FH Technikum Wien

MASTER THESIS

zur Erlangung des akademischen Grades
„Master of Science in Engineering“
im Studiengang Multimedia und Softwareentwicklung

Consuming Library Linked Data

ein Prototyp einer Linked Data Webapplikation

ausgeführt von Ulrike Krabo, BSc
1200 Wien, Höchstädtplatz 5

1. Begutachter: Dipl.-Ing. (FH) Mag. (FH) Markus Aulenbach
2. Begutachter: Dr. Wolfram Seidler

Wien, 17.05.2011

FACHHOCHSCHULE
TECHNIKUM WIEN

Eidesstattliche Erklärung

„Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbständig angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher weder in gleicher noch in ähnlicher Form einer anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.“

Ort, Datum	Unterschrift

Kurzfassung

Anwenderinnen und Anwender sind im Web zunehmend mit einem Überfluss an Informationen konfrontiert. Die Auswahl relevanter Informationen fällt immer schwerer, da klassische Volltextsuchmaschinen mittlerweile an ihre Grenzen stoßen. Ein Ansatz zur Überwindung dieser Grenzen ist die Implementierung einer explorativen Suche mithilfe semantischer Technologien.

„*Linked Data*“ beschreibt ein „Konzept“ zum Publizieren von Daten durch semantische Auszeichnung der Informationen und deren Beziehungen. Durch diese semantischen Auszeichnungen wird Computerprogrammen ermöglicht, diese Daten in einem Kontext einzuordnen.

Bibliografische Informationen eigenen sich aufgrund ihrer Strukturiertheit gut dazu semantisch aufbereitet und als *Linked Data* publiziert zu werden. Bibliotheken erhoffen sich dadurch eine bessere Auffindbarkeit im Web und somit eine bessere Nutzung ihrer Dienstleistungen durch die Anwenderinnen und Anwender.

Derzeit existieren nur wenige Webapplikationen, die diese semantisch aufbereiteten (bibliografischen) Informationen nutzen. In der vorliegenden Arbeit werden bestehende *Linked Data* Webapplikationen hinsichtlich Architektur, Workflows zum Nutzen der Daten und Userinterfaces untersucht. Darauf aufbauend wird ein Prototyp präsentiert, der bibliografische Daten unterschiedlicher Datenquellen explorativ zur Recherche und zur Navigation anbietet.

Schlagwörter: Linked Data, Library Linked Data, Linked Data Applikationen, exploratory search

Abstract

Web users increasingly have a hard time finding relevant material because traditional full text search engines hit the wall. One solution to overcome these limits is to implement an exploratory search by using semantic web technologies.

“Linked Data” describes a “concept” of publishing data through semantic markup of the information and its relations. As a result software can “understand” the meaning of the described information.

Due to its structure, bibliographic information lends itself to being semantically rehashed and published as Linked Data. Libraries expect a better visibility in the web and therefore a better use of their services.

Currently there are few web applications that use bibliographically linked data. This paper analyses in detail *Linked Data* web applications and discusses the aspects of data integration, the architecture of Linked Data web applications as well as visualization techniques to implement exploratory searches.

For the purpose of this thesis, a prototype of a web application demonstrates which bibliographic information in the Linked Data can be found.

Keywords: Linked Data, Library Linked Data, Linked Data Applications, exploratory search

Inhaltsverzeichnis

1 Einleitung 8
1.1 Motivation 10
1.2 Aufgabenstellung 10
1.3 Fragestellungen und Hypothesen 11
1.4 Vorgehensweise 12

2 Linked Data 14
2.1 Prinzipien von Linked Data 14
2.1.1 URIs und Identifier 14
2.1.2 RDF und Ontologien 15
2.1.3 Verlinkung 16
2.2 SPARQL 17
2.3 Web of Data 18
2.4 Möglichkeiten der Bereitstellung 21

3 Consuming Linked Data 24
3.1 Definition „Consuming“ 24
3.2 Anforderungen 25
3.3 Strategien 27
3.3.1 Crawling pattern 27
3.3.2 On-the-fly dereferencing pattern 29
3.3.3 Query federation pattern 31
3.3.4 Auswahl der „richtigen“ Strategie 33
3.4 Aspekte der Datenintegration 34
3.4.1 Mapping von Vokabularien 34
3.4.2 Mapping von Ressourcen 34
3.4.3 Datenqualität 35
3.4.4 Lokale persistente Speicherung (Caching) 37
3.5 Auswahl von Datasets 38
3.6 Ausgewählte Tools und Frameworks 40
3.7 Zusammenfassung 41

4 Linked Data Webapplikationen 42
4.1 Klassifikation 42

4.2 Software-Architekturen 45
4.2.1 Referenz-Architektur für semantische Webapplikationen 46
4.2.2 Architektur für LD Webapplikationen 49
4.2.3 Architektur für LD Applikationen (crawling pattern) 50
4.2.4 Schlussfolgerungen 52
4.3 User-Interface Anforderungen 53
4.3.1 Exploratory Search 53
4.3.2 Usability 55

5 Fallstudien 59
5.1 LED 59
5.1.1 Userinterface 59
5.1.2 Architektur und Workflow 61
5.1.3 Schlussfolgerungen 62
5.2 Sig.ma 63
5.2.1 Userinterface 63
5.2.2 Architektur und Workflow 64
5.2.3 Technologie 67
5.2.4 Schlussfolgerungen 67
5.3 RKB Explorer 68
5.3.1 Userinterface 69
5.3.2 Workflow der Integration neuer Datasets 69
5.3.3 Architektur 70
5.3.4 Schlussfolgerungen 70
5.4 Zusammenfassung 71

6 Prototyp 73
6.1 Library Linked Data 73
6.1.1 Anwendungsfälle 74
6.1.2 Analyse LLD Datasets 76
6.2 LLD Search Prototyp 81
6.2.1 Zielgruppen und Anwendungsfälle 81
6.2.2 Konzept 82
6.2.3 Consuming-Strategien 84
6.2.4 Software-Architektur 85

6.2.5 Backend und Indexierung 87
6.2.6 Userinterface 91
6.2.7 Fazit und Future Work 98

7 Zusammenfassung 100

8 Literaturverzeichnis und Quellen 108

Abbildungsverzeichnis 113

Tabellenverzeichnis 114

Abkürzungsverzeichnis 115

Anhang A: LLD Datasets 116

Anhang B: Ausschnitte aus Sourcecode 119

1 Einleitung

Das Web ist im Alltags- und Berufsleben vieler Menschen immer wichtiger, wenn nicht unabdingbar, geworden und bietet die Möglichkeit, jederzeit und von überall auf aktuelle Informationen zuzugreifen. [1], S. 9

Gleichzeitig bietet das Web eine Informationsvielfalt, die für die Benutzerinnen und Benutzer kaum noch aufzunehmen ist. Klassische Suchmaschinen (wie Google) unterstützen die Anwenderinnen und Anwender beim Auffinden von Informationen im Web. Trotz intelligenter Such- und Ranking-Algorithmen stoßen diese Suchmaschinen mittlerweile an ihre Grenzen.

Die Suchmaschinen basieren meist nur auf einer reinen Volltextsuche, was bei vielen Suchanfragen zu verwirrenden Ergebnissen führen kann. Wenn ein Begriff mehrere Bedeutungen hat, kann eine Suchmaschine unmöglich „unterscheiden“ welche dieser Bedeutungen die Anwenderin oder der Anwender gemeint hat.

Weitere Ursachen für die Probleme im Web nennen Hitzler et al. Die dezentrale Struktur und Organisation des Webs führt zwangsläufig zu einer Heterogenität von publizierten Informationen auf unterschiedlichen Ebenen wie Dateiformate, Kodierungstechniken, natürlichen Sprachen, Struktur von Webseiten. Dieser Art der Probleme kann durch intelligente Algorithmen zumindest teilweise begegnet werden. [1], S. 10

Ein weitaus größeres Problem für Suchmaschinen ist aber das „Verstehen“ von implizitem Wissen. So ist es möglich, Informationen nicht nur in Form von Text auf einer Webseite zu kodieren, sondern auf mehreren Webseiten zu verteilen. Der menschliche Nutzer kann diese Informationen kombinieren und somit dieses implizite Wissen aufnehmen. [1]
Sollen Computerprogramme (z.B. von Suchmaschinen) diese Informationen auch kombinieren können, müssen Metainformationen zu diesen Informationen zur Verfügung stehen. Dieses Ziel verfolgt das Konzept des „*Semantic Web“* (SW).

2001 beschreibt Berners-Lee die Idee des SW wie folgt:

> “*The Semantic Web is not a separate Web but an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation.*”[2], S. 37

Das heißt also, dass Webinhalte durch Informationen angereichert werden, um somit eine bessere Zusammenarbeit zwischen Computern und Menschen zu ermöglichen.
Hitzler et al. präzisieren diese Definition noch indem sie erklären, dass das SW Computern ermögliche, Webinhalte intelligenter zu suchen, zu kombinieren und zu verarbeiten,

basierend auf der Bedeutung, die diese Webinhalte für den menschlichen Benutzer hat. [3], S. 11

Praktisch betrachtet, ist das SW keine konkrete Erweiterung des Web, sondern eher eine Wunschvorstellung, wie sich das Web in den nächsten Jahren entwickeln kann. [3], S. 12

Das Web besteht aus einer Vielzahl unterschiedlichen Sprach- und Datenkonstrukten. Im Umfeld der Implementierung des SW wird das Bewusstsein für die Notwendigkeit einheitlicher Standards zunehmend gestärkt. Eine zentrale Rolle bei der Etablierung von Standards übernimmt hier das „World Wide Web Consortium“ (W3C). Das Konsortium beschäftigt sich hauptsächlich mit Standards zu Interoperabilität und Datenaustausch im Web. [3], S. 13

Für die Etablierung des SW eine Vielzahl von Technologien und Protokollen bereit, siehe Abbildung 1. Das erschwert die Umsetzbarkeit des SW und ist sicher ein Grund für die bisherige Nicht-Umsetzung der 2001 veröffentlichten Vision des SW von Tim Berners-Lee.

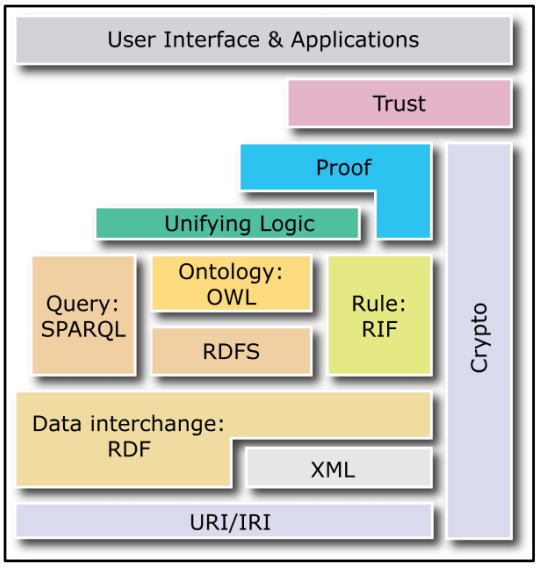

Abbildung 1: Protocol Stack des Semantic Web [4]

Um das semantische Web tatsächlich zu realisieren, muss dafür gesorgt werden, dass semantisch annotierte Daten im Web veröffentlicht werden. Dazu hat Berners-Lee 2006 das „Konzept“ von *Linked Data* (LD) publiziert. Dieses besagt, dass Informationen (die bisher auf Webseiten veröffentlicht wurden) semantisch aufbereitet über eindeutige Identifier angeboten werden. Die Informationen werden semantisch verlinkt, so dass die Art der Beziehung ebenso beschrieben wird. Das ermöglicht ein semantisch beschriebenes Netz verteilter Informationen.

1.1 Motivation

Der Einsatz semantischer Technologien verspricht eine Vielzahl neuer Möglichkeiten, angefangen von verbesserten Recherchemöglichkeiten in Suchmaschinen bis zu verbesserten *Information Extraction* und *Information Retrieval* Ergebnissen.
Das Thema der semantischen Technologien gerät immer mehr in das Blickfeld der Öffentlichkeit. So hat beispielsweise Google im Mai 2009 angekündigt, nun auch Webseiten mit semantischen Annotierungen in Form von Mikroformaten bei der Indexierung zu berücksichtigen.[5]

Webinhalte werden zunehmend semantisch aufbereitet publiziert. Das ermöglicht nun auch Applikationen zu entwickeln, die diese Daten nutzen können. In der Literatur wird zumeist der Begriff *„Linked Data Applications“* (oder *„Linked Data-driven Applications“*) verwendet. Bisher hatten diese Applikationen oftmals nur wissenschaftlichen Charakter. Diese neu-publizierten Daten liefern nun die Grundlage für die Verwendung von semantischer Technologien bei der Entwicklung von Standardapplikationen, die von einer Vielzahl von Benutzern und Benutzerinnen verwendet werden.

Der Boom im Bereich der Applikationen im mobilen Bereich liefert interessante Anwendungsfälle für semantische Technologien, um dem Benutzer abhängig vom Standort bestimmte Dienste anbieten zu können. Aber auch Webapplikationen können durch die Verwendung von semantischen Technologien den Benutzern und Benutzerinnen gezielt die Informationen liefern, die sie auch tatsächlich gesucht haben und darüber hinaus Informationen liefern, die zusätzlich relevant sind, die der Benutzer oder die Benutzerin gar nicht gesucht hat.

Bibliografische Daten sind seit jeher gut strukturiert und eignen sich bestens dazu, diese Informationen fürs Semantic Web bereitzustellen. In der Tat gibt es diverse bibliografische Inhalte, die in den letzten Jahren auf diesem Weg publiziert wurden. Warum gibt es also nicht schon mehr Applikationen, die diese Informationen auch in einsetzbarer Form anbieten?

1.2 Aufgabenstellung

In dieser Arbeit soll der aktuelle Forschungsstand des Bereichs der LD Webapplikationen untersucht werden, also jene Webapplikationen, die die Daten des LD Web nutzen.

Ziel dieser Arbeit soll die Implementierung eines Prototyps einer LD Webapplikation sein, die bibliografische Daten des LD Web explorativ zur Recherche anbietet.

Zur Konzipierung dieses Prototyps müssen zunächst die Strategien zum „Consuming“ von *LD* als auch Software-Architekturen von *LD* Webapplikationen untersucht werden.

1.3 Fragestellungen und Hypothesen

Als zentrale Fragestellung soll in dieser Arbeit geklärt werden, welche Prinzipien Webapplikationen implementieren sollten, um semantisch annotierte Daten in Form von LD zu nutzen (am Beispiel des Bereichs bibliografischer Daten). Dazu werden zwei Bereiche untersucht: „Consuming LD“ und „LD Webapplications“.

Die Fragestellung zum Themenbereich „Consuming“ lautet:

- Welche Strategie ist am geeignetsten, um Daten des LD Web in Webapplikationen zu nutzen?

Bei dieser Fragestellung wird es zu einem argumentativen Vergleich unterschiedlicher Ansätze kommen, wie die dezentral vorliegenden Daten abgefragt werden können.
Folgende Hypothesen werden angenommen:

- Aufgrund der Dezentralität des Webs ist zu erwarten, dass die Vielzahl solcher Datenquellen nicht adhoc abgefragt werden können, und somit eine offline Vorverarbeitung nötig ist.
- Weiter ist zu erwarten, dass komplexe Methoden zur Datenintegration notwendig sind, um gleiche oder ähnliche Daten zu erkennen und für einen potenziellen Benutzer entsprechend aufzubereiten.

Die Fragestellung zum Themenbereich „Linked Data Webapplications“ lautet:

- Wie muss eine Webapplikation aufgebaut sein, um die Daten des LD Web für einen Benutzer effizient recherchierbar zu machen?

Hier wird es zu einer Analyse der Architekturen von LD Webapplikationen kommen. Folgende Hypothesen werden angenommen:

- Es ist zu erwarten, dass es Komponenten klassischer Webapplikationen geben wird aber auch neue Komponenten, die insbesondere die Datenintegration semantisch annotierter Daten unterstützen.
- Auch ist zu erwarten, dass die Visualisierung der Daten eine wesentliche Rolle spielen wird, um die Möglichkeiten von LD auszunutzen.

Die gewonnenen Erkenntnisse aus beiden Fragestellungen werden zunächst mittels drei Fallstudien überprüft und sollen dann mithilfe der Implementierung eines Prototyps

empirisch ausgetestet werden. Dieser Prototyp ist eine Webapplikation, die bibliografische Daten des LD Web, explorativ recherchierbar macht.

1.4 Vorgehensweise

Kapitel 2: Linked Data

Zunächst werden im Kapitel zwei die Grundlagen von *Linked Data* erklärt. Dazu werden technische Konzepte und Aspekte der Datenmodellierung dargelegt. Dieses Kapitel dient zum grundlegenden Verständnis des Themengebiets „*Linked Data*".

Kapitel 3: Consuming Linked Data

In Kapitel drei werden Strategien zum Nutzen („Consuming") der Daten des LD Webs vorgestellt und diskutiert. Dabei werden unterschiedliche Strategien des Consuming aufgezeigt und argumentativ diskutiert. Zudem werden Herausforderungen der Datenintegration beim *Consuming* dieser Daten aufgezeigt und Lösungsansätze präsentiert.

Kapitel 4: Linked Data Webapplikationen

In Kapitel vier werden dann LD Webapplikationen untersucht, die diese *Consuming*-Strategien einsetzen. Dazu wird zunächst der Begriff der „Linked Data Webapplikation" definiert, Software-Architekturen aufgezeigt und Userinterface-Anforderungen für diese Applikationen definiert.

Kapitel 5: Fallstudien

In Kapitel 5 werden dann die Konzepte aus Kapitel drei und vier kombiniert und beispielhaft drei LD Webapplikationen bezüglich Consuming-Strategien, Software-Architekturen und UserInterfaces analysiert.
Darauf aufbauend werden Empfehlungen für die Konzeption von LD Webapplikationen gegeben, die dann in Kapitel 6 mittels eines Prototyps teilweise umgesetzt werden.

Kapitel 6: Prototyp

In diesem Kapitel wird schließlich ein Prototyp einer *LD* Webapplikation präsentiert, der bibliografische Daten des *LD* Web zur Recherche und zur explorativen Recherche anbietet.
Consuming-Strategien aus Kapitel drei werden hier ausgetestet, als auch Aspekte der LD Applikationen aus Kapitel vier finden hier ihre Anwendung.

Zunächst werden Datenquellen des *LD* Web analysiert, um dann beispielhaft zwei bis drei Datasets für diesen Prototyp auszuwählen.

