorm verbessern und damit auch die Ergebnisse der sich darauf stützenden Berechnungen. Verbesserungen innerhalb von An-

wendungen der Extraktion von Metadaten stellen aus Sicht des Autors die wertvollste Weiterentwicklung dar.

Auch die Einbeziehung weiterer Metadaten kann zum Ausbau dieses Systems verwendet werden. So wird z.B. die Veröffentlichung von Publikationen in der gleichen Zeitschrift für die Distanz von Zitationen genutzt [GB09]. Für Publikationen gleicher Autoren besteht die Möglichkeit der Verwendung für Literaturempfehlungen [knowAAN].

Auch könnte eine weitere, umfangreichere Befragung zur Einschätzung des Nutzwertes der verschiedenen Bibliometriken stattfinden. Mit Ergebnissen einer solchen Umfrage bestünde die Möglichkeit, eine besser abgestimmte Kombination der Bibliometriken zu entwickeln. Eine Umsetzung wäre durch Anpassung der Gewichtungen aus Formel 3 auf Seite 63 möglich.

Neben der Verwendung von Bibliometriken und Metadaten für Literaturempfehlungen können dafür auch Varianten von Volltextanalysen [Var13, Schck] verwendet werden. Ebenso ist eine Kombination in einem hybriden Ansatz denkbar.

Eine sehr naheliegende und einfach zu realisierende Erweiterung stellt eine Vergrößerung des verwendeten Dokumentenkorpus dar. Diese Möglichkeit profitiert von der steigenden Verbreitung des Open Access.

# A

# ANHANG

## A.1 BEFRAGUNG ZUR EVALUIERUNG

Vor der Abfrage der Items wurden den Probanden die zwei folgenden Texte zur Einführung angezeigt:

> Sehr geehrter Besucher,
> in meiner Abschlussarbeit beschäftige ich mich mit Literaturvorschlägen für wissenschaftliche Publikationen. Dabei werden die Vorschläge auf Grundlage einer ausgewählten wissenschaftlichen Publikation berechnet.
> Dieser Fragebogen dient zur Evaluierung der Abschlussarbeit. Die Beantwortung des Fragebogens nimmt etwa 10 Minuten in Anspruch. Zur Beantwortung des Fragebogens verschaffen Sie sich zunächst einen Eindruck über die Literaturvorschläge auf der Webseite. Anschließend beantworten Sie Fragen zur Nutzung der Webseite.
> Die Webseite finden sie unter der Adresse inspire.cs.upb.de
> Vielen Dank für Ihre Unterstützung! Adrian Wilke

> Wählen Sie auf der Webseite inspire.cs.upb.de zumindest eine Ihnen bekannte wissenschaftliche Publikation.
> Auf der Startseite können Sie nach Titeln (1) oder Nachnamen von Autoren (2) suchen. Oder Sie wählen eine Publikation aus der Liste von UPB-Publikationen (3).

Die folgende Liste von Items wurde für die Befragung verwendet:

ID$^{T/O}$ Angabe der ID(s) der von Ihnen gewählten Publikation(en). Die Beantwortung dieser Frage ist optional. Für die Eingabe mehrere IDs trennen Sie diese durch Leerzeichen.

USE1 Ich habe mich bereits intensiv mit den Inhalte der von mir gewählten Publikation auseinandergesetzt

USE2$^{JUN}$ Ich habe bereits eine akademische Arbeit im Bereich der betrachteten Publikation verfasst (z.B. Bachelorarbeit)

USE3$^{JUN}$ Ich habe bereits an einer wissenschaftlichen Publikation aus dem Bereich der betrachteten Publikation mitgewirkt

BEN1 Die zur gewählten Publikation vorgeschlagenen Dokumente eigenen sich gut als Zusatzliteratur

BEN2 Durch die Nutzung Webseite konnte ich nützlich Dokumente finden, die mir bisher unbekannt waren

BIB$^{X}$ Die Publikationen in der Liste eignen sich gut als Vorschläge für zusätzliche Literatur

BIB1 References

BIB2 Cited by

BIB3 Bibliographic coupling

BIB4 Co-citation

BIB5 Citation distances

BIB6 Combined bibliometrics

EOU1 Der Umgang mit der Webseite ist für mich klar und verständlich

EOU2 Der Umgang mit der Webseite erfordert von mir keine große geistige Anstrengung