Das Konzept des Prototyps wird zunächst präsentiert und dann die Implementierung dieses Konzepts anhand ausgewählter Aspekte aufgezeigt.

Kapitel 7: Zusammenfassung

Dieses Kapitel bildet den Abschluss und fasst die Ergebnisse dieser Arbeit abschließend zusammen.

Anmerkungen:

Aus Gründen der besseren Lesbarkeit wird auf die ausführliche Schreibweise „Anwenderinnen und Anwender“ verzichtet. Stattdessen wird die kurze Schreibweise (männliche Form) verwendet. Selbstverständlich werden damit die weibliche und männliche Form gleichberechtigt angesprochen.

Viele der in der Arbeit verwendeten Fachbegriffe sind in englischer Sprache übernommen und werden nicht übersetzt.

2 Linked Data

Das folgende Kapitel gibt eine Einführung in das Thema *Linked Data* (LD). Es wird das technische Konzept von LD erklärt und später wird auf die Möglichkeiten der Bereitstellung und Nachnutzung eingegangen.

Daten sind in vielen unterschiedlichen Formen im Web verfügbar. Teilweise als Webseiten mit direkt crawl-baren Inhalten oder versteckten Inhalten. Teilweise aber auch über APIs, wie sie große Dienstleister im Web wie Amazon oder Google anbieten.

Das große Problem ist die Vielfalt der Möglichkeiten. Sollen bestimmte Datenquellen verwendet werden, muss zunächst geklärt werden, wie auf diese Datenquellen zugegriffen werden kann. Das Konzept von LD bietet einen allgemeinen Ansatz, der die Grenzen zwischen Applikationen sprengen könnte.

Bisher wurden die Daten auf Applikationsebene (in Form von Mashups) kombiniert. Mit LD können Daten jetzt bereits auf Datenebene kombiniert werden.

2.1 Prinzipien von Linked Data

“Linked Data” ist kein Standard, vielmehr handelt es sich dabei um eine Sammlung von Best Practices, um strukturierte Daten im Web zu veröffentlichen und zu verlinken. [6], Kapitel 2

Diese Prinzipien wurden 2006 von Berners-Lee veröffentlicht und umfassen folgende Regeln:

1. „*Use URIs as names for things.*
2. *Use HTTP URIs, so that people can look up those names.*
3. *When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL).*
4. *Include links to other URIs, so that they can discover more things.*”[7]

Diese Prinzipien werden in den nachfolgenden Unterkapiteln näher erläutert.

2.1.1 URIs und Identifier

Im klassischen Web werden (HTML-)Dokumente veröffentlicht, was zur Folge haben kann, dass innerhalb eines Dokuments Informationen über viele verschiedene *Themen* publiziert werden.

Mit LD werden diese „*Themen*“ als Ressource bezeichnet und separat über URIs angeboten:

> *„A resource is simply anything that can be identified with a Universal Resource Identifier (URI). And by design, anything we can talk about can be assigned a URI*". [8], S. 65

Die zweite Regel besagt, dass HTTP als Protokoll verwendet werden soll. Somit ist diese Ressource nicht nur adressierbar, sondern auch dereferenzierbar. Das heißt, die Information zu dieser Ressource kann jederzeit aufgerufen werden. [6], Kapitel 2.2

Einer der wichtigsten Grundsätze in diesem Zusammenhang ist die Stabilität und Verfügbarkeit von http URIs. Die Notwendigkeit, dass URIs stabil bleiben sollten, und sich nach Möglichkeit nie ändern, wird von Sauermann und Cyganiak in [9] beschrieben.

Auch sollte berücksichtigt werden, dass die Informationen, die über diesen Identifier ausgeliefert werden für den menschlichen Benutzer/Benutzerin genauso zu lesen sein sollten wie für ein Computerprogramm. Das bedeutet es müssen unterschiedliche Formate ausgeliefert werden – HTML für den menschlichen Leser und RDF für ein Computerprogramm. (Weitere Erklärungen zu RDF folgen im nächsten Kapitel). Dieses Prinzip wird als *Content Negotiation* bezeichnet.[9]

2.1.2 RDF und Ontologien

Die dritte Regel der vier LD Prinzipen besagt, dass „nützliche Informationen" unter Zuhilfenahme von Standards wie RDF und SPARQL ausgeliefert werden sollen.

Das „Resource Description Framework"(RDF) ist ein Standard, um Webinhalte beschreiben zu können. Dabei stellt RDF lediglich ein graphenbasiertes Datenmodell zur Verfügung. Dieses Datenmodell kann in unterschiedlichen Formaten serialisiert werden, beispielsweise RDF/XML, RDFa, Turtle, N-Triples, N3, RDF/JSON.

Die Beschreibung einer Ressource in RDF wird durch eine Vielzahl von sogenannten Triples ausgedrückt. Ein Tripel besteht dabei aus den drei Teilen Subjekt, Prädikat und Objekt. Am Beispiel des natürlich sprachlichen Satzes „Ulrike studiert am Technikum Wien." würde bedeuten, dass „Ulrike" das Subjekt, „studiert" das Prädikat und „am Technikum Wien" das Objekt ist.

Im RDF Datenmodell werden Literale und Links unterschieden. Literale können Strings, Zahlen oder Daten sein. Links hingegen beschreiben eine Beziehung zwischen Ressourcen. Prädikate werden durch RDF-Links auf semantische Modelle repräsentiert. Damit kann der Kontext dieses Prädikats (und somit der Aussage) ermittelt werden. [6], Kapitel 2.4.1

Zur Beschreibung dieser semantischen Modelle Simple Knowledge Organization System (SKOS), RDF Schema (RDFS) und Web Ontology Language (OWL) zur Verfügung. SKOS wird hauptsächlich für Thesauri, Taxonomien und Verzeichnisstrukturen verwendet, um Konzepte in Form von Klassen und deren Eigenschaften zu beschreiben. Mit RDFS und OWL können klassifizierte Beziehungen zwischen Termen (z.B. in Form von Ontologie) beschrieben werden. Damit können implizite Informationen durch Schlussfolgerungen gewonnen werden. [6], Kapitel 4.4

Es steht eine Vielzahl von Vokabularien und Ontologien zur Beschreibung der semantischen Informationen zur Verfügung. Bei der Datenmodellierung sollte darauf geachtet werden, weitverbreitete Vokabularien zu verwenden und neue Ontologien nur zu definieren, wenn es unbedingt notwendig ist. Eine Liste weitverbreiteter Vokabularien nennen Heath und Bizer. [6], Kapitel 4.4.

2.1.3 Verlinkung

Die vierte Regel der LD Prinzipien besagt, dass Ressourcen untereinander verlinkt werden sollen. Diese Verlinkung wird ebenfalls über RDF ausgedrückt. Somit ist es möglich typisierte Links zu anderen Ressourcen zu setzen. Erst durch die Verlinkung entsteht ein Netz aus in-Beziehung-stehenden Daten.

Heath und Bizer unterscheiden folgende Arten von Links:

- „Relationship Links
- Identity Links
- Vocabulary Links“[6], Kapitel 2.5

Als *Relationship Links* sind jene Links anzusehen, die ausdrücken, dass eine Ressource mit anderen Ressourcen in Beziehung steht, beispielsweise Menschen, Firmen, Orte, Bücher etc. [6], Kapitel 2.5

Identity Links werden verwendet um auszudrücken, dass es noch weitere URIs gibt, die diese Ressource beschreiben. So können zum Beispiel Informationen zu Personen in unterschiedlichen Quellen ähnliche Informationen enthalten. Dass es mehrere URIs für die gleiche Ressource gibt hat laut Heath und Bizer folgende Vorteile:
Die Ressource kann aus unterschiedlichen Blickwinkeln betrachtet werden und somit unterschiedliche Meinungen widerspiegeln. Die Verlinkung von unterschiedlichen URIs zur selben Ressource ermögliche den Datennutzern einschätzen zu können, wer (z.B. welche Institution) etwas über die Ressource ausgesagt hat. Zudem gäbe es keinen „central point of failure“, wenn die Daten dezentral verteilt sind. [6], Kapitel 2.5

Dennoch ist es notwendig, alle Informationen zu einer Ressource zu aggregieren. Dazu werden in den Daten Same-As-Beziehungen definiert, die aussagen unter welchen URIs noch Informationen zu dieser Ressource existieren. Dazu gibt es Services, über die solche Same-As-Beziehungen abgefragt werden können.

Vocabulary Links verlinken die Beschreibung der verwendeten Vokabularien. Somit ermöglicht man eine Art Selbstbeschreibung der Daten. Potentielle Benutzer (egal ob Mensch oder Maschine) könnten so diese Informationen in einen Kontext einordnen und deren Bedeutung erfahren.[6], Kapitel 2.5

2.2 SPARQL

„SPARQL Protocol and RDF Query Language“ (SPARQL) ist ein Standard zur Abfrage von RDF-basierten Informationen zur Repräsentation der Ergebnisse in XML und ein Protokoll. [3], S. 262

Die „SPARQL Query Language“ ist eine Abfrage-Sprache für RDF-basierte Datenquellen. Zur Formulierung einer Query wird die Turtle-Syntax verwendet. Mit Hilfe von Variablen ist es möglich jene Werte anzugeben, die im Ergebnis aufscheinen sollen. [3], S. 262

Abbildung 2zeigt ein Beispiel einer SPARQL-Query. Analog zu SQL wird im SELECT angegeben, welche „Felder“ im Ergebnis angezeigt werden sollen. In der WHERE-Klausel können beliebige Bedingungen definiert werden. Variablen ermöglichen es auf einzelne konkrete Werte zuzugreifen. In dieser Beispiel-Abfrage werden URIs jener Ressourcen ermittelt, die ein Schlagwort haben. Ausgegeben werden die URI der Ressource und die URI des Schlagworts.

```
SELECT DISTINCT ?uri, ?subject
WHERE
{
   ?uri <http://purl.org/dc/terms/subject> ?subject .
}
LIMIT 100
```

Abbildung 2: SPARQL Query

Die SPARQL Query Language hat noch einige weitere Merkmale typischer Query-Languages wie Gruppierungen, Funktionen, Optionale Werte, Prüfung von Datentypen, Operatoren (String, Boolean, Arithmetisch), Sortierkriterien und Von-Bis-Bereiche. ([3], S.262-282)

Eine SPARQL-Query kann auf unterschiedliche Arten gestellt werden. So ist es möglich, eine SPARQL-Query direkt auf eine Ressource (URI) anzuwenden oder aber auf ein SPARQL-Service via HTTP oder SOAP. [10][3][11]

Über das Protokoll ist es auch möglich verteilte SPARQL-Queries über mehrere SPARQL-Dienste durchzuführen. Dazu wird das keyword SERVICE benötigt – siehe Abbildung 3.

```
SELECT ?coauthor ?name
WHERE {
  {
    SERVICE <http://dblp.uni-trier.de/> {
       ?coauthor foaf:name ?name
    }
    UNION
    SERVICE <http://dbpedia.org> {
?coauthor foaf:name ?name
    }
  }
…
}
```

Abbildung 3: verteilte SPARQL-Query (Auszug aus [12], S. 127)

2.3 Web of Data

Publiziert ein Anbieter Daten unter den LD Prinzipien wird das in der Literatur üblicherweise als *LD Dataset* bezeichnet. Alle Datasets zusammengenommen, die miteinander verlinkt sind, werden als „*Web of Data“* bezeichnet.

Dieses *Web of Data* enthält Billionen[13] von RDF Statements unterschiedlichster Quellen aus Bereichen wie geografische Orte, Menschen, Firmen, Bücher, wissenschaftliche Veröffentlichungen, Filme etc. [6], Kapitel 3

Abbildung 4(auf Seite 20) zeigt eine grafische Visualisierung des Web of Data, der sogenannten Linking Open Data Cloud (LOD Cloud). Datasets, die unter Open-Data-Standards publiziert werden, werden als *Linked Open Data* bezeichnet. Datasets hingegegen, die bestimmten Restriktionen unterliegen werden meist als *Linked Closed Data* bezeichnet – oftmals auch als *Linked Enterprise Data*.

Diese Abbildung wird aus Daten generiert, die im Comprehensive Knowledge Archive Network (CKAN) veröffentlicht sind. CKAN ist ein Katalog, der Open-license Datasets in beliebigen Formaten sammelt.[6], Kapitel 3.1

Die unterschiedlichen Farben in der Abbildung repräsentieren unterschiedliche Domains – siehe Legende in der Grafik. Der Bereich der „Publications“ (grün) wird für die Realisierung des Prototyps in Kapitel 6 genauer betrachtet. Auf die anderen Domains wird in dieser Arbeit nicht weiter eingegangen. Wie in Tabelle 1 zu erkennen ist, stellen die *Publications* zwar die größte Anzahl an Datasets zur Verfügung, jedoch 8,45% aller Triples in der LOD Cloud. 19,71% aller ausgehenden Links stammen aus dem Bereich der Publications.

Domain	Datasets	Triples	%	Triples/Dataset	(Out-)Links	%2
Government	25	11.620.223.569	43,00%	464808942,76	17.658.869	4,46%
Geographic	16	5.907.260.125	21,86%	369203757,81	16.618.815	4,20%
Life sciences	42	2.664.119.184	9,86%	63431409,14	200.417.873	50,67%
Media	26	2.454.298.811	9,08%	94396108,12	50.376.504	12,74%
Publications	67	2.282.435.732	8,45%	34066204,96	77.951.898	19,71%
Cross-domain	20	2.041.092.718	7,55%	102054635,90	29.114.820	7,36%
User-generated conten	7	57.463.756	0,21%	8209108,00	3.402.228	0,86%
		27.026.893.895	100,00%		395.541.007	100,00%

Tabelle 1: LOD Cloud Datasets (aus [13] am 11.03.2011)

Die aktuelle Statistik der LOD Cloud ist unter [13]abrufbar.

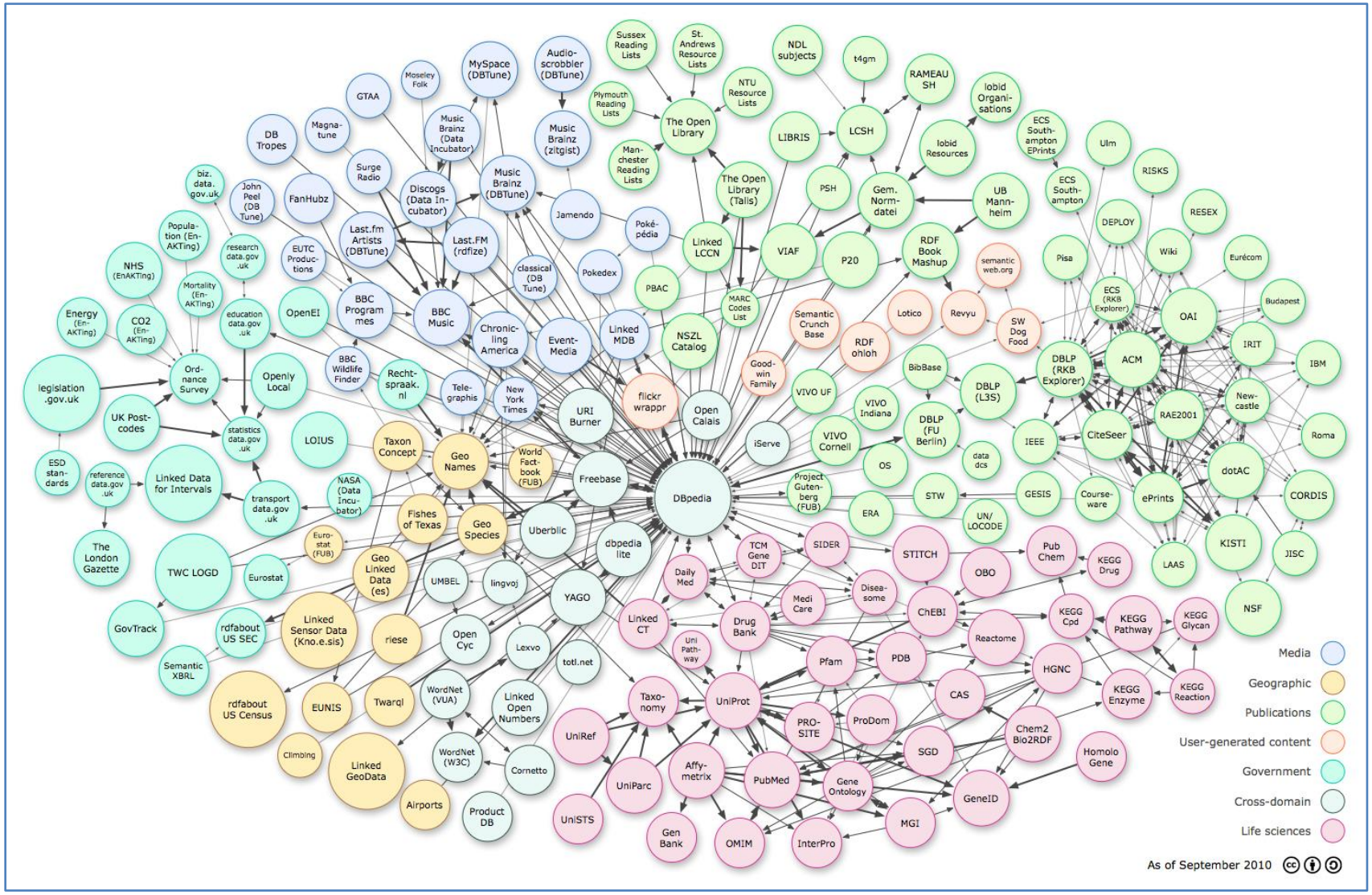

Abbildung 4: Linking Open Data Cloud, Stand: September 2010[14]

2.4 Möglichkeiten der Bereitstellung

In diesem Kapitel werden kurz die Möglichkeiten der Bereitstellung von Daten als LD Prinzipien aufgezeigt. Jedoch kann das nur ein kleiner Überblick sein, da dieses Thema ausführlich auf diversen Konferenzen, studentischen Abschlussarbeiten und Büchern diskutiert wurde.

Abbildung 5 (auf Seite 22) zeigt eine Übersicht über möglicher Vorgangsweisen Daten nach LD Prinzipien zu veröffentlichen. In dieser Abbildung sind unterschiedliche Publikations-Workflows dargestellt:

- Wrapper für Daten in relationalen Datenbanken
- Auszeichnungssprachen für HTML-Seiten
- Konvertierung der Daten in RDF und Speicherung in einem RDF-Store
- Konvertierung der Daten in RDF und Speicherung der Dateien im Filesystem[6], Kapitel 5.1

Für alle diese Arten stehen unterschiedliche Tools und Programme zu Verfügung und können unter [6] nachgelesen werden. Auch eine Checkliste der wichtigsten Schritte wird hier angeboten.

Zusätzlich zu der Möglichkeit Daten am eigenen Webserver zu veröffentlichen stehen auch einige Hosting-Dienste zur Verfügung: Amazon[1], RKBExplorer[2], Open Virtuoso[3], Talis[4], publishmydata[5] etc.

[1] http://aws.amazon.com/publicdatasets/ [zuletzt angesehen 17.05.2011]

[2] http://rkbexplorer.com[zuletzt angesehen 17.05.2011]

[3] http://lod.openlinksw.com/[zuletzt angesehen 17.05.2011]

[4] http://blogs.talis.com/n2/cc [zuletzt angesehen 17.05.2011]

[5] http://publishmydata.com [zuletzt angesehen 17.05.2011]

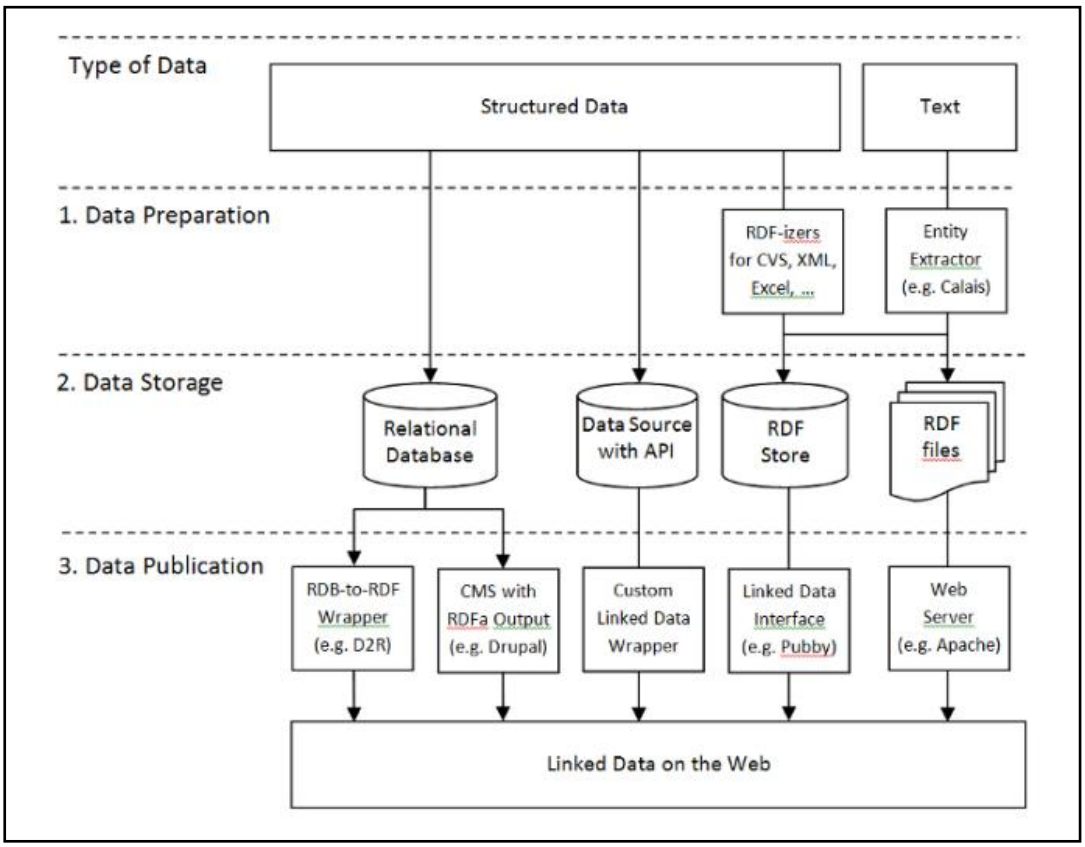

Abbildung 5: Übersicht "Publishing Linked Data"[6], Abbildung 5.1

Zusätzlich zur reinen Bereitstellung der Daten als LD soll die Verlinkung zu bestehenden Ressourcen vorangetrieben werden. Hilfreich sind hier Mapping-Dienste wie sameAs.org oder lokal installierbare Tools wie der SILK-Server[6].

Einer der wichtigsten Schritte ist auch die Beschreibung des publizierten Datasets durch Metadaten. Empfohlen werden hier Semantic Sitemaps und Beschreibungen mithilfe des „Vocabulary of Interlinked Datasets“ (voiD). Beschreibungen dieser Art ermöglichen potenziellen Nutzern dieser Daten vorab einen Einblick in die Art der Daten zu erhalten, um beispielsweise eine Qualitätsbeurteilung durchzuführen. Dies könnte ein Kriterium dafür sein, um zu entscheiden, ob die Daten die Anforderungen des potenziellen Nutzers erfüllen.
Abbildung 6 (auf Seite 23) zeigt, wie diese Daten dann von potenziellen Nutzern verwendet werden könnten.

[6]http://www4.wiwiss.fu-berlin.de/bizer/silk/server/[zuletzt angesehen 17.05.2011]

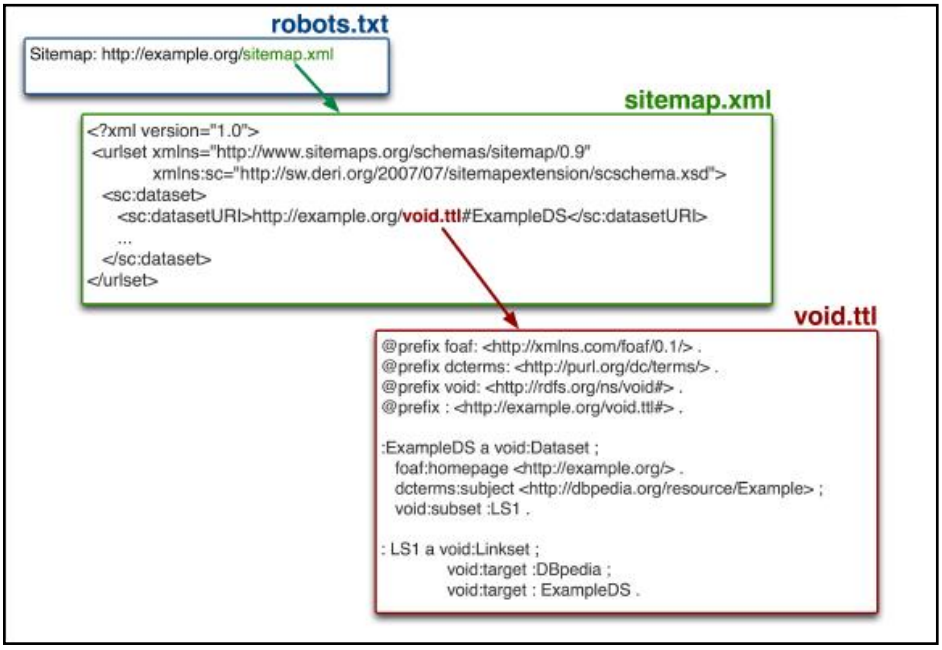

Abbildung 6: semantic sitemaps und voiD [15]

Zusätzlich zur Bereitstellung der Ressourcen über URIs und SPARQL wird von diversen Autoren empfohlen die Daten über REST-Schnittstellen anzubieten, da es mittlerweile Standard in der Webentwicklung ist. [16][17].
Ein Vergleich zwischen LD-Prinzipien und REST-Prinzipien stellen Page et al. [16] an und empfehlen RESTful-Services auch für LD-Services u.a. als einen alternativen Zugang zu SPARQL und als Schreib-Zugang auf die Ressourcen. Eine konkrete Beispiel-Implementierung für RESTful SPARQL beschreiben Wilde und Hausenblas [17].

Die Möglichkeiten zur Nachnutzung von Daten, die als LD bereitgestellt wurden sind vielfach und werden im nächsten Kapitel beschrieben.

3 Consuming Linked Data

Nachdem in Kapitel zwei grundlegend die Prinzipien von LD erklärt wurden, werden in diesem Kapitel nun verschiedene Strategien zum Nutzen („Consuming“) dieser Daten diskutiert.

Zunächst werden Anforderungen für Systeme definiert, die die verteilten Daten des LD Web nutzen wollen. Diese Anforderungen werden dann auf die zu diskutierenden Strategien angewendet.

Ziel des Kapitels soll schließlich eine Empfehlung sein, welche Strategie bei derzeitigem Stand der Technik (für die Entwicklung von Webapplikationen) angewendet werden kann.

LD wird in unterschiedlichen Formaten im Web veröffentlicht:

- HTTP URIs
- RDF-Dumps
- SPARQL-Interfaces

Jedes dieser Formate ist für andere Anwendungszwecke geeignet. HTTP URIs sind für die Adhoc-Abfrage der Informationen zu einer Ressource geeignet. RDF-Dumps können heruntergeladen werden und in einer eigenen Datenbank eingespielt werden. SPARQL-Interfaces eigenen sich für komplexe Abfragen eines oder mehrerer Datasets. Dementsprechend können zur Online und Adhoc-Abfrage der Daten des LD Webs HTTP URIs und SPARQL-Interfaces verwendet werden.

3.1 Definition „Consuming“

„Consuming“ ist das „Nutzen“ von LD zum Zweck der Verarbeitung und Anzeige setzt sich aus folgenden Teilbereichen zusammen:

- „discovery
- accessing
- processing“

[15], S. 12

Als **discovery** (oder „resource discovery“) wird das Auffinden von Ressourcen bezeichnet. Dazu muss die URI oder Metainformationen dazu über eine allgemein bekannte Adresse veröffentlicht werden. [15], S. 12ff

Teilbereiche des *discovery* sind „*lookup services*“ und „*browsing*“.

Als **lookup services** werden Dienste bezeichnet, die es ermöglichen einen Term (z.b. ein keyword) nachzuschlagen und als Resultat eine URI zu erhalten. Zumeist ist als Input auch eine URI erlaubt, um beispielsweise ähnliche Links zu einer Ressource zu erhalten.[15], S. 13

Als **browsing** wird das Nachverfolgen von Links bezeichnet. Das ist auch das Prinzip auf dem das klassische Web aufgebaut ist. Dieses Prinzip wird als „follow-your-nose“ (FYN) bezeichnet.[15], S. 14

Das physikalische Zugreifen auf ein Serialisierungsformat einer Ressource wird als „***access***“ bezeichnet.[15], S. 14

Zusätzlich zum Auffinden und Zugreifen auf die Informationen sollen die Daten letztendlich auch genutzt, also verarbeitet werden. Dieser Teilbereich wird als **processing** bezeichnet. Ein Beispiel für die Verarbeitung ist das Reasoning, womit es möglich wird aus den vorhandenen Informationen neue Informationen zu generieren und somit womöglich neues Wissen zu generieren.[15], S. 14

3.2 Anforderungen

Die Dezentralität des LD Web stellt Entwickler von LD Applikationen vor eine Reihe von Herausforderungen. Einige dieser Herausforderungen und Methoden, und wie ihnen begegnet werden kann, wird in diesem Kapitel dargelegt.

Abfragesprache

Görlitz und Staab erklären, dass es eine deklarative Abfragesprache geben muss mit der die unterschiedlichen Datenquellen abgefragt werden können. Für RDF-Daten ist das **SPARQL**.[12], S. 113
Das würde bedeuten, dass jedes Dataset, das abgefragt werden soll, ein SPARQL-Interface zur Verfügung stellen muss oder dass es einen Data-Aggregator-Dienst gibt, der diese Daten importiert und dafür dann ein SPARQL-Interface anbietet.

Datenkatalog

Unterschiedliche Datasets können verschiedene oder auch gleiche Aussagen über Ressourcen veröffentlichen. Gleichsam können diese unterschiedlichen Ressourcen verschiedene Vokabularien verwenden, um die gleiche Aussage zu formulieren.

Daher ist es notwendig, dass das System einen **Datenkatalog** besitzt mit dem es sowohl möglich ist gleiche Ressourcen zu erkennen als auch gleiche Aussagen (unterschiedlicher Vokabulare) zu identifizieren, durch Mapping der jeweiligen Begriffe. [12], S. 113

Query Optimierung

Je mehr Datasets verwendet werden sollen, desto wichtiger ist es die Anfragen so zu optimieren, dass die Kosten für die Kommunikation und Verarbeitung der Anfrage so niedrig wie möglich gehalten werden. [12], S. 113
Görlitz und Staab beschreiben die Idee der Query Optimierung wie folgt:

> „[…] *to find a query execution plan which minimizes the processing cost of the query and the communication costs for transmitting query and results between mediator and Linked Data sources*“ [12], S. 126

Weiter wird ein „query execution plan“ wie folgt definiert:

> „*A query execution plan is an executable query plan where logical operators are replaced by physical operators*. “[12], S. 127

Teilweise kann hier auf Algorithmen der Entwicklung von verteilten Datenbanken zurückgegriffen werden, teilweise müssen hier im Kontext des Webs neue Algorithmen entwickelt werden.[18]

Datenprotokoll

Görlitz und Staab nennen eine weitere Anforderung, die Verwendung eines **Datenprotokoll**s, mit dem es möglich ist zu bestimmen, wie Anfragen und Ergebnisse zwischen Datasets auszutauschen sind. [12], S. 113

Bewertung der Datenqualität

Je mehr Datasets an der Beantwortung einer Suchanfrage beteiligt sind, desto wichtiger ist es über das **Ranking** der Ergebnisse nachzudenken.
Aussagen (Triples) können nicht aktuell, ungenau oder auch falsch sein. Daher ist es wichtig die Qualität der Aussagen zu ermitteln. Die Datenqualität könnte somit ein Kriterium sein, um Ergebnisse zu ranken. Weitere Kriterien liegen in der Antwortzeit, Popularität oder der Stellenwert der Datenquelle sein – ähnlich dem „Pagerank“-Algorithmus von Google. [6], Kapitel 6.3.5

Ein Kriterium, um die Qualität des Datasets u.a. zur Verwendung in Ranking-Algorithmen zu messen, sind die sogenannte „**Provenance**“ Informationen. Provenance Informationen stellen Aussagen über die Aussage bereit – d.h.

beispielsweise WER hat diese Aussage erzeugt und WER hat sie WANN veröffentlicht. Durch diese Art der Information kann (eventuell) die Qualität der Aussage (und des Datasets) und die Glaubwürdigkeit und somit der Stellenwert dieses Datasets bestimmt werden.[19], S. 1

3.3 Strategien

Heath und Bizer unterscheiden drei Consuming-Strategien:

- „Crawling pattern
- On-the-fly dereferencing pattern
- Query federation pattern“ [6], Kapitel 6.3

Bei Systemen, die das „***Crawling pattern***“ verwenden, werden (relevante) Daten in einem Batchvorgang vorab geholt („gecrawled“) und dann in einer zentralen Datenbank gespeichert. Abfragen werden dann an diese lokale Datenbank gestellt.

Systeme, die das „***On-the-fly dereferencing pattern***“ verwenden, basieren ausschließlich auf den Daten, die zum Zeitpunkt der Abfrage verfügbar sind – mit der Einschränkung, dass der Nutzer/die Nutzerin genau wissen muss, wo sich die Information befindet.

Systeme, die das „**Query federation pattern**“ einsetzen, führen eine verteilte Suche in allen bekannten (oder ausgewählten) Datenquellen durch.

In den folgenden Abschnitten werden diese drei Strategien genauer betrachtet und mögliche Vorteile und Nachteile diskutiert und dabei Bezug auf die in Kapitel 3.2beschriebenen Anforderungen genommen.

3.3.1 Crawling pattern

Heath und Bizer beschreiben, dass bei diesem Ansatz das „Webofdata“ vorab „gecrawled“ wird, indem RDF Links verfolgt werden und diese Daten (oder relevante Teile davon) lokal gespeichert werden. [6], Kapitel 6.3

Diese Definition beschreibt nur die Verfolgung von RDF Links (also URIs) im Web und schließt aber RDF Dumps und SPARQL-Quellen aus. Harth et al prägen den Begriff „materialization-based approaches“ (MAT), der über das reine Crawlen noch hinausgeht. Dieser wird von ihnen wie folgt erklärt:

> „[…] collect the data from all known sourcesin advance, preprocess the combined data, and store the results in a centraldatabase;queries are evaluated using the local database.“ [18], S. 1

Somit sind MAT-Systeme also jene Systeme die vorab Daten abholen, diese aufbereiten und dann in einer zentralen Datenbank speichern. Solche Systeme werden üblicherweise als Datawarehouse-Systeme oder (semantische) Suchmaschinen bezeichnet. Bei Datawarehouse-Systemen steht das Speichern der Daten im Vordergrund, mit dem es dann möglich ist, Abfragen über mehrere Datenquellen zu kombinieren. Semantische Suchmaschinen haben eher den Fokus auf der Indexierung. Mit diesen Systemen ist es möglich, große Datenmengen in sehr geringer Zeit zu durchsuchen.

Diskussion der Anforderungen

Die Algorithmen der Batchverarbeitung zum Holen (crawlen) der Daten müssen unterschiedliche Abfragesprachen (z.B. SPARQL) und unterschiedliche RDF-Serialisierungsformate unterstützen, um alle gefundenen Daten auch zu verarbeiten. Zudem ist es empfehlenswert in der Verarbeitung Datenformate so zu konvertieren, dass diese lediglich in einem Datenformat gespeichert werden, um Abfragen des Systems entsprechend zu vereinfachen.

Da alle Daten in einer oder mehreren lokalen Datenbanken gespeichert sind, ist es möglich, einen zentralen Datenkatalog zu erstellen, der Mappings zwischen Ressourcen und Vokabularien ermöglicht. Somit können ähnliche Aussagen (aufgrund der Verwendung unterschiedlicher Vokabularien) und gleiche Ressourcen (basierend auf sameAs-Links) identifiziert werden.

Query Optimierungen aus der Forschung der relationalen Datenbanken können bei dieser Art von Systemen gut angewendet werden, da hier alle Daten in verfügbaren Datenbanken vorliegen.

Ranking-Algorithmen basierend auf Provenance-Informationen und Datenanalyse (z.B. Information Extraction) können hier relativ einfach implementiert werden, da auf alle Daten lokal zugegriffen wird.