EOU3 Der Umgang mit der Webseite stellt für mich kein Problem dar

EOU4 Ich finde die Webseite ist leicht zu bedienen

U1 Ich finde die Webseite ist nützlich für meine Literaturrecherche

U2 Die Nutzung der Webseite steigert die Effektivität meiner Literaturrecherche

U3 Die Nutzung der Webseite verbessert meine Literaturrecherche

U4 Die Nutzung der Webseite erhöht die Produktivität meiner Literaturrecherche

A1 Die Nutzung der Webseite zur Literaturrecherche ist eine gute Idee

A2 Die Nutzung der Webseite zur Literaturrecherche ist eine schlechte Idee

A3 Die Nutzung der Webseite macht Spaß

A4 Die Nutzung der Webseite gestaltet die Literaturrecherche interessanter

A5 Ich nutze die Webseite gerne

BI1 Angenommen die Webseite bleibt online verfügbar, dann beabsichtige ich sie zu nutzen

BI2 Wenn die Webseite online verfügbar bleibt, sage ich voraus, dass ich sie nutzen werde

BI3 Ich würde die Webseite in Zukunft immer nutzen, sofern sich die Gelegenheit dazu ergibt

PERS1$^{G/O}$ Geschlecht

PERS2$^{Z/O}$ Alter

UNI1$^{A}$ Ich bin Bachelor-Student an einer Hochschule

UNI2$^{A}$ Ich bin Master-Student an einer Hochschule

UNI3$^{A}$ Ich bin Mitarbeiter an einer Hochschule

UNI4$^{A}$ Sonstiges

STUD1$^{A}$ Mein Fach-/Studiengebiet ist Informatik

STUD2$^{A}$ Mein Fach-/Studiengebiet ist Mathematik

STUD3$^{A}$ Sonstiges

UBP$^{JN}$ Ich arbeite/studiere an der Universität Paderborn

Wenn nicht anders gekennzeichnet, konnten die Items mit einer 5-Punkte-Skala bewertet werden. Dazu wurde jeweils dieser Hilfetext angezeigt: „1: Stimme überhaupt nicht zu 5: Stimme vollständig zu“ Für die gekennzeichneten Items gab es folgende Möglichkeiten zur Beantwortung:

A Auswahl zwischen den Items der jeweiligen Gruppe

G Auswahl zwischen weiblich / männlich / keine Antwort

JUN Auswahl zwischen Ja / Unsicher / Nein

JN Auswahl zwischen Ja / Nein

O Angabe optional

T Angabe von Text

Z Angabe von Ziffern

X Keine Angabe möglich. Das Item dient zur Beschreibung der Itemgruppe

## A.2 SQL ANFRAGEN ZUR BERECHNUNG DER BIBLIOMETRIKEN

Listing 2: SQL Anfrage zu Referenzen

```
1  SELECT id,
2         title,
3         year_year AS YEAR,
4         group_concat(author_surname separator ', ') AS
              authors
5  FROM
6    (SELECT *
7     FROM reference AS REF
8     LEFT JOIN (publication AS pub) ON REF.target=pub.id
9     WHERE REF.source = _ID_) AS sources
10 LEFT JOIN publication_has_author AS p_h_a ON sources.id
       = p_h_a.publication_id
11 GROUP BY publication_id
12 ORDER BY YEAR DESC, title
```

Listing 3: SQL Anfrage zu eingehenden Referenzen

```
1  SELECT id,
2         title,
3         year_year AS YEAR,
4         group_concat(author_surname separator ', ') AS
              authors
5  FROM
6    (SELECT *
7     FROM reference AS REF
8     LEFT JOIN (publication AS pub) ON REF.source=pub.id
9     WHERE REF.target = _ID_) AS targets
10 LEFT JOIN publication_has_author AS p_h_a ON targets.id
       = p_h_a.publication_id
11 GROUP BY publication_id
12 ORDER BY YEAR DESC, title
```

Listing 4: SQL Anfrage zur Bibliografischen Kopplung

```
SELECT COUNT,
        id,
        title,
        group_concat(author_surname separator ', ') AS
            authors,
        year_year AS YEAR
FROM
   (SELECT SOURCE,
           COUNT(SOURCE) AS COUNT
    FROM reference
    INNER JOIN
      (SELECT target
       FROM reference
       WHERE SOURCE=_ID_) AS targets ON reference.target
           = targets.target
    WHERE reference.SOURCE != _ID_
    GROUP BY reference.SOURCE) AS counts
LEFT JOIN (publication AS pub) ON counts.SOURCE = pub.
    id
LEFT JOIN publication_has_author AS p_h_a ON counts.
    SOURCE = p_h_a.publication_id
GROUP BY id
ORDER BY COUNT DESC, YEAR DESC, title
```