Vorteile und Nachteile

Die Vorteile von Systemen, die das „*crawling pattern*“ implementieren, zeigen sich vor allem in der Skalierbarkeit. Diese Systeme bieten eine vertretbare Performance, um komplexe Anfragen über eine große Anzahl von Datenquellen zu stellen. [6], Kapitel 6.3

Zudem könnten neue Datenquellen (durch Verfolgen von Links) zur Laufzeit des Crawlers gefunden werden und in den Datenbestand integriert werden. [6], Kapitel 6.3

Hartig und Langenegger erklären, dass der Datawarehousing-Ansatz der optimale Ansatz ist, um RDF Dumps in eine große Datenbank einzuspielen, um so die Daten der unterschiedlichen Datasets einfach kombiniert abzufragen zu können. [20], S. 5

Problematisch (und somit ein Nachteil) bei dieser Art von Systemen ist die redundante Speicherung der Daten. Das benötigt zum einen viele Hardware-Ressourcen zur Speicherung und Indexierung dieser Daten.Zum Anderen werden aber auch viele Ressourcenbei der Vorverarbeitung der Daten beim Crawling und Verarbeiten der Daten benötigt. [6], Kapitel 6.3[20], S. 5

Daten werden einmalig, regelmäßig oder auch unregelmäßig geholt. Dennoch werden die Daten nie so aktuell sein wie in der Datenquelle selbst. Aussagen könnten mittlerweile bereits „out-of-date“ oder schlicht falsch sein. [6], Kapitel 6.3

Beispiele

Beispiele für semantische Suchmaschinen sind Sindice[7], Falcons[8] oder SWSE[9]. Beispiele für Datawarehouse-Systeme sind RKBExplorer[10] und „Virtuoso hosted instance of LOD“[11].

3.3.2 On-the-fly dereferencing pattern

Bei diesem Ansatz werden URIs dereferenziert und Links in dem Moment verfolgt, wenn der Benutzer die gesuchten Informationen benötigt. [6], Kapitel 6.3

Görlitz und Staab beschreiben diesen Ansatz als „explorative query processing“.Bei diesem Ansatz startet der Nutzer oder die Nutzerin bei einer URI eines bestimmten Dataset. Es werden alle Links verfolgt, die dieseURI liefert. Das wird solange durchgeführt, bis keine weiteren Links mehr verfolgbar sind.[12], S. 113

[7]http://www.sindice.com [zuletzt angesehen 17.05.2011]

[8]http://iws.seu.edu.cn/services/falcons/ [zuletzt angesehen 17.05.2011]

[9]http://swse.deri.org/ [zuletzt angesehen 17.05.2011]

[10]http://www.rkbexplorer.com [zuletzt angesehen 17.05.2011]

[11]http://lod.openlinksw.com/ [zuletzt angesehen 17.05.2011]

Hartig et al verwenden in ihrem Ansatz für verteilte SPARQL-Queries ebenfalls das *On-the-fly dereferencing pattern*, um Datenquellen zu finden, die möglicherweise für die Beantwortung einer Anfrage relevant sind. Die gefundene (möglicherweise relevante) Datenquelle wird dann als Startpunkt für die SPARQL-Anfrage verwendet. [21]

Diskussion der Anforderungen

Für das Lesen der RDF-Informationen (Triples), die über eine URI ausgeliefert wird, kann die Abfragesprache SPARQL verwendet werden.
Werden die Daten über einen RDF-Parser gelesen, ist es notwendig, Informationen über das verwendete RDF Format zu haben.

Für das Lesen von RDF-Informationen einer URI ist es nicht notwendig, Informationen zum Mapping in einem Datenkatalog zu speichern. Werden aber mehrere URIs verfolgt und soll erst am Ende dem Benutzer/der Benutzerin die Informationen ausgeliefert werden, kann es sehr wohl nützlich sein, einen (zumindest temporären) Datenkatalog zu pflegen, der Mappings zwischen den URIs speichert.

Da hier lediglich RDF-Informationen zu einer Ressource abgefragt werden, ist es kaum möglich Query Optimierungen und Ranking-Algorithmen anzuwenden.

Vorteile und Nachteile

Dieser Ansatz ist sehr kostengünstig, da keine zusätzlichen Ressourcen zur Speicherung oder Indexierung benötigt werden. [12], S. 113

Ein Vorteil dieses Pattern ist, das die Abfrage der Daten immer auf einer aktuellen Datenbasis durchgeführt wird. Somit erhält der Benutzer/die Benutzerin keine möglicherweise abgelaufenen Daten.[6]
Jedoch kann bei diesem Ansatz keine Garantie gegeben werden, dass die gewünschte Information auch gefunden wird. Die Erfolgsaussichten hängen stark davon ab, ob ein guter „Einstiegspunkt“ in das Browsen des Graphs gefunden wurde. [12], S. 113

Problematisch bei Systemen, die diesen Ansatz verfolgen ist die Performance. Komplexere Operationen sind sehr langsam, da diese möglicherweise tausende URIs im Hintergrund dereferenzieren müssen, um an die gewünschte Information zu gelangen. [6]
Performance-Probleme könnten mit Caching- oder Crawling-Strategien gemindert werden. [20], S. 8

Das von Hartig et al. beschriebene Verfahren wird auch als „link traversal based query excecution“ bezeichnet und stellt trotz möglicher Performance-Probleme eine enorme Chance für das Abfragen von LD Quellen dar, da mit diesem Ansatz just-in-time neue Datenquellen gefunden werden können. [20], S. 7

Beispiele

Beispiele für diese Art von Systemen sind SQUIN[12], Tabulator[13] und Researchers Map[14].

3.3.3 Query federation pattern

Heath und Bizer beschreiben, dass bei diesem Ansatz komplette Anfragen oder Teile von Anfragen direkt an SPARQL-Interfaces von Datenquellen gesendet werden. [6], Kapitel 6.3

„Datasource federation“, wie dieses Pattern auch genannt wird, versucht die Vorteile von zentralen Ansätzen (Indexierung und Statistiken) und die Vorteile des On-the-fly dereferencing pattern (Verwendung der Originalquelle) in einem System zu vereinen. So werden zentral nur Metainformationen über die Datenquellen gespeichert. Die Suchanfrage (oder Teile davon) werden an die konfigurierten Datenquellen weitergeleitet und dort ausgeführt. Die Ergebnisse werden letztendlich zusammengeführt und dem Benutzer in einem/einer Ergebnis/Ergebnisliste präsentiert.[12], S. 114

Diese Architektur wird von Harth et al als distributed query processing approaches (DQP) bezeichnet und meint eigentlich das Query Federation Pattern. [18], S. 1
Sie definieren diesen Ansatz wie folgt:

> „[…] *parse, normalise and split the query into subqueries, determine the sources containing results for subqueries, and evaluate the subqueries against the sources directly.*“[18], S. 1

Das Programmstück, dass die Verwaltung der Queries und deren Abarbeitung übernimmt wird in der Literatur zumeist als Query Federator oder Mediator bezeichnet - vergleiche beispielsweise.[20], S. 5

[12]http://squin.sourceforge.net/ [zuletzt angesehen 17.05.2011]
[13]http://www.w3.org/2005/ajar/tab [zuletzt angesehen 17.05.2011]
[14]http://researchersmap.informatik.hu-berlin.de/ [zuletzt angesehen 17.05.2011]

Diskussion der Anforderungen

Systeme, die dieses Pattern implementieren, können nur auf Daten zugreifen, die SPARQL-Interfaces auf ihre Daten anbieten. Das Protokoll ermöglicht einen einheitlichen Zugang zu RDF-Daten – egal in welchem Format die Daten in den einzelnen Datasets serialisiert sind.

Der Datenkatalog kann zwei unterschiedliche Arten von Mappings enthalten. So können Beziehungen zwischen Ressourcen (z.B. durch sameAs oder seeAlso-Links) im Datenkatalog gespeichert werden. Auch ist es möglich zu speichern welche RDF Terme (oder ganze Graphen) in welchen Datasets enthalten sind.[12], S. 122

Mit dem ORDER_BY keyword ist es möglich SPARQL-Queries nach bestimmten Kriterien zu sortieren. Oftmals sind aber bestimmte Resultate generell relevanter als andere (unabhängig von gewählten ORDER_BY-Kriterium). Daher ist es nötig, Kriterien wie Antwortzeit, Datenqualität oder „Trust“ in einem Ranking-Algorithmus mit einzubeziehen. [12], S. 132

Vorteile und Nachteile

Ein Vorteil der Query Federation im Vergleich zu Systeme, die das crawling pattern implementieren, ist die Tatsache, dass keine zusätzlichen Ressourcen benötigt werden, da die Abfragen just-in-time über das Web gestellt werden. Somit ist keine Vorverarbeitung der Daten (wie das Crawling) notwendig. Auch werden keine Ressourcen zur Speicherung benötigt. [20], S. 5

Diese Vorverarbeitung der Daten bei MAT-Ansätzen bringen jedoch den großen Vorteil, dass diese Systeme in der Regel schnellere Antwortzeiten haben. Bei Query-Federation wird jedes Mal eine Anfrage übers Web gesendet, was nicht nur von der Verfügbarkeit und Antwortzeit der entfernten Interfaces abhängt, sondern auch von der Anzahl der Abfragen. [20], S. 5

Um in diesen Systemen auch bessere Antwortzeiten zu erzielen, müssen zusätzliche Techniken wie Caching oder Optimierungen von Algorithmen genutzt werden. [20], S. 5

Für die Optimierung der Algorithmen kann bereits auf Algorithmen von verteilten Datenbanken zurückgegriffen werden. In diesen Systemen ist es üblich, dass der Federator das Schema der entfernten Datenbanken im zentralen Datenbankkatalog vorliegen hat. Basierend auf diesen Informationen können dann die Queries lokal optimiert werden. Für LD Datasets ist das schwierig, da es keine solchen Schematas für Datasets gibt. Vielmehr kann die Art der

Beschreibung der Triples in einem Dataset sehr stark variieren. Ein weiteres Problem ist die Anzahl der Joins. Im schlimmsten Fall hat jedes Triple einen Link zu einem weiteren Dataset, das in der Abfrage berücksichtigt werden sollte. [20], S. 5

Problematisch bei diesem Ansatz ist auch, dass die Datenquellen, die für eine Anfrage in Frage kommen, von vorneherein feststehen. Es könnte sein, dass diese vorhandenen Datasets keine Antwort liefern, aber nicht bekannte Datasets könnten durchaus relevante Informationen enthalten könnten.
Auch muss der Benutzer Informationen über das verwendete Vokabular haben, um effektive Anfragen stellen zu können.

Beispiele
Beispiele für diese Art von Systemen sind SQUIN[15], DARQ[16] und LOQUS[17].

3.3.4 Auswahl der „richtigen“ Strategie

Die Vor- und Nachteile der jeweiligen Strategie wurden bereits diskutiert. Welche der Pattern für die Implementierung einer LD Applikation verwendet werden kann, hängt aber stark von den jeweiligen Anforderungen ab.

Mögliche Faktoren, die bei der Auswahl der Pattern berücksichtigt werden sollten, sind:

- Anzahl der relevanten Datasets
- Grad der Aktualität der Daten
- Antwortzeiten für Queries und User-Interaktionen
- Auffinden neuer Datenquellen zur Laufzeit

[6], Kapitel 6.3

Es ist zu erwarten, dass LD Applikationen kurz- und mittelfristig das „*Crawling pattern*“ verwenden werden, da derzeit noch nicht absehbar ist, wie mit „*On-the-fly link traversal pattern“* und „*Query federation pattern*“ akzeptierbare Antwortzeiten erzielt werden können. [6], Kapitel 6.3

[15] http://squin.sourceforge.net/ [zuletzt angesehen 17.05.2011]
[16] http://darq.sourceforge.net/ [zuletzt angesehen 17.05.2011]
[17] http://www.knoesis.org/faculty/pascal/resources/publications/loqus-tr-2010.pdf [zuletzt angesehen 17.05.2011]

3.4 Aspekte der Datenintegration

Zusätzlich zur Entscheidung einer Consuming-Strategie müssen noch diverse Aspekte der Datenintegration beachtet werden. Dieses Kapitel diskutiert ausgewählte Aspekte.

3.4.1 Mapping von Vokabularien

Zur Beschreibung von Ressourcen als LD stehen diverse Vokabularien und Ontologien zur Verfügung. Oftmals werden in unterschiedlichen Vokabularien (oder Ontologien) gleiche Konzepte beschrieben.
Für Abfragen über Datasets, die in unterschiedlichen Ontologien beschrieben sind, aber ähnliche Konzepte adressieren, sollte ein Mapping existieren, um Informationen über gleiche Dinge sofort zu erkennen.

Millard et al. meinen, dass sich mittelfristig die Verwendung bestimmter Vokabularien durchsetzen wird. Dennoch muss davon ausgegangen werden, dass es immer mehrere Wege geben wird, um allgemeine Konzepte wie Menschen, Orte, Organisationen, Themen und Klassifikationen zu beschreiben. [22], S. 3f

RDFS und OWL ermöglichen Mappings zwischen Ontologien zu beschreiben. Weiter existieren unterschiedliche Mapping-Sprachen wie SPARQL++[18], Alignment API[19], Rules Interchange Format[20] (RIF) und weitere, die es ermöglichen Mappings zwischen Vokabularien zu beschreiben.[6], Kapitel 6.3.2

3.4.2 Mapping von Ressourcen

Dadurch, dass es bei LD keine zentrale Stelle gibt, die die Publikation von Daten und Datasets überwacht, können in unterschiedlichen Datasets Aussagen über die gleiche Ressource veröffentlicht werden. Diese Informationen sind dann unter verschiedenen URIs auffindbar, können aber ähnliche oder sogar gleiche Informationen zu dieser Ressource enthalten. [6], Kapitel 2.5.2

Das Problem des mehrfachen Vorkommens unterschiedlicher URIs für dasselbe Konzept wird von Millard et al als „co-reference“ bezeichnet. Weiter nennen sie Gründe für dieses Vorgehen. Es gäbe keine globale URI, die Datenproduzenten verwenden könnten. Auch meint Millard et al, dass es für LD Anbieter sehr

[18]http://www.scharffe.fr/pub/sparql++-for-mapping-between-rdf-vocabularies-ODBASE-2007.pdf [zuletzt angesehen 17.05.2011]
[19]http://www.springerlink.com/content/dy8y9f31a9gt9762 [zuletzt angesehen 17.05.2011]
[20]http://www.w3.org/TR/rif-in-rdf/ [zuletzt angesehen 17.05.2011]

zeitaufwendig wäre, nach einer passenden bereits existierenden URI zu recherchieren. Es ist sehr viel einfacher selbst eine neue URI zu generieren, z.B. unter Verwendung von automatischen Export-Tools. Auch sei es so, dass gerade namhafte Institutionen (Behörden etc.) keine „fremden“ URIs verwenden möchten, sondern die Verwaltung dieser URI lieber in eigener Hand behalten. [22], S. 2
Ein Benutzer oder eine Benutzerin möchte innerhalb einer LD Applikation aber alle verfügbaren Informationen erhalten. Daher ist es notwendig, dass Links zwischen diesen URIs gesetzt werden, die dem Benutzer anzeigen, dass diese beiden URIs Aussagen zur gleichen Ressource treffen. Im Allgemeinen können hier sameAs-Links verwendet werden. [6], Kapitel 2.5.2

Die Verlinkung der Ressourcen im LD Web ist noch nicht sehr stark ausgeprägt (siehe auch Analyse in Kapitel 6.1.2 (ab Seite 76). Daher ist es notwendig allgemeine Services („co-reference resolution services“) anzubieten, die die LD Ressourcen analysieren und Links zu bereits existierenden LD Ressourcen anbieten. Solche Services können im Web angeboten werden wie sameAs.org und CultureGraph.org oder lokal unter Verwendung des SILK-Servers[21] oder ähnlicher Tools.

Viele der LD Anbieter haben zunächst viel Zeit und Kosten in die Bereitstellung „ihrer eigenen“ Daten gesteckt. Die Verlinkung zu ähnlichen Ressourcen anderer Anbieter sei eine „follow-up activity“. Daher sind viele Ressourcen untereinander noch nicht verlinkt obwohl es sich hier um „co-references“ handelt. [22], S. 2

3.4.3 Datenqualität

Um dem User in LD Applikationen „richtige“ Informationen liefern zu können, ist es unbedingt notwendig die Datasets in Bezug auf Datenqualität und somit Glaubwürdigkeit der Informationen einschätzen zu können.

Hartig beschreibt, dass die Glaubwürdigkeit einer Datenquelle von verschiedenen Einflüssen abhängen kann. So klassifiziert er folgende Kriterien:

- Qualität der Informationen („information quality“)
- Herkunft der Informationen („provenance“) und
- Andere („Other“) [23], S. 2

Informationen zur Herkunft der Daten sind jede Art/Form von Metadaten, die Informationen dazu enthalten, wie der aktuelle Zustand der Informationen entstanden ist:

[21] http://www4.wiwiss.fu-berlin.de/bizer/silk/server/ [zuletzt angesehen 17.05.2011]

> „[…] *everything that is related to how the object in its current state came to be*“. [23], S. 4

Informationsqualität umfasst jene Eigenschaften, die dazu beitragen, dass Daten als glaubwürdig eingeschätzt werden können:

> „*Knowing about a lack of correctness, accuracy, or consistency in the data reduces our belief in the truth of the information represented by that data. “*[23], S. 4

Weitere Glaubwürdigkeitskriterien könnte die Bewertung der Glaubwürdigkeit der Informationsquelle durch andere „Consumer“ sein – analog dem „**Web of trust“**-Prinzip.[23], S. 5

Heath und Bizer beschreiben einen möglichen Qualitätsindikator, der aus den drei Komponenten **content-based** heuristics, **context-based** heuristics und **rating-based** heuristics bestehen könnte. Wobei content-based heuristics sich auf die Analyse der Informationen und in Beziehung stehender Informationen beziehe, Context-based heuristics dagegen auf Metainformationen über die Informationen, beispielsweise wer wann eine Aussage getroffen hat. Rating-based heuristics bezieht sich auf Ratings der Informationen, des Datasets oder des Providers und kann vom Provider oder von externen Anbietern selbst stammen. [6], Kapitel 6.3.5

Hartig gibt darüber hinaus einen konkreten Überblick über mögliche Qualitätskriterien für LD Datasets, der sich aber nur auf die content- und context-based heuristics bezieht. Darin werden die vier Bereiche „Content“, „Representation“, „Usage“ und System identifiziert, wobei jeder der vier Bereiche noch genauer unterteilt wird.[24]

Weitere Kriterien für die Bewertung der Qualität eines Datasets (vor allem für Linked Closed Data) nennen Cobden et al. Kosten für den Zugriff (abhängig von Lizenzmodell) und Authentifizierungsmethoden.[25], S. 6

Weiter geben Cobden et al. zu bedenken, dass Informationsqualität und Provenance-Informationen kombiniert betrachtet werden müssen, da Provenance-Informationen Einfluss auf die Bewertung der Informationsqualität haben können. [25], S. 6

3.4.4 Lokale persistente Speicherung (Caching)

RDF-Stores

Aufgrund vielfach dargelegter Gründe sind Ansätze gleichzeitig in vielen SPARQL-Datasets zu suchen derzeit noch nicht performant genug, um diese Art der Suche in produktiven Systemen zu verwenden. Daher werden die Daten in einen lokalen RDF Store gespeichert. SPARQL-Anfragen können dann (z.B. im Batch) lokal durchgeführt werden. Beispiele für RDF-Stores sind: Sesame, SDB und Virtuoso.

Indexierungs-Arten

Für Query-Optimierungen oder Ranking-Algorithmen werden oft Datenstatistiken benötigt, die sich am Leichtesten mit Hilfe eines Index realisieren lassen. Görlitz und Staab unterscheiden verschiedene Arten der Indexierung:

- „Item Counts
- Fulltext Indexing
- Schema Level Indexing
- Structural Indexing“

[12], S. 122

„**Item Counts**“ enthalten Informationen darüber, wie viele Instanzen von Subjekt, Prädikat und Objekt und Kombinationen von diesen Elementen existieren. [12], S. 122

Bei „**Full text indexing**“ gehen Görlitz und Staab davon aus, dass ein RDF Graph analog zu einem Dokument zu sehen ist und somit mit Techniken des Information Retrieval indexiert werden kann, wie beispielsweise die Verwendung von Stoppwörtern oder Stemming. [12], S. 123

Bei „**Schema level indexing**“ werden die Instanzen bzw. Ressourcen indexiert. Problematisch bei dieser Art der Indexierung ist, dass Prädikate in unterschiedlichen Kontexten unterschiedliche verwendet werden.[12], S. 123

Bei dem Konzept von „**Structual Indexing**“ versucht man die Strukturen zwischen Datasets zu indexieren, um so bei joins in einer SPARQL-Abfrage schneller auf die Daten zugreifen zu können. [12], S. 123

Speicherungsarten

Görlitz und Staab unterscheiden auch unterschiedliche Arten der Speicherung von Indexen:

- „Data source index

- Virtual data source index
- Federation index
- Distributed federation index“

[12], S. 125

Als „**Data Source Index**“ werden jene Informationen benannt, die Daten zum Index des Datasets ausliefern, beispielsweise in Form von voiD-Beschreibungen. [12], S. 125

Sind diese Informationen nicht verfügbar, müssen diese Informationen von Query zu Query lokal selbst gebildet werden. Dieser wird als „**Virtual Data Source Index**“ bezeichnet. [12], S. 125

Ein „**Federation Index**“ ist ein zentraler Index, der Informationen zu allen bekannten Datenquellen sammelt und aktualisiert.[12], S. 125

„**Distributed Federation Index**“ ist ein verteilter federated index, der über alle Datenquellen verteilt ist. Analog zu Peer-to-peer Systemen bilden alle Komponenten gemeinsam diesen Index. Ziel dieser Art von Indexes ist die Steigerung der Performance. [12], S. 125

Typische Vertreter sind: Lucene Siren, Sesame, Jena SDB, Jena TDB.

3.5 Auswahl von Datasets

Die Auswahl der richtigen Datasets und somit das Finden der gesuchten Informationen ist eine elementare Anforderung an LD Applikationen. Für die Auswahl von Datasets wird zwischen „*dataset discovery“* und „*dataset selection“* unterschieden.

„Dataset discovery“ beschreibt das Auffinden von unbekannten Datasets im Vorhinein einer Query. Automatische Methoden, die Links verfolgen, werden derzeit am meisten verwendet, um neue Datasets zu finden. Momentan werden dabei folgende Methoden verwendet:

- Web crawling,
- Anfragen an Suchmaschinen durch APIs und
- Anfragen an Dataset Verzeichnissen wie CKAN[25] oder voiD-Services[22].

[25], S. 4f

[22]http://void.rkbexplorer.com[zuletzt angesehen 17.05.2011]

„Dataset selection“ beschreibt die Auswahl von Datasets aus einer Menge von potenziell relevanten Datasets. Zur Beurteilung der Qualität des Datasets können Metadaten des Datasets verwendet werden. Als Grundlage zur Messung der Qualität können unterschiedliche Beschreibungen des Datasets herangezogen werden: Semantic Sitemaps und Beschreibung in (voiD). Weitere Aspekte (zusätzlich zur Qualität) sind Relevanz, Kosten und Lizenzmodell. [25], S. 6

Cobden et al. beschreiben ein Modell zum „Consuming Linked Closed Data“[25], S. 6, das prinzipiell auch auf Linked Open Data angewendet werden kann. Abbildung 7(auf Seite 39) zeigt die Schritte dieses Modells.

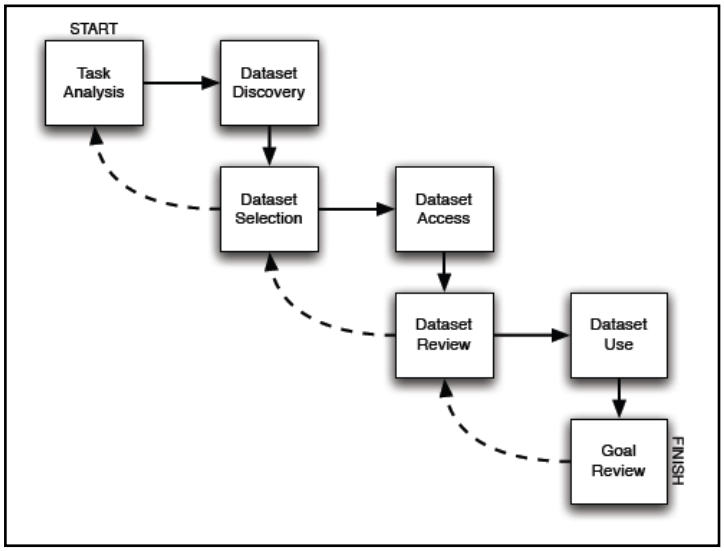

Abbildung 7: Prozess "Linked Data Consumption" [25], S. 8

In diesem Modell werden die einzelnen Schritte im logischen Ablauf gezeigt. Die Beurteilung der Qualität eines Datasets kann nur dann geschehen, wenn zu Beginn eine bestimmte Aufgabe steht. Basierend auf dieser Aufgabe können relevante Datasets gefunden („*Dataset disc*overy“) sowie die Relevanz dieser Datasets beurteilt („*Dataset Selection“*) werden. „*Dataset Access“* prüft den Zugriff auf die ausgewählten Datasets – das kann weitere Methoden zur Prüfung der Datenqualität des Datasets beinhalten. In der Phase des „*Dataset Use“* wird dann letztendlich die Query durchgeführt. Als letzter Schritt, in der Phase des Goal Review wird noch das Ergebnis der Query geprüft und überprüft, ob die Aufgabe durch die Abfrage dieses Datasets erfüllt wurde. [25], S. 7-10

Die Anzahl der publizierten Datasets wächst ständig und stellt Entwickler von LD Applikationen vor die große Herausforderung der Skalierbarkeit der Anwendungen. Dienste, die Daten mehrere Datasets aggregieren, können helfen mit der steigenden Anzahl von Dataset umzugehen. [25], S. 2

3.6 Ausgewählte Tools und Frameworks

Dieses Kapitel soll einen kleinen Überblick über Tools und Frameworks geben, die helfen können architektonische Pattern in einem System umzusetzen.

LDSpider ist ein erweiterbares LD Crawling framework, das es ermöglicht URI des LD Web zu verfolgen und die Daten zu „konsumieren". Dieses Framework kann mittels einer API in Java-Programmen verwendet werden, als auch als commandline-tool. LDSpider. unterstützt die meisten RDF-Serialisierungsformate wie RDF/XML, Turtle, N3, RDFa und weitere Mikroformate. Heruntergeladene Daten können in Dateien oder RDF-Stores gespeichert werden. Es werden unterschiedliche Crawling-Strategien unterstützt wie breadth-first-traversal und load-balancing wobei das Crawling selbst in einer multi-thread-Umgebung durchgeführt wird. [26]

Jenaist ein Java-Framework, das es ermöglicht Applikationen für das SW zu entwickeln. Dieses Framework stellt eine API zum Lesen, Schreiben und Verarbeiten von RDF-Daten zur Verfügung. Dabei werden diverse RDF-Serialisierungsformate unterstützt. Weiter werden OWL-Funktionalitäten für Reasoning unterstützt. Dieses Framewok unterstützt in-memory als auch persistente Speicherung (SDB[23] und TDB[24]) von RDF-Modellen. Zusätzlich wird eine SPARQL-Query-Engine unterstützt. [27]

trdf[25] ist eine Erweiterung des Jena-Frameworks zur Evaluierung der Datenqualität. tSPARQL[26] ermöglicht es in SPARQL-Abfragen Kriterien der Datenqualität (Trust) mit zu berücksichtigen.

Semantic Web Client Library[27] ermöglicht eine SPARQL-Abfrage unter Verwendung der Verfolgung von Links. Dieses Framework basiert auf Jena und ist als API und comandline Tool verfügbar. [28]

Ripple[28] ist eine Skriptsprache für LD, die es ermöglicht Anfragen als LD Quellen durch Verfolgung von URIs zu ermöglichen. Diese Abfragesprache ist für

[23] http://openjena.org/wiki/SDB [zuletzt angesehen 17.05.2011]
[24] http://openjena.org/wiki/TDB [zuletzt angesehen 17.05.2011]
[25] http://trdf.sourceforge.net/index.shtml [zuletzt angesehen 17.05.2011]
[26] http://trdf.sourceforge.net/tsparql.shtml [zuletzt angesehen 17.05.2011]
[27] http://www4.wiwiss.fu-berlin.de/bizer/ng4j/semwebclient/ [zuletzt angesehen 17.05.2011]
[28] http://code.google.com/p/ripple/ [zuletzt angesehen 17.05.2011]

Batchabfragen für LD Applikationen entwickelt worden und setzt auf dem Java-Framework Sesame auf.[29]

Einige RDF-Stores bieten zusätzlich die Möglichkeit der Indexierung an. Somit ist es z.B. mit Jena TDB möglich schnell auf die Daten des RDF-Stores zuzugreifen. Indexierungen können aber auch gesondert, beispielsweise über die Java-Suchmaschine Lucene gemacht werden. **Siren**[29], setzt auf Lucene auf und ermöglicht RDF-Daten zu indexieren. [30]

3.7 Zusammenfassung

In diesem Kapitel wurden verschiedene Consuming-Strategien diskutiert und für verschiedene Anwendungsfälle Empfehlungen für den Einsatz der jeweiligen Strategie gegeben:

- Crawling pattern
- On-the-fly linktraversal und
- Query federation

Abhängig von Anforderungen wie Performance, Zuverlässigkeit und Ressourcenbedarf kann die Empfehlung für eine Strategie oder eine Mischform mehrerer Strategien gegeben werden.

Zusätzlich zur Consuming-Strategie sind Aspekte der Datenintegration beim „Consuming“ zu beachten:

- Mapping von Ressourcen („co-reference mapping“)
- Mapping von Vokabularien
- (automatische) Dataset Discovery und Dataset Selection
- Bewertungskriterien zur Datenqualität

Ziel des Consuming von LD ist es schließlich diese Daten in Form von Applikationen den Benutzern in leicht bedienbarer Form anzubieten. Das nächste Kapitel analysiert Anforderungen an LD Webapplikationen zu diesem Zweck.

[29] http://siren.sindice.com/ [zuletzt angesehen 17.05.2011]

4 Linked Data Webapplikationen

Dieses Kapitel soll einen Überblick über die Eigenschaften von LD Webapplikationen geben. Dazu werden zunächst LD Webapplikationen klassifiziert. Anschließend werden Software-Architekturen dieser Applikationen analysiert und Anforderungen an Userinterfaces definiert.
Diese erarbeiteten Konzepte von LD Webapplikationen und die Konzepte von Consuming-Strategien aus dem vorherigen Kapitel dienen dann als Grundlage für die Fallstudien in Kapitel fünf.

Hausenblas definiert LD-driven Webapplikation als jene Webapplikationen, die als Datenquelle u.a. das „Web of data“ verwenden. So heißt es:

> “A linked data-driven Web application is a Web application that *consumes* and potentially *manipulates* linked data *on the Web*”. [15], S. 1

“Consume” bezieht sich hier nicht nur auf das Verwenden von Daten aus LD Datasets zur Bildung von Mashups, sondern auch auf Datenquellen, die vielleicht im Original gar nicht direkt als LD publiziert sind. Diverse Datenquellen können auch über LD-Wrapper und andere Tools on-the-fly in RDF-Daten konvertiert werden. [15], S. 1

„Manipulate“ beschreibt die Tatsache, dass nicht nur Daten gelesen werden können (beispielsweise in Form von Mashups), sondern auch geschrieben werden. [15], S. 1 (Anmerkung: beispielsweise in Form von SPARQL Updates)

4.1 Klassifikation

Es existiert eine Vielzahl von LD Applikationen, die verschiedenste Anforderungen erfüllen. Dieses Kapitel gibt einen ersten Überblick über verschiedene Arten von LD Applikationen und versucht diese zu klassifizieren.

Klassifikation nach Komplexität

Vom W3C wurden folgende Kategorien vorgeschlagen, um LD Applikationen entsprechend des Levels der Verwendung von LD einzuordnen:

- „Infrastructural level
- Low level access and manipulation LOD applications
- End user LOD applications“[31]

„Infrastructural level“ LD Applikationen sind jene „Applikationen“, die grundlegende Dienste des LD-Web bereitstellen: die Publikation von Datasets, die

Bereitstellung von Co-reference-Diensten, und Dienste von Aggregatoren, die Daten des LD Webs crawlen und die Daten aufbereitet publizieren.

„Low level access and manupulation"-Applikationen verwenden Daten des LD Web aktiv, beispielsweise zum Browsen in den Daten, Integration der Daten in Mashups und ähnlicher Dienste. Oftmals wird für diese Art der LD Applikationen Hintergrundwissen zu semantischen Technologien benötigt und sind somit nicht optimal für den Enduser geeignet.

„End user LOD applications" ermöglichen eine bestimmte Abstraktions-Schicht von den komplexen semantischen Technologien. Der Benutzer benötigt somit kein zusätzliches Wissen von semantischen Technologien – er oder sie arbeitet mit dieser Applikation wie mit anderen Webapplikationen.

Eine Liste von Applikationen, die bis 2008 veröffentlicht wurden, sind in einer Umfrage enthalten, die unter [32] aufgeführt ist. Weitere Applikationen sind im ESW LOD Wiki[31] zu finden.

Klassifikation nach Anwendungsbreite

Eine weitere Art der Klassifizierung wird von Heath und Bizer vorgestellt:

- „Generic applications
- Domain-specific applications"

[6], Kapitel 6.1

"**Generic applications**" werden dann noch weiter unterteilt in:

- „LD browsers und
- LD search engines"

[6], Kapitel 6.1.1

LD Browser

Analog zu normalen Webbrowsern, ermöglichen LD Browser RDF-Daten des „Web of Data" zu betrachten und über RDF Links zu navigieren. [33], S. 9ff

Eine Übersicht verfügbarer LD Browser ist [34] zu entnehmen. Prinzipiell werden zwischen serverseitigen und clientseitigen LD Browsern unterschieden. Serverseitige LD Browser sind jene Webapplikationen, welche die RDF-Daten aufbereiten und in HTML anzeigen – zumeist in Form einer Tabelle. Beispiele sind Disco[30], Sig.ma[31], Marbles[32] und Graphite[33]. Clientseite LD Browser sind Desktop

[30] http://www4.wiwiss.fu-berlin.de/bizer/ng4j/disco/ [zuletzt angesehen 17.05.2011]

[31] http://sig.ma/ [zuletzt angesehen 17.05.2011]

Applikationen oder Browser-Plugins. Beispiele sind: Tabulator[34], OpenDataLink Explorer[35] oder Fenfire[36].

LD Suchmaschinen

LD Suchmaschinen sind Systeme, die das LD Web crawlen durch Verfolgung von RDF- Links und Möglichkeiten zur Abfrage der aggregierten Daten bieten.[6], Kapitel 6.1.1

Analog zu klassischen Suchmaschinen bieten die semantischen Suchmaschinen Möglichkeiten zur Eingabe von Keywords, um eine Suche abzusetzen. Aber im Gegensatz zu den klassischen Suchmaschinen können diese Applikationen die Ergebnisse weitaus besser darstellen. Durch die Auswertung der strukturierten Daten können gezielt die gewünschten Informationen ausgegeben werden und nicht nur der Link zur Ressource, wo diese Information zu finden ist. [33], S. 11ff [6], Kapitel 6.1.1

Einige Applikationen, wie sig.ma (siehe auch Fallstudie dazu in Kapitel 5.2 auf Seite 63), ermöglichen sogar Benutzerinterkationen z.B. die Beschränkung der Suche auf bestimmte (qualitativ gute) Datenquellen. [6], Kapitel 6.1.1

Einige Applikationen, wie VisiNav, ermöglichen es sogar komplexere Suchanfragen zu stellen wie sie klassische Suchmaschinen wie Google derzeit nicht beantworten können – beispielsweise: „URLs aller Blogs, die von Menschen geschrieben wurden, die Tim Berners-Lee kennen“.[6], Kapitel 6.1.1

Beispiele für semantische Suchmaschinen sind Sig.ma[37], Falcons[38], SWSE[39] oder VisiNav[40].