Listing 5: SQL Anfrage zur Kozitation

```
SELECT COUNT,
        id,
        title,
        group_concat(author_surname separator ', ') AS
            authors,
        year_year AS YEAR
FROM
   (SELECT target,
           COUNT(target) AS COUNT
    FROM reference
    INNER JOIN
      (SELECT SOURCE
       FROM reference
       WHERE target=_ID_) AS sources ON reference.SOURCE
           = sources.SOURCE
    WHERE reference.target != _ID_
    GROUP BY reference.target) AS counts
LEFT JOIN (publication AS pub) ON counts.target = pub.
    id
LEFT JOIN publication_has_author AS p_h_a ON counts.
    target = p_h_a.publication_id
GROUP BY id
ORDER BY COUNT DESC, YEAR DESC, title
```

Listing 6: SQL Anfrage zu Distanzen von Zitationen

```
1  SELECT id,
2         dist,
3         title,
4         year_year AS YEAR,
5         group_concat(author_surname separator ', ') AS
              authors
6  FROM (
7           (SELECT publication_y AS distid,
8                   sum(distance) AS dist
9           FROM distance
10          WHERE publication_x= _ID_
11          GROUP BY distid) AS dist
12        LEFT JOIN publication AS pub ON dist.distid=pub.
              id)
13 LEFT JOIN publication_has_author AS p_h_a ON pub.id =
       p_h_a.publication_id
14 GROUP BY publication_id
15 ORDER BY dist DESC,
16          YEAR DESC, title
```

# LITERATURVERZEICHNIS


[BSK12] Kevin W. Boyack, Henry Small und Richard Klavans. Improving the Accuracy of Co-citation Clustering Using Full Text. In *Proceedings of 17th International Conference on Science and Technology Indicators*, Bd. 1, S. 155–165, 2012.

[CDG+06] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes und Robert E. Gruber. Bigtable: a distributed storage system for structured data. In *Proceedings of the 7th symposium on Operating systems design and implementation*, OSDI '06, S. 205–218, Berkeley, CA, USA, 2006. USENIX Association.

[CGK08] Isaac G. Councill, C. Lee Giles und Min-Yen Kan. ParsCit: An open-source CRF reference string parsing package. In *Proceedings of the Language Resources and Evaluation Conference (LREC 08)*, 2008.

[DBW89] Fred D Davis, Richard P Bagozzi und Paul R Warshaw. User acceptance of computer technology: a comparison of two theoretical models. *Management science*, 35(8):982–1003, 1989.

[DG04] Jeffrey Dean und Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *OSDI'04: Proceedings of the Sixth Symposium on Operating System Design and Implementation*, San Francisco, CA, December 2004.

[DTS+07] Min-Yuh Day, Richard Tzong-Han Tsai, Cheng-Lung Sung, Chiu-Chen Hsieh, Cheng-Wei Lee, Shih-Hung Wu, Kun-Pin Wu, Chorng-Shyong Ong und Wen-Lian Hsu. Reference metadata extraction using a hierarchical knowledge representation framework. *Decision Support Systems*, 43(1):152 – 167, 2007.

[Eur12] European Commission: Eurostat. Internet access and use in 2012. Press release, 18 December 2012, 2012.

[GB09] Bela Gipp und Jöran Beel. Citation Proximity Analysis (CPA) - A new approach for identifying related work based on Co-Citation Analysis. In Birger Larsen and Jacqueline Leta (Hrsg.), *Proceedings of the 12th International Conference on Scientometrics and Informetrics (ISSI'09)*, Bd. 2, S. 571–575, Rio de Janeiro (Brazil), 2009.

[GBH09] Bela Gipp, Jöran Beel und Christian Hentschel. Scienstein: A research paper recommender system. In *Proceedings of the International Conference on Emerging Trends in Computing (ICETiC'09)*, S. 309–315, Virudhunagar (India), 2009.

[GBL98] C. Lee Giles, Kurt D. Bollacker und Steve Lawrence. CiteSeer: An Automatic Citation Indexing System. In *Proceedings of the third ACM conference on Digital libraries*, 1998.

[Geo11] Lars George. *HBase: The Definitive Guide*. O'Reilly Media, Inc., 2011.

[GHJV11] Erich Gamma, Richard Helm, Ralph Johnson und John Vlissides. *Entwurfsmuster: Elemente wiederverwendbarer objektorientierter Software*. Addison-Wesley, München, 6., 2011. Deutsche Übersetzung von Dirk Riehle, Originalausgabe: Design Patterns (1995).