Einige dieser Suchmaschinen bieten auch APIs an, so dass deren Funktionalitäten auch in LD Applikationen verwendet werden können. Somit

---

[32]http://wiki.dbpedia.org/Marbles?v=71e [zuletzt angesehen 17.05.2011]

[33]http://graphite.ecs.soton.ac.uk/browser/ [zuletzt angesehen 17.05.2011]

[34]http://dig.csail.mit.edu/2005/ajar/ajaw/About.html [zuletzt angesehen 17.05.2011]

[35]http://www.w3.org/wiki/OpenLinkDataExplorer [zuletzt angesehen 17.05.2011]

[36]http://events.linkeddata.org/ldow2008/papers/14-hastrup-cyganiak-browsing-with-fenfire.pdf [zuletzt angesehen 17.05.2011]

[37]http://sig.ma[zuletzt angesehen 17.05.2011]

[38]http://ws.nju.edu.cn/falcons/objectsearch/index.jsp [zuletzt angesehen 17.05.2011]

[39]http://swse.deri.org/ [zuletzt angesehen 17.05.2011]

[40]http://visinav.deri.org/ [zuletzt angesehen 17.05.2011]

könnten Applikationen diese Indizes abfragen statt selbst diese Daten zu crawlen.[33], S. 13

Domainspezifische LD Applikationen

Während LD Browser und LD Suchmaschinen eher allgemeine Funktionalitäten bereitstellen, verfolgen **domainspezifische LD Applikationen** zumeist nur die Anforderungen von bestimmten Benutzergruppen. [33], S. 13ff[6], Kapitel 6.1.2

Das nächste Kapitel untersucht Aufbau (Software-Architektur) domainspezifischer LD Webapplikationen detailliert und versucht die Frage zu klären, welche Komponenten LD Webapplikationen enthalten sollten.

4.2 Software-Architekturen

Dieses Kapitel soll einen Überblick schaffen, wie Software-Architekturen von LD Webapplikationen aufgebaut sind und somit Aspekte diskutieren, die bei der Implementierung von LD Webapplikationen zu beachten sind.

Der Bereich der semantischen Webapplikationen ist ein sehr komplexes Gebiet, da hier Konzepte und Architekturen der Webentwicklung und semantischer Technologien zusammentreffen. Der Bereich der Webapplikationen ist bereits hinreichend (aber wohl nie ausschöpfend) erforscht und bietet eine Reihe von Frameworks zur Entwicklung von (modernen) Webanwendungen.

Im Vergleich dazu fehlt es im SW noch an Frameworks um ähnlich leicht Webapplikationen mit semantischen Technologien implementieren zu können. Zur Entwicklung solcher Frameworks ist es notwendig die Anforderung an die Architektur solcher Applikationen zu kennen. So beschreiben Heitmann et al. etwa das Charakteristikum von semantischen Webapplikationen:

> „Semantic Web applications are decentralized and open, operate on distributed data that can be published anywhere, may conform to arbitrary vocabulary and follow semi-structured schemas.“[35], S. 2

Im Allgemeinen wird für die Architektur des *Semantic Web* immer der Protokollstack (siehe Abbildung 1 Seite 9) als Architektur herangezogen. Das reicht aus, um die Vielschichtigkeit des *Semantic Web* als Ganzes zu beschreiben, aber leider nicht, um Software-Architekturen von Applikationen des *Semantic Web* detailliert zu beschreiben.

Eine Software-Architektur besteht aus Komponenten, Schnittstellen und Daten, und beschreibt Muster wie die Komponenten auf sinnvolle Art miteinander kombiniert werden können. [35], S. 1

Zur Beschreibung einer Software-Architektur kann ein Referenzmodell dienen, welches die typischen Eigenschaften einer Gruppe von Applikationen beschreibt. Es werden dabei abstrakte Konzepte und Terminologien einer Softwarearchitektur beschrieben ohne jedoch Interfaces fix zu bestimmen. [35], S. 3

Im Folgenden werden nun drei Architektur-Modelle von Applikationen im Umfeld von *Semantic Web* und LD beschrieben und verglichen. Basierend darauf werden die wichtigsten architektonischen Aspekte zusammengefasst, die von LD Applikationen berücksichtigt werden sollen.

4.2.1 Referenz-Architektur für semantische Webapplikationen

Das Referenzmodell für semantische Webapplikationen von Heitmann et al. ist in Abbildung 8 (auf Seite 47) dargestellt und besteht grundlegend aus sieben Komponenten. Diese Komponenten sind wiederverwendbar und unabhängig von der Domain, wo die Anwendung eingesetzt werden soll. Diese Komponenten sind:

- „Data interface
- Persistence layer
- User interface
- Integration service
- Search engine
- Annotation user interface
- Crawler“

[35], S. 3f

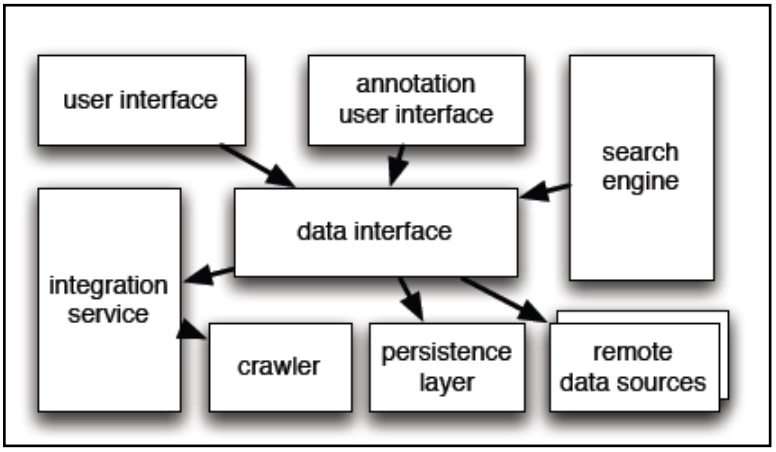

Abbildung 8: Referenz-Architektur nach Heitmann et al. [35]

Im Folgenden werden nun kurz die Komponenten beschrieben und dargestellt welchen Funktionsumfang diese Komponenten beinhalten *können*. Alle Komponenten sind optional, abhängig von den Anforderungen und Funktionalitäten die in der jeweiligen Applikation implementiert werden sollen. [35]

Das „**data interface**“ (auch „data adapter“ oder „access provider“) ist die zentrale Komponente, um von der Applikationslogik auf die lokalen oder entfernte Datenquellen zuzugreifen. [35], S. 3
Auf die lokalen Datenquellen kann durch eine API oder durch SPARQL zugegriffen werden. Der Zugriff auf entfernte Datenquellen wird durch SPARQL oder HTTP-Requests ermöglicht. Zusätzlich können Import- oder Export-Funktionalitäten implementiert werden. [35], S. 3

Das „**persistence layer**“ (auch „persistence store“ oder „triple store“) stellt einen Speicher für Daten und Laufzeitstatus der Applikation bereit. Auf diese Komponente wird durch das „data interface“ zugegriffen.[35], S. 3
Der Speicher selbst kann strukturierte, semi-strukturierte oder unstrukturierte Daten enthalten. Indexes ermöglichen einen schnelleren Zugriff auf die Daten oder die Struktur der Daten. Es werden unterschiedliche Formate unterstützt werden wie XML und RDF zur Repräsentation, OWL und RDFS zur Modellierung der Metadaten und SQL und SPARQL als Abfragesprachen. Zusätzlich könnten noch Möglichkeiten für „Inference“ oder „Reasoning“ unterstützt werden. [35], S. 3

Das „**user interface**“ (auch „portal interface“ oder „view“) stellt ein Interface zur Verwendung der Applikation und das Betrachten der Daten für den menschlichen Anwender dar. Das Modifizieren oder Erzeugen von Daten ist dabei nicht inkludiert – das übernimmt das „annotation interface“. [35], S. 3

Das „user interface“ kann als Webapplikation oder als Desktop-Applikation realisiert sein. Die Navigation in der Applikation kann entweder auf den Daten selbst, oder auf Metadaten basieren. Diese kann in Form eines dynamischen Menus oder als facettierte Navigation realisiert sein. Die Präsentation der Daten kann als einfache Tabelle oder in Form einer domainspezifischen Visualisierung (beispielsweise eine Karte für geografische Daten) angeboten werden. [35], S. 3

Das „**integration service**“ (auch „aggregation“, „mediation“ oder „extraction layer“) stellt eine einheitliche Sicht auf die Daten der unterschiedlichen Datenquellen bereit. Damit wird sichergestellt, dass auf die Daten unabhängig von Format oder Struktur einheitlich zugegriffen werden kann. [35], S. 3f
Unterschiedliche Schemata der einzelnen Datenquellen können automatisch oder manuell gemappt oder gruppiert werden. Aufgaben der Datenintegration können auf lokal gespeicherten Daten oder auf (in memory) Query-Resultaten basieren. Das „integration service“ kann zusätzlich Algorithmen zum Reasoning, Interference oder anderer domainspezifischer Logik enthalten. [35], S. 3f

Die „**search engine**“ (auch „query engine“ oder „query interface“) ermöglicht Suchanfragen an die Daten basierend auf dem Inhalt, der Struktur oder anderer domainspezifischer Funktionalitäten. Es werden Interfaces für den menschlichen Anwender als auch für Computerprogramme bereitgestellt. [35], S. 4
Als Recherche-Arten können sowohl Volltext- als auch Metasuchmöglichkeiten angeboten werden. Zudem sollten aber Wege bereitgestellt werden, die Suchanfrage im Nachhinein zu ändern oder die Ergebnisse einzuschränken. Die Art und die Anzahl der Suchergebnisse sollten dem Benutzer erklärt werden. Das Interface für Computerprogramme sollte die Ergebnisse in Form von SPARQL, Webservice oder REST bereitstellen. [35], S. 4

Der „**crawler**“ (auch „harvester“, „scutter“ oder „spider“) wird benötigt, wenn Daten erst aufgefunden und von externen Datenquellen geholt werden müssen, bevor die Daten integriert werden können. [35], S. 4

Das „**annotation user interface**“ ermöglicht Anwendern neue Daten einzugeben, bestehende Daten zu ändern und Daten zu importieren oder zu exportieren. Diese Komponente erweitert die Möglichkeiten des „user interface“ zum Ändern und Speichern von Daten. [35], S. 4
Für das Schreiben der Daten kann das Schema, der Inhalt oder die Struktur der Daten herangezogen werden. Hier könnten Möglichkeiten zum Editieren von RDF, RDFS, OWL oder XML geboten werden. Weiter wäre es auch möglich, leicht-strukturierten Text, beispielsweise in Form einer Wiki-Syntax, anzugeben.

Empfehlungsdienste zur Unterstützung der Eingabe könnten basierend auf Vokabularien oder der Struktur der Daten erstellt werden. [35], S. 3

4.2.2 Architektur für LD Webapplikationen

Da LD nur ein kleiner Teil des SW ist, werden in LD Applikationen auch nur Teile dieses Modells umgesetzt. Hausenblas beschreibt mögliche Komponenten einer LD Webapplikation – siehe Abbildung 9 (auf S. 49). Es werden zunächst die Komponenten dieses Modells näher erklärt und dann mit dem Modell von Heitmann verglichen.

Hausenblas‘ Modell enthält 6 Komponenten: User-Interface (UI), Logic, Data-Republisher, Data Integrator, RDF Store und Config. Die Komponenten UI und Logic sind typische Komponenten einer normalen Webapplikation und müssen hier nicht näher betrachtet werden. Die weiteren Komponenten v.a. „Data Integrator“, „RDF Store“ und „Data Republisher“ sind typisch für LD Webapplikationen. [15], S. 16

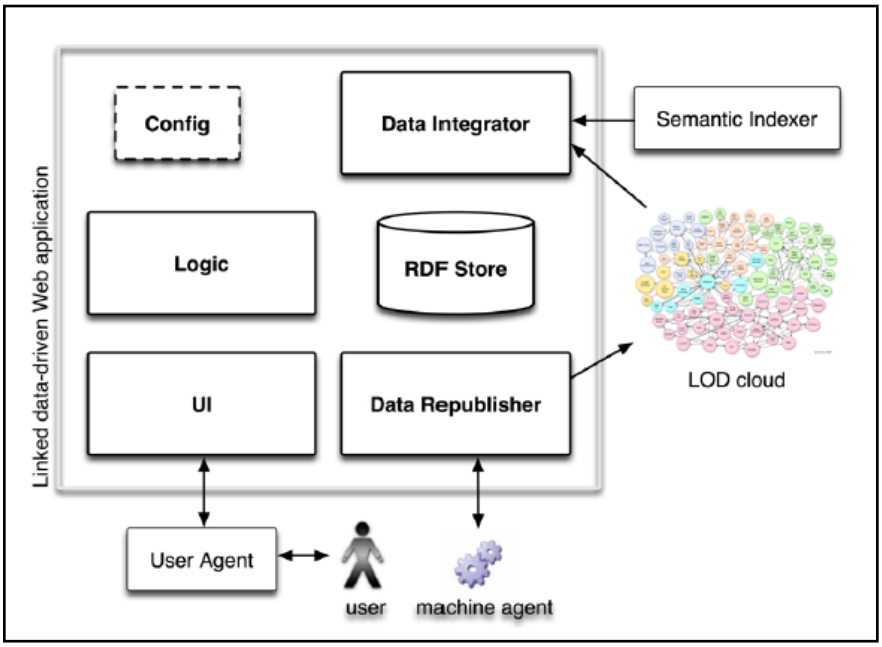

Abbildung 9: Architektur einer LD Webapplikation [15], S. 17

Die Komponente „**Data Integrator**“ stellt den Zugriff auf die Daten des LD Web durch semantische Indizes und Datasets der LOD Cloud bereit. [15], S. 16
Im Vergleich zu Heitmann’s Modell beinhaltet diese Komponente Funktionalitäten der „*data interface“*, „*data integration“* und „*crawler“*.

Der **RDF Store** wird verwendet, um Ergebnisse zwischenzuspeichern und so beispielsweise Benutzeraktionen zu verfolgen. Hausenblas weist darauf hin, dass es oft notwendig ist Daten lokal zu speichern, um so performante Services für den Benutzer anzubieten. ([15], S. 16)
Der RDF-Store ist vergleichbar mit Heitmann's *persistence layer*, da hier Daten lokal gespeichert werden.

Die Komponente **Data Republisher** ist eine Eigenheit von Applikationen im LD-Umfeld. Werden Daten von LD Datasets neu kombiniert (verlinkt) oder anderweitige zusätzliche Informationen gewonnen, sollten diese Daten wieder als LD publiziert werden. Diese Komponente übernimmt die Aufgabe der Bereitstellung der Daten. ([15], S. 16)
Diese Komponente wird in Heitmann's Modell nicht berücksichtigt.

Die von Heitman beschriebene „search engine" wird von Hausenblas nicht direkt genannt, könnte aber Teil des „*data integrator*s" oder Teil der „Logic"-Komponente sein.

Auch weist Hausenblas keine Komponente zum Modifizieren der Daten aus (in Heitmanns Modell das „annotation user interface"). Allerdings ist anzunehmen, dass Hausenblas die geholten Daten auch in irgendeiner Weise modifiziert oder zumindest aggregiert, da sonst die Komponente „*data republisher"* wohl keine Berechtigung hätte.
Als Beispiel führt Hausenblas eine Applikation an, die Inhalte eines Microblogging-Systems (Twitter) mit LD Datasets abgleicht und verlinkt, um damit Interessensgebiete von Personen zu ermitteln und wieder als LD zu publizieren. [15], S. 17f

Heitmanns „*user interface"* ermöglicht den Zugriff auf die Applikation für menschliche Anwender als auch für Computerprogramme. Bei Hausenblas wird das „User Interface" nur vom menschlichen Anwender verwendet. Computerprogramme können auf die Daten durch die „*data republisher"* Komponente zugreifen.

4.2.3 Architektur für LD Applikationen (crawling pattern)

Heath et al. [6]beschreiben schließlich noch eine Architektur für LD Applikationen, die das crawling-pattern implementieren – siehe Abbildung 10 (auf Seite 51).

Dieser Architektur-Ansatz ist in Schichten aufgebaut und hat einen starken Fokus auf die Datenintegration.

Die unterste Schicht, das „**Publication Layer**“, beschreibt die unterschiedlichsten Formate, wie Daten für das LD Web bereitgestellt werden: RDF-Files, RDF-Inhalte in Dateien oder durch LD Wrapper veröffentlichte Daten. [6], Kapitel 6.3

Die nächste Schicht, das „**Web of Data“**, beschreibt die Verlinkungen im LD Web. Diese Daten sollen in LD Webapplikationen verwendet werden. [6], Kapitel 6.3

Die dritte Schicht „**Data Access, Integration and Storage Layer**“ umfasst die Problematik der Datenintegration, die auch schon in den zwei zuvor beschriebenen Modellen eine wichtige Rolle spielt. [6], Kapitel 6.3

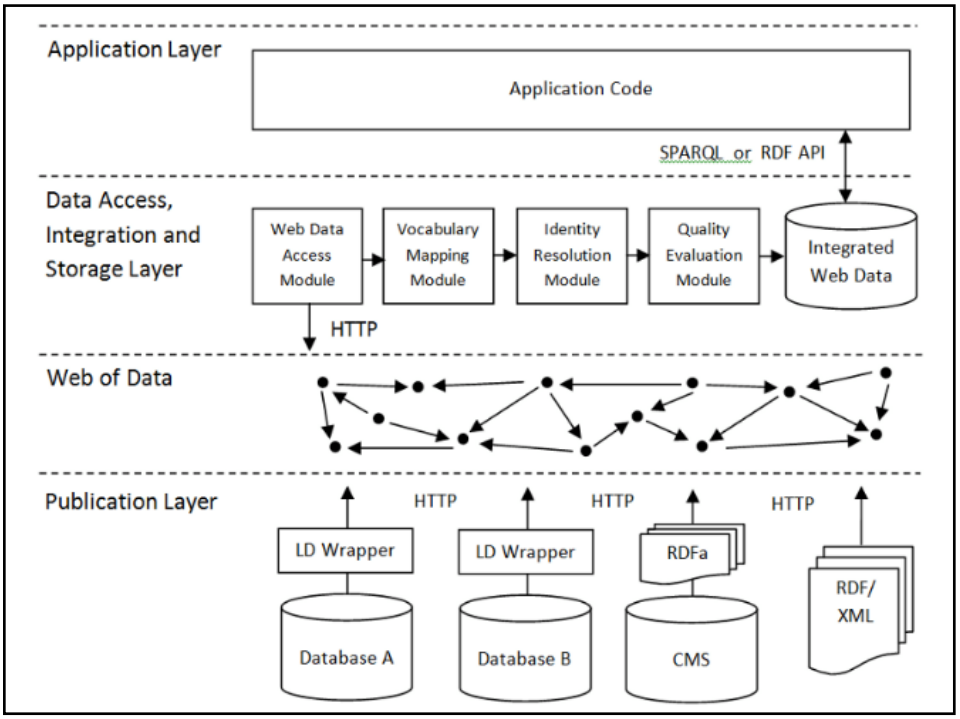

Abbildung 10: Architektur einer LD Applikation (mit crawling pattern)

Heath et al. definieren den Workflow der Datenintegration noch genauer. So werden folgende Komponenten durchlaufen:

- „Web Data Access Module
- Vocabulary Mapping Module
- Identity Resolution Module
- Quality Evaluation Module
- Integrated Web Data“[6], Kapitel 6.3

Das „*Web Data Access Module“* enthält Funktionalitäten zum Dereferenzieren der HTTP URIs in RDF Beschreibungen und zum Verfolgen von darin enthaltenen

Links. Auch die Möglichkeiten von semantischen Suchmaschinen oder SPARQL werden hier genutzt. [6], Kapitel 6.3

Das „*Vocabulary Mapping Module“* mappt die RDF-Daten, die in verschiedenen Vokabularien bereitgestellt werden in eine einheitliches Vokabular.[6], Kapitel 6.3

Das „*Identity Resolution Module“* analysiert die RDF-Daten der Beschreibung der Ressourcen auf co-reference-Beziehungen.[6], Kapitel 6.3

Das “*Quality Evalution Module”* enthält die Teil-Module “*Provenance Tracking”* und “*Data Quality Assessment”*. Provenance-Informationen und deren Auswertung sind für Systeme, die das „*Crawling pattern*“ implementieren sehr wichtig, da darüber bestimmt werden kann, welche Informationen aktualisiert werden müssen. Die Auswertung der Daten-Qualität wird dann besonders relevant, wenn Daten aus vielen unterschiedlichen Datenquellen integriert werden. [6] Die Datenqualität könnte ein Kriterium für das Ranking der Daten sein.

Die Daten werden dann lokal gespeichert (vergleichbar mit den RDF Store von Hausenblas), um so einen performanten Zugriff auf die Daten durch die nächste Schicht, dem Application Layer, zu ermöglichen.

Die **Applikations-Komponente** kann die bereitgestellten Daten in unterschiedlichen Formaten anzeigen, beispielsweise in Tabellen, Diagrammen oder anderen interaktiven Visualisierungsformen. Es ist aber möglich, dass die Applikationskomponente eine zusätzliche Logik hat, um Daten zu aggregieren oder neue Informationen aus der Datenbasis zu generieren, das passiert durch Mining-Methoden oder Reasoning. [6]

4.2.4 Schlussfolgerungen

Nach der Vorstellung der drei Modelle in den vorangegangen Kapiteln werden diese abschließend gegenübergestellt und folgende Empfehlungen gegeben.

Eine ausreichende Abstraktion der einzelnen Applikationsbestandteile, insbesondere bei der Datenintegration als auch bei der Verarbeitung der Daten ist notwendig, um sie sowohl für menschliche Anwender (in Form von Webapplikationen) als auch für Computerprogramme (in Form von APIs) anbieten zu können.

Werden Daten neu aggregiert oder gar angereichert ist es empfehlenswert, diese Daten auch wieder als neues Dataset im LD Web zu veröffentlichen.

Unabhängig davon, wie auf die Daten zugegriffen wird (remote oder -gecrawled) werden Methoden zur Datenintegration benötigt. Dazu gehören das Mapping von Ressourcen (co-reference), das Mapping von Vokabularien (auf ein einheitliches Format) als auch die Auswertung von Provenance und Qualitätsaspekten beispielsweise für ein Ranking-Verfahren.

Werden die Daten nicht nur abgefragt und visualisiert, sondern mit logischen Schlussfolgerungen (Reasoning und Inference) weiterverarbeitet, kann das eine eigene Komponente (Logik) sein, die das User-Interface verwendet oder aber auch Teil der Datenintegration sein.

Zur Bewertung von LD Webapplikationen reicht es nicht aus Software-Architekturen zu beschreiben. Auch müssen die User-Interface Anforderungen bekannt sein. Daher wird im nächsten Kapitel auf spezielle Anforderungen an das Userinterface von LD Webapplikationen eingegangen.

4.3 User-Interface Anforderungen

Um Informationen im klassischen Web aufzufinden, wird zumeist mithilfe keyword-basierten Suchmaschinen recherchiert. Die Ergebnisse, Links, werden dann nach dem trial-and-error-Prinzip verfolgt und angezeigt. Der Benutzer muss prüfen, ob dieses Ergebnis tatsächlich seinen Suchanforderungen entspricht. Falls nicht, muss das nächste Ergebnis geprüft oder die Suchanfrage geändert werden. [36], S. 3

Die Erwartungen der Anwender verlagern sich allmählich von einer reinen keyword-basierten Suche zu eher explorativen Recherche-Ansätzen, bei der Methoden des Lernens und der Erkundung von Wissensgebieten miteinbezogen werden.[36], S. 3

4.3.1 Exploratory Search

Die keyword-basierte Suche, die meist angeboten wird, ermöglicht Anwendern nur nach Begriffen zu suchen, die bereits bekannt sind. Dieses Prinzip wird als *lookup* bezeichnet. [36], S. 3

„*Exploratory search*“(explorative Suche) geht noch einen Schritt weiter und möchte Anwendern Methoden des „*Learning*“ und der „*Investigation*“ bereitstellen. Somit wird dem Benutzer ermöglicht, alternative Recherche-Pfade einzugehen, oder aber auch einen Schritt vor und wieder zurück zu machen. Es werden

beispielsweise verwandte oder untergeordnete Begriffe des Suchterms dem Benutzer bei der Recherche angeboten. [36], S. 3

Für das Navigieren in Ergebnislisten des klassischen *lookup*-Ansatzes wird bereits häufig die „**facetted search“ (facettierte Suche)** angeboten. Damit kann ein Anwender leichter durch eine Ergebnisliste navigieren, da über bestimmte Filter die Ergebnismenge eingeschränkt wird. ([36], S. 3) [37]
Mit der facettierten Suche findet der Anwender jedoch keine neuen Informationen, die nicht schon in der Ergebnismenge der keyword-basierten Suche vorhanden sind.[36], S. 3

LD schafft nun die Voraussetzungen zur Implementierung einer explorativen Suche. Dank der typisierten Verlinkungen können Beziehungen zwischen Informationen hergestellt werden. Durch Auswertung dieser Beziehungen können den Anwendern verwandte oder untergeordnete Themen des eingegebenen Suchbegriffs präsentiert werden.

Das Ergebnis der explorativen Suche, also die Auswertung der Beziehungen, kann mehrdimensional sein. Daher müssen hier neue Visualisierungstechniken angewendet werden, die dem Anwender dann eine geeignete Interaktion in diesem Ergebnis ermöglicht. [36], S. 3
Konkrete Techniken werden im Abschnitt 4.3.2 aufgezeigt.

Die Möglichkeiten für neue Funktionalitäten in Webapplikationen zur Umsetzung der explorativen Suche scheinen unbegrenzt. Jedoch müssen die Anforderungen der Benutzer an solche Systeme immer berücksichtigt werden.

Wilson et al. beschreiben unterschiedliche Voraussetzungen, die Benutzer haben können:

- Sie sind nicht vertraut mit der Domain und derer Terminologie.
- Sie sind nicht vertraut mit dem System und dessen Möglichkeiten.
- Sie sind nicht vertraut mit dem zu erwartenden Ergebnis.

[38], S. 10

Weiter heißt es, dass Experten weitere Anforderungen an Recherchesysteme haben könnten:

- Umfangreiche Suchanfragen, um alle relevanten Datensätze zu finden.
- Negative Suchanfragen, um zu prüfen, ob es wirklich keine relevanten Informationen in einem Bereich gibt.
- Suche von Ausnahmen, um einen Bereich aus einem anderen Blickwinkel zu betrachten.

- Übergreifende Suchen, um zwei Bereiche miteinander zu verbinden.

[38], S. 10

Um diese unterschiedlichen Anforderungen der Anwender an die Recherchesysteme berücksichtigen zu können, wird empfohlen die explorative Suche zu personalisieren. Das heißt, Anwender müssen die Systeme während der Nutzung an ihre Bedürfnisse anpassen können, beispielsweise bestimmte Funktionalitäten oder Datenquellen aus- oder einschalten. [36], S. 6[39]

Wilson et al. fassen abschließend die Herausforderungen bei der Implementierung der explorativen Suche zusammen:

> "*Exploratory search scenarios are characterized by needs that are open-ended, persistent, and multifaceted, and information-seeking processes that are opportunistic, iterative, and multitactical*"' [38], S. 10

4.3.2 Usability

Der Erfolg der *explorativen Suche* und LD hängt sehr stark davon ab, ob es gelingen kann zugängliche und bedienbare Interfaces zur Visualisierung der Ergebnisse und der Beziehungen zwischen Entitäten zu entwickeln. [36], S. 6

Um das zu erreichen sollten solche Applikationen:

- alle relevanten Informationen anzeigen,
- zusammenhängende Informationen aggregieren und
- vernachlässigbare Informationen verbergen

[36], S. 6

Ein wichtiger Punkt bei der Implementierung der explorativen Suche, mithilfe von LD, ist die Interaktion des Benutzers mit dem System. So sollte ein Benutzer durch Suchpfade navigieren können und auf die angezeigten Ergebnisse Einfluss nehmen können. Das System sollte entsprechend der Suchgewohnheiten anpassbar sein.

Benutzeranfragen

Anwender wollen nicht nur reine keyword-basierte Suchanfragen stellen, bei der textuelle Ergebnislisten erhalten. Oftmals besteht die Suche aus einer Vielzahl von Strategien, um das Ziel der Suche zu erreichen – beispielsweise durch Vergleiche, Schlussfolgerungen oder Evaluierungen. [38], S. 10

Daher sollten Suchanfragen und Ergebnisse durch einen “history”-Mechanismus oder die Möglichkeit zum Taggen von (Teil-)Ergebnissen angeboten werden. [38], S. 10
Das ermöglicht Benutzern später zu diesen Suchpfaden oder Ergebnissen zurückzukehren.

Hardman et al. identifizieren zwei Arten der Suche, die mit LD realisiert werden können: „*comprehensive search*“ und „*topic search*“. Die „***comprehensive search***“ ist jene Suchanfrage über die Unterschiede oder Gemeinsamkeiten von Objekten ermittelt werden sollen. Die „**topic search**“ ist jene Suchanfrage, die alle möglichen Informationen zu einem Thema ermitteln soll. [40], S. 2

Bei klassischen Suchmaschinen ist für beide Arten der Suche eine Vielzahl von keyword-basierten Suchanfragen notwendig.

Waitelonis et al geben zu bedenken, dass es dem Benutzer auch ermöglicht werden muss, komplexe Anfragen mithilfe intuitiv-bedienbarer Interfaces zu stellen. [36], S. 6
Dafür könnten SPARQL-Builder wie iSPARQL[41] oder auch Systeme wie Explorator[42] als Vorbild dienen.

Benutzer werden in modernen Webapplikationen bereits bei der Eingabe ihrer Suchanfragen unterstützt, beispielsweise durch eine Autosuggestion-Funktion, die dem Benutzer während der Eingabe Vorschläge für die Suchanfrage macht. Bei vielen Systemen basiert diese Funktion auf der Statistik der häufigsten Suchanfragen.
Diese Autosuggestion-Funktion könnte auch mit einem Thesaurus (oder anderen semantischen Modellen) hinterlegt werden und so den Kontext der Suchanfrage mit einbeziehen. Bei einer Suche nach „Java“ kann dann ein Benutzer auswählen, ob er die Insel, die Kaffeesorte oder die Programmiersprache meinte.

Visualisierung der Ergebnisse

Typische Visualisierungs-Beispiele, die aus einer Vielzahl von modernen Webapplikationen bekannt sind, sind beispielsweise Tagclouds (für die Visualisierung von Begriffen), Maps (für die Visualisierung von geografischen Orten), Timelines (für die Visualisierung von Zeitpunkten) und viele mehr.

[41] http://demo.openlinksw.com/isparql [zuletzt angesehen 17.05.2011]
[42] http://www.tecweb.inf.puc-rio.br/explorator/demo[zuletzt angesehen 17.05.2011]

Eine Kombination von Visualisierungsformen, wie sie für die explorative Suche möglich ist, zeigt Abbildung11, ein Screenshot der Openlibrary[43]. Hier wird eine Suche und die Navigation in allen frei verfügbaren E-Books angeboten.

Grafische Visualisierungen eigenen sich nicht für jede Art der Informationen einer Ergebnismenge. Bestimmte Daten müssen nachwievor in textueller Form angezeigt werden. Für solche Arten von Informationen sollte eine facettierte Navigation ermöglicht werden, wie sie in Kapitel 4.3.1 beschrieben wurde.

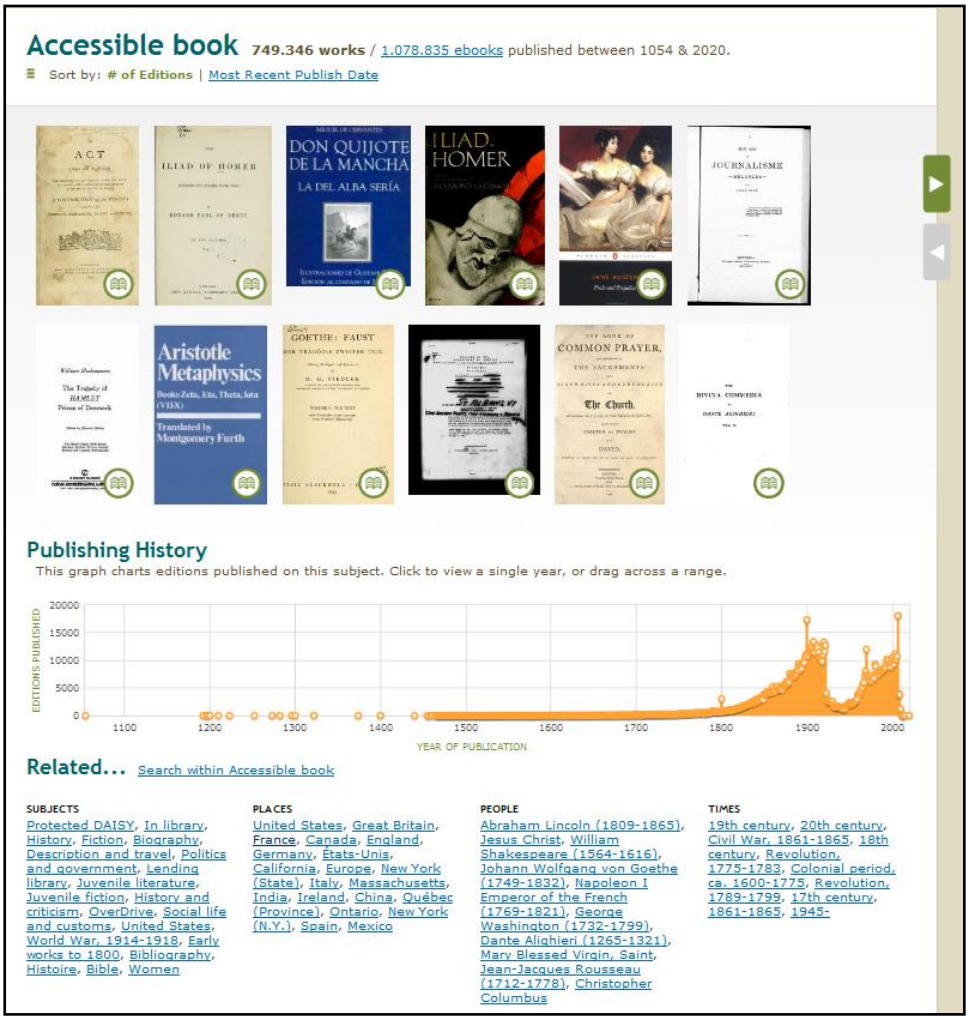

Abbildung11: exploratory search von openlibrary.org

Datenqualität

Intuitiv bedienbare Visualisierungen werden nur dann möglich, wenn die Daten der Provider auch entsprechend „reichhaltig“ sind. Waitelonis et al. weisen darauf hin, dass es wichtig ist Methoden zur Verbesserung der Datenqualität,

[43]http://openlibrary.org/subjects/accessible_book[zuletzt angesehen 17.05.2011]

Vollständigkeit und der Reichhaltigkeit semantischer Informationen zu implementieren. Dazu müssten sehr große Knowledgebases eingerichtet werden, die diese Aufgaben erledigen können. [36], S. 6

Die fehlende Datenqualität und Genauigkeit der Daten im LD Web ist insbesondere dann ein Problem, wenn Daten aus mehreren Datasets aggregiert angezeigt werden. [36], S. 6

Eine Möglichkeit ist, Benutzer bestimmen zulassen, welche Datasets für sie am relevantesten sind. [41], S, 4
Diese Methodik wird beispielsweise bei sig.ma (Siehe Fallstudie Kapitel 5.2) verwendet.

Performance

Araújo und Schwabe weisen darauf hin, dass die Performance (hauptsächlich der Antwortzeit einer Abfrage) einen starken Einfluss auf User Experience hat. [42], S. 9

Sollten in der Applikation entfernte Datenquellen (mit einer langsamen Antwortzeit) verwendet werden, empfiehlt es sich, Caching-Mechanismen zu implementieren.

5 Fallstudien

In diesem Kapitel sollen nun anhand von Fallstudien drei LD Webapplikationen genauer untersucht werden. Dazu werden diese drei Applikationen in Bezug auf die verwendeten Software-Architekturen und Elemente der explorativen Suche genauer untersucht.

Für die detailliertere Betrachtung wurden Applikationen ausgewählt, die Daten aus mehreren Datenquellen bei der Recherche berücksichtigen. Folgende Applikationen kommen in Betrachtung: „LED“, „sig.ma“ und „RKBExplorer“.

5.1 LED

LED (Akronym für Lookup Explore Discovery) ist eine LD Webapplikation, die Benutzern einen besseren Zugang zur Recherche in klassischen Suchmaschinen (Google, Yahoo, Bing) ermöglichen soll.

Verwandte Themen zur Suchanfrage werden dem Benutzer in Form einer Tagcloud angeboten. Das ermöglicht Benutzern, neue Themen schneller zu erschließen als es in klassischen Ergebnislisten der Suchmaschinen Google, Yahoo und Bing der Fall ist.