[HHKL04] I-Ane Huang, Jan-Ming Ho, Hung-Yu Kao und Wen-Chang Lin. Extracting citation metadata from online publication lists using blast. In Honghua Dai, Ramakrishnan Srikant und Chengqi Zhang (Hrsg.), *Advances in Knowledge Discovery and Data Mining*, Bd. 3056 of *Lecture Notes in Computer Science*, S. 539–548. Springer Berlin Heidelberg, 2004.

[Joc09] Maike Jockisch. Das Technologieakzeptanzmodell. In Gerhard Bandow and Hartmut H. Holzmüller (Hrsg.), *„Das ist gar kein Modell!"*, S. 233–254. Gabler, 2009.

[Kes63] M. M. Kessler. Bibliographic coupling between scientific papers. *American Documentation*, 14(1):10–25, 1963.

[LC12] Shengbo Liu und Chaomei Chen. The proximity of cocitation. *Scientometrics*, 91(2):495–511, 2012.

[LCB+06] Huajing Li, Isaac G. Councill, Levent Bolelli, Ding Zhou, Yang Song, Wang-Chien Lee, Anand Sivasubramaniam und C. Lee Giles. CiteSeer$^X$: a scalable autonomous scientific digital library. In *Proceedings of the 1st international conference on Scalable information systems*, 2006.

[Lev66] Vladimir Iosifovich Levenshtein. Binary codes capable of correcting deletions, insertions und reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.

[Lop09] Patrice Lopez. GROBID: Combining Automatic Bibliographic Data Recognition and Term Extraction for Scholarship Publications. In *Proceedings of ECDL 2009, 13th European Conference on Digital Library*, 2009.

[Lop10] Patrice Lopez. Automatic Extraction and Resolution of Bibliographical References in Patent Documents. In Hamish Cunningham, Allan Hanbury und Stefan Rüger (Hrsg.), *Advances in Multidisciplinary Retrieval*, Bd. 6107 of *Lecture Notes in Computer Science*, S. 120–135. Springer Berlin Heidelberg, 2010.

[LR10] Patrice Lopez und Laurent Romary. HUMB: Automatic key term extraction from scientific articles in GROBID. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval '10, S. 248–251, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[LWB+11] Mikael Laakso, Patrik Welling, Helena Bukvova, Linus Nyman, Bo-Christer Björk, und Turid Hedlund. The development of open access journal publishing from 1993 to 2009. *PLoS ONE*, 6(6):e20961, 06 2011.

[Mar73] Irena Marshakova. System of document connections based on references. *Nauchno-Tekhnicheskaya Informatsiya*, 2(6):3–8, 1973.

[MS13] Donald Miner und Adam Shook. *MapReduce Design Patterns*. O'Reilly Media, Inc, 2013.

[PM04] Fuchun Peng und Andrew McCallum. Accurate information extraction from research papers using conditional random fields. In *Human Language Technology conference / North American chapter of the Association for Computational Linguistics annual meeting*, 2004.

[Sam12] Eric Sammer. *Hadoop Operations*. O'Reilly Media, Inc., 2012.

[Schck] Nicolas Schelp. Ähnlichkeitsbestimmung wissenschaftlicher Publikationen – Identifikation von Grundähnlichkeiten und Implementierung eines Algorithmus für dynamisch erweiterbare Clusterings. Masterarbeit, Universität Paderborn, in Druck.

[Sma73] Henry Small. Co-citation in the scientific literature: A new measure of the relationship between two documents. *Journal of the American Society for Information Science*, 24(4):265–269, 1973.

[SW11] Naiara Escuerdo Sanchez und Adrian Wilke. Tools for Awareness in Distributed Research Networks. Arbeit im Seminar Future Social Learning Networks, 2011.

[Var13] Tobias Varlemann. CRITIC: Near Copy Detection in large text corpora – Eine prototypische Umsetzung . Masterarbeit, Universität Paderborn, 2013.

[Whi12] Tom White. *Hadoop: The Definitive Guide*. O'Reilly Media, Inc., 3., 2012.

[Wil10] Adrian Wilke. Analysis and integration of Web 2.0 data sources into a system for analysis and storage of Artefact-Actor-Networks, 2010. Bachelorarbeit.

# VERZEICHNIS DER WEBADRESSEN


[Boot] Bootstrap.
http://twitter.github.io/bootstrap/.
Eingesehen am 30.04.2013

[COinS] COinS: OpenURL ContextObject in SPAN.
http://ocoins.info/.
Eingesehen am 10.02.2013

[CROSS] Turning DOIs into formatted citations.
http://www.crossref.org/CrossTech/2011/11/turning_dois_into_formatted_ci.html.
Eingesehen am 23.04.2013

[Cite] CiteSeerX.
http://citeseerx.ist.psu.edu/.
Eingesehen am 20.04.2013

[DOAJ] Directory of Open Access Journals.
http://www.doaj.org/.
Eingesehen am 25.04.2013

[DOI] Digital Object Identifier System.
http://www.doi.org/.
Eingesehen am 15.04.2013

[FSMI] Fachschaft Mathematik/Informatik Uni Paderborn: Umfrage: Literaturempfehlungen.
https://fsmi.uni-paderborn.de/newsarchiv/news/?tx_ttnews[tt_news]=1849.
Eingesehen am 22.04.2013

[GRO-Demo] GROBID - GeneRatiOn of BIbliographic Data.
http://grobid.no-ip.org/.
Eingesehen am 08.02.2013

[GRO-Git] grobid/grobid · GitHub.
https://github.com/grobid/grobid.
Eingesehen am 10.02.2013

[GRO-SVN] grobid | Free software downloads at SourceForge.net.
http://sourceforge.net/projects/grobid/.
Eingesehen am 05.03.2013

[HBP] HTML5 Boilerplate.
http://html5boilerplate.com/.
Eingesehen am 30.04.2013

[HBase] Apache HBase.
http://hbase.apache.org/.
Eingesehen am 23.04.2013

[HCPA] Hadoop Cluster for Large Scale Publication Analyses.
http://isitjustme.de/2012/07/hadoop-cluster-for-large-scale-publication-analyses/.
Eingesehen am 25.04.2013

[Had-0.20] Hadoop Change Log Release 0.20.0 - 2009-04-15.
http://web.archive.org/web/20130123210955/http://hadoop.apache.org/docs/r0.20.0/changes.html.
Eingesehen am 13.03.2013

[Hadoop] Apache Hadoop.
http://hadoop.apache.org/.
Eingesehen am 23.04.2013

[LIn] LinkedIn.
http://www.linkedin.com/.
Eingesehen am 18.04.2013

[MSAS] Microsoft Academic Search.
http://academic.research.microsoft.com/.
Eingesehen am 20.04.2013

[Maven] Apache Maven.
http://maven.apache.org/.
Eingesehen am 27.04.2013

[Mendeley] Mendeley.
http://www.mendeley.com/.
Eingesehen am 20.04.2013

[MySQL] MySQL: The world's most popular open source database.
http://www.mysql.com/.
Eingesehen am 21.04.2013

[Neo4j] Neo4j, the Graph Database.
http://www.neo4j.org/.
Eingesehen am 21.04.2013

[Nty] Netty.
http://netty.io/.
Eingesehen am 18.04.2013

[PDFBox] Apache PDFBox - Java PDF Library.
http://pdfbox.apache.org/.
Eingesehen am 18.03.2013

[PUSHPIN] PG PUSHPIN Blog | Supporting Scholarly Awareness in Publications and Social Networks.
http://pgpushpin.wordpress.com/.
Eingesehen am 25.04.2013

[ParsCit] ParsCit: An open-source CRF Reference String and Logical Document Structure Parsing Package.
http://aye.comp.nus.edu.sg/parsCit/.
Eingesehen am 09.02.2013

[Play] Play Framework.
http://www.playframework.org/.
Eingesehen am 18.04.2013

[PubUPB] Informatik Publikationen der Universität Paderborn.
http://ddiupbblogs.wordpress.com/2013/02/25/informatik-publikationen-der-universitat-paderborn/.
Eingesehen am 20.04.2013

[RGate] ResearchGate.
http://www.researchgate.net/.
Eingesehen am 20.04.2013

[Scholar] Google Scholar.
http://scholar.google.com/.
Eingesehen am 20.04.2013

[TEI] TEI: Text Encoding Initiative.
http://www.tei-c.org/.
Eingesehen am 10.02.2013

[Titan] TITAN: Distributed Graph Database.
http://thinkaurelius.github.io/titan/.
Eingesehen am 21.04.2013

[URL] RFC 3986: 1.1.3. URI, URL, and URN.
http://tools.ietf.org/html/rfc3986#section-1.1.3.
Eingesehen am 21.04.2013

[knowAAN] Blog der PG knowAAN | Knowledge Awareness in Artefact-Actor-Networks.
http://pgknowaan.wordpress.com/.
Eingesehen am 25.04.2013

# ABBILDUNGSVERZEICHNIS

## TABELLENVERZEICHNIS

# EIDESSTATTLICHE ERKLÄRUNG

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen worden ist. Alle Ausführungen, die wörtlich oder sinngemäß übernommen worden sind, sind als solche gekennzeichnet.

*Paderborn, 30. April 2013*

Adrian Wilke