Möchte ein Benutzer einen Überblick zu einem Thema erhalten, muss beim Browsen durch die klassischen Ergebnislisten nach dem trial-and-error-Prinzip vorgegangen werden. Die Visualisierung untergeordneter Themen (beispielsweise in Form einer Tagcloud) ermöglicht gezielter das Thema zu erschließen, indem der Benutzer oder die Benutzerin Unterthemen recherchieren kann. [43], S. 4

Grundlage für die Erzeugung der Tagcloud sind LD Datasets, wobei die Tags Ressourcen des „Web of Data“ (derzeit nur des Datasets DBPedia) repräsentieren. Jede Ressource (und somit jeder Tag) kann über eine URI nachgeschlagen werden und bildet somit die Grundlage für weitere Informationen (beispielsweise für die Ermittlung weiterer Unterthemen). [43], S. 1

5.1.1 Userinterface

Abbildung 12 (auf Seite 60) zeigt einen Screenshot der LED-Applikation. Das UI besteht vertikal aus drei Bereichen. Im oberen Bereich, im Suchfeld, können Suchanfragen eingegeben werden, wobei die Eingabe durch eine Autosuggestion-Funktion unterstützt wird. In der Autosuggestion-Funktion werden

DBPedia[44] Ressourcen angezeigt. Wird ein Eintrag der Autosuggestion-Funktion ausgewählt, wird im mittleren Bereich die Tagcloud generiert.

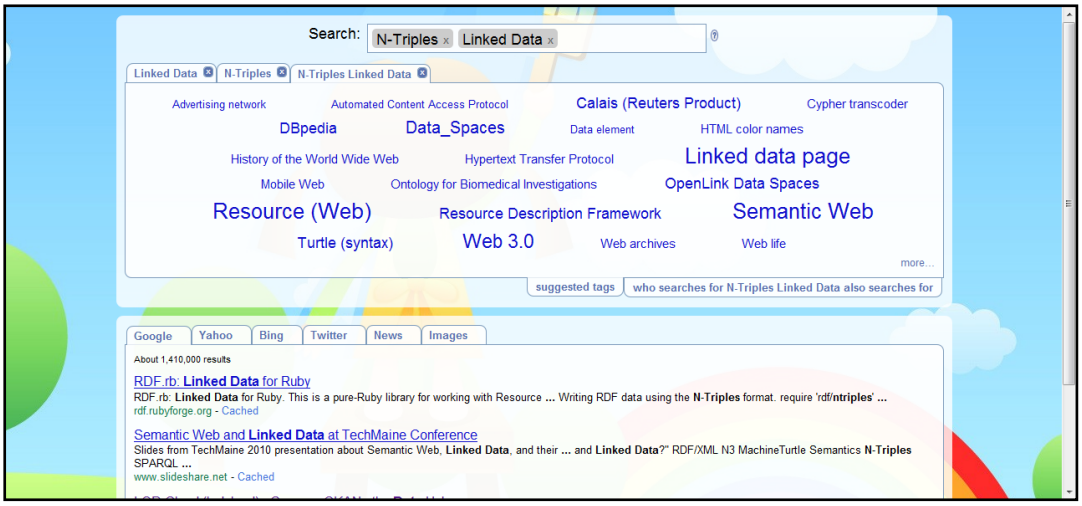

Abbildung 12: LED Screenshot

Der mittlere Bereich, die Tagcloud, zeigt verwandte Themen zur ausgewählten Ressource. Für jeden Tag wird ein Tooltip dargestellt, der eine Beschreibung der entsprechenden Ressource anzeigt. Neben dem Tag wird ein Pluszeichen angezeigt. Das ermöglicht dem Benutzer oder der Benutzerin die ursprüngliche Suchanfrage durch diesen Begriff zu erweitern. Auch ist es möglich Tags durch Ziehen mit der Maus in das Feld der Suchanfrage hinzuzufügen.
Für jede neue Suchanfrage wird die Tagcloud in einem separaten Tab angezeigt und ermöglicht dem Benutzer, jederzeit auf vorherige Suchanfragen zurückzukehren.

Der untere Bereich ist eine sogenannte Meta-Suchmaschine. Diese sucht in diversen Suchmaschinen (z.B. Google, Yahoo und Bing) nach der eingegebenen Suchanfrage und zeigt die Resultate in separaten Tabs an.

Die drei Bereiche repräsentieren das Akronym LED. „L“ steht für Lookup meint die Suchbox im oberen Teil der Applikation (inklusive Autosuggestion-Funktion). „D“ steht für discover und bezeichnet die Tagcloud zum Browsen in verwandten Themen. „E“ steht für explore und ermöglicht hier das Auffinden von Informationen in unterschiedlichen Suchmaschinen. [43], S. 2

[44]DBPedia: semantische Form der Wikipedia. Jede Wikipedia-Seite stellt ein Konzept und somit eine Ressource dar die als LD publiziert wird.

5.1.2 Architektur und Workflow

Abbildung 13 zeigt die grundlegende Software-Architektur von LED, die prinzipiell aus dem Backend (dem sog. „DBPedia Ranker“), dem LED-Interface und externen APIs besteht.

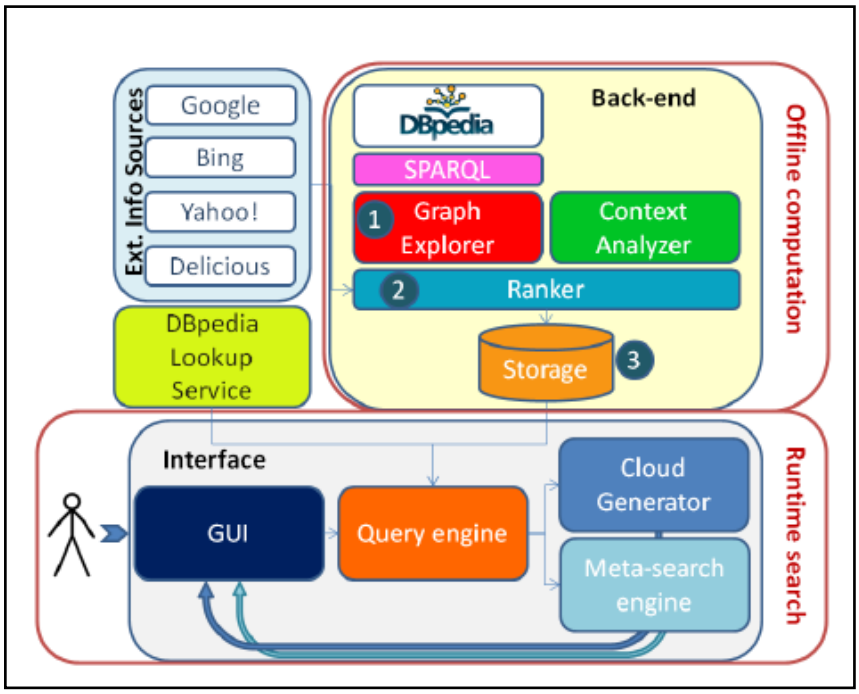

Abbildung 13: LED Architektur [43]

Die Daten für die Generierung der Cloud werden vorab (offline) im Backend aufbereitet. Diese Aufbereitung durchläuft die drei Komponenten „Graph Explorer“, „Context Analyzer“ und „Ranker“.

Zunächst werden im **Graph Explorer** die Ressourcen über das SPARQL-Interface der DBPedia abgefragt und skos:subject[45]- oder skos:broader[46] Links rekursiv weiterverfolgt. Alle Links im Abstand von zwei gelten als in Beziehung-stehend zur ursprünglich untersuchten URI (Ressource).[43], S. 5

Im „**Context Analyzer“** wird der Context der DBPedia URI genauer analysiert. Dazu werden die Wikipedia-Kategorien verwendet. Wird eine neue URI hinzugefügt, wird diese gegen alle schon existierenden URIs dieses Contexts überprüft. [43], S. 6f

[45]Property zur Beschreibung eines Schlagworts

[46] Property zur Beschreibung eines Oberbegriffs

Der „**Ranker“** ermittelt schließlich Ähnlichkeiten zwischen den gefundenen URIs und gewichtet diese so, dass die URIs gereiht werden können. Zur Ermittlung der Gewichtung werden zwei unterschiedliche Arten von externen Quellen verwendet:

- Suchmaschinen Google, Yahoo und Bing
- Social-Tagging-Systeme wie Delicious[43], S. 5f

Durch das Abfragen der Suchmaschinen kann die „popularity“ der Ressource ermittelt werden. Dazu wird die URI in Kombination mit dem keyword als Suchanfrage an die Suchmaschine gestellt. Somit kann festgestellt werden, wie viele Webseiten diese URI verlinken. Weiter werden zwei URIs in Kombination untereinander und mit keyword gesucht. Dadurch kann festgestellt werden, wie eng diese URIs in Beziehung stehen.[43], S. 6
Zusätzlich zur „popularity“ von Webseiten, soll auch die „popularity“ seitens der User berücksichtigt werden. Daher werden Social-Tagging-Systeme wie Delicious zusätzlich herangezogen. [43], S. 6
Zu Ermittlung der Ähnlichkeit werden die (DBPedia) Ressourcen nochmal genauer analysiert, insbesondere, ob die Ressourcen sich gegenseitig verlinken (dbprop:wikilink[47]) oder beispielsweise ob im Abstract der einen Ressource die andere Ressource genannt wird. [43], S. 6

Ist die Analyse abgeschlossen, werden Paare von Ressourcen im **Storage**, einer MySQL-Datenbank gespeichert.[43], S. 7

5.1.3 Schlussfolgerungen

LED verwendet die Consuming-Strategie „crawling pattern“ zur Kalkulation der Tagcloud, da die Berechnung der semantischen Ähnlichkeiten zur Laufzeit nicht möglich ist.
Für die Analyse von 8596 DBPedia-Ressourcen benötigte eines Rechner (2,6GHz quad-core, 4GB RAM) zwei Wochen. [43], S. 7

Bezüglich der Software-Architektur werden hier Teile der von Hartig et al. beschriebenen Architektur (siehe Kapitel 4.2.3 ab Seite 50) angewendet werden. Allerdings sind hier nur bestimmte Modelle der Datenintegrations-Schicht notwendig, da bisher nur eine Datenquelle integriert wird.

Das „Web Data Access Module“ wird in LED durch SPARQL-Schnittstellen des „Graph Explorer“ repräsentiert. Das „Vocabulary Mapping Module“ wird nicht

[47] Property zur Beschreibung eines Wikipedia-Links

verwendet, da für die Beschreibung aller DBPedia-Ressourcen das gleiche Vokabular verwendet wird. Aspekte des „Identity Resolution Module“ und „Quality Evaluation Module“ werden durch den „Context Analyzer“ repräsentiert.

Bezüglich der UI-Anforderungen sind Ansätze zu erkennen. So wird eine Autosuggestion Funktion basierend auf semantischen Modellen (der DBPedia-Ontologie) angeboten.

Weiter wird dem Benutzer ermöglicht alternative Recherchepfade auszuwählen. So werden Unterbegriffe (Teilgebiete) des recherchierten Suchbegriffs angeboten.

Auch gibt es eine History-Funktion, die dem Benutzer ermöglicht auf frühere Suchanfragen zurückzugehen oder Suchanfragen zu kombinieren. Jede Suchanfrage wird in unterschiedlichen Tabs angeboten, zwischen denen der Benutzer jederzeit wechseln kann.

5.2 Sig.ma

„Sig.ma“ ist eine LD Webapplikation die Informationen des Semantic Web aus unterschiedlichen Datenquellen in einem Mashup anzeigt. Zusätzlich wird ein Service angeboten, der diese Daten auch für maschinelle Programme zur Verfügung stellt.

Die Basisidee von sig.ma (sowohl der Anwendung als auch der API) ist die Bereitstellung von „Entity Profiles“ – eine Zusammenfassung von Informationen, die zu einem Thema gefunden werden. In der Webapplikation werden die Informationen visuell in einem Mashup aufbereitet. Die API (zur Verwendung in Programmen) stellt die Daten über ein JSON-Array zur Verfügung.[41], S. 2

Sig.ma verwendet eine Vielzahl von unterschiedlichen Technologien und Methoden wie semantische Suchmaschinen, Datenaggregation, Mapping von Ontologien, externe Webservices und das Lernen durch Benutzerverhalten.

5.2.1 Userinterface

Die Webapplikation stellt dem Benutzer oder der Benutzerin zunächst ein Eingabefeld für die Eingabe eines Suchbegriffs (beispielsweise eines Personennamens) bereit. Die Ergebnisse dieses Suchbegriffs werden dann in einem Mashup aggregiert. Der Benutzer/die Benutzerin hat dann die Möglichkeit die Ergebnisse zu verbessern, in dem Ergebnisse bestimmter Datenquellen ignoriert, höher gerankt oder umsortiert werden.

Abbildung 14 zeigt einen Screenshot des Mashup für die Suchanfrage „Bernhard Schandl“. Im linken Bereich werden die gefundenen Informationen für den Benutzer aufbereitet angezeigt. Diese Anzeige ist nach dem Prinzip key-value tabellarisch aufgebaut, analog zu klassischen LD Browsern, wobei bei jedem Wert immer eine oder mehrere Zahlen in eckigen Klammern angezeigt werden. Diese Zahlen repräsentieren die Datenquelle, in der die Information gefunden wurde.

Abbildung 14: Sig.ma Mashup

In der rechten Seite der Anzeige sind alle Datenquellen aufgelistet. Der Benutzer hat die Möglichkeit, das Ergebnis zu verfeinern, indem er bestimmte Aussagen komplett aus der Ergebnisliste entfernt oder sämtliche Aussagen bestimmter Datenquellen von der Ergebnisliste entfernt. Zusätzlich besteht die Möglichkeit neue Datenquellen hinzuzufügen.

5.2.2 Architektur und Workflow

Abbildung 15(auf Seite 65) zeigt den Ablauf der Verarbeitung einer Benutzereingabe. Rechts von der gestrichelten Linie sind externe Services gelistet, die während der Verarbeitung verwendet werden. Links der gestrichelten Linie ist der lokale Ablauf innerhalb der Applikation dargestellt. Die wesentlichsten Komponenten werden in der Abbildung zusammengefasst.

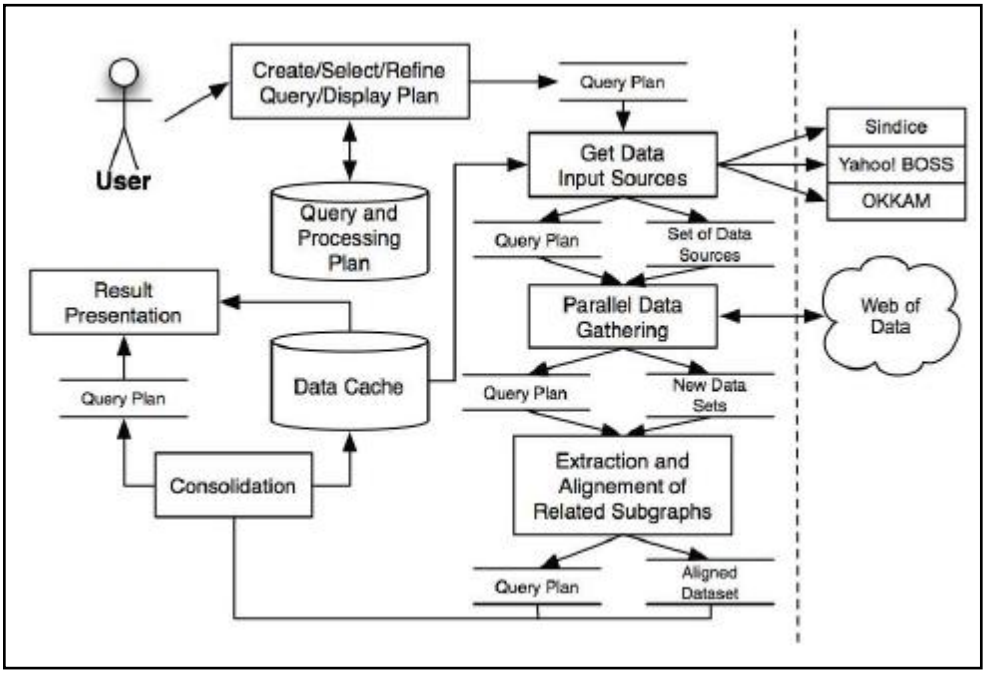

Abbildung 15: Workflow Recherche in sig.ma [41], S. 2

Ein wichtiger Schritt für eine performante Antwortzeit bei der Suche ist die Beschränkung auf bestimmte Datenquellen.

Im Schritt „**Get Data Input Sources**“ werden externe Services abgefragt, die URIs liefern, welche Informationen zur gestellten Suchanfrage des Benutzers enthalten. Dazu werden die drei Datenquellen „Sindice“, „Yahoo! BOSS“ und „OKKAM“ abgefragt. *Sindice* ist eine semantische Suchmaschine, die auch via APIs abrufbar ist. *Yahoo! BOSS* ist eine API von Yahoo, die es ermöglicht Anfragen an die YAHOO Suchmaschine zu stellen. *OKKAM* ist ein co-reference-Dienst der es ermöglicht URIs zu finden, die die gleiche Ressource beschreiben. [41], S. 2

Im Schritt „**Parallel Data Gathering**“ werden die 20 ersten URI im „Web of Data“ abgefragt und durch den any23[48]-Parser werden die RDF-Daten extrahiert. [41]

Im nächsten Schritt, dem „**Extraction and Alignment of Related Subgraphs**“, werden die RDF-Beschreibungen der Ressourcen aus jedem RDF-File extrahiert. Die RDF-Daten einer URI können mehrere *„resource descriptions“* enthalten, beispielsweise die eigentliche Beschreibung der Ressource aber auch zusätzliche Beschreibungen wie beispielsweise Provenance Metadaten. [41], S. 2
Diese Informationen werden dann gerankt, indem jedes Literal überprüft wird, ob die keywords der Suchanfrage des Benutzers enthalten sind. Weiter werden die

[48]http://developers.any23.org[zuletzt angesehen 17.05.2011]

URIs (RDF Links) nach dem gleichen Verfahren überprüft. Übereinstimmungen in Literalen, haben eine höhere Bedeutung (und somit eine höhere Gewichtung) als Übereinstimmungen in RDF Links. Ressourcen, deren Beschreibung mit dem Suchwort übereinstimmen, werden höher gerankt als Ressourcen, die keine Übereinstimmung ergeben. Ressourcen, die beim Ranking unter einen definierten Schwellenwert fallen, werden für die Generierung des Ergebnisses für den Benutzer nicht berücksichtigt. [41], S. 2

Im Schritt „**Consolidation**“ wird schließlich das Suchergebnis für den Benutzer aufbereitet. Dabei ist es notwendig bestimmte Beschreibungen (z.B. URIs die Prädikate oder Objekte beschreiben) so darzustellen, dass ein menschlicher Benutzer die Informationen leicht verstehen kann. [41], S. 3

Dazu werden zunächst die ***Prädikate*** normalisiert und aggregiert, um so ein Ranking der nach Prädikaten zu ermöglichen. Durch dieses Ranking ist es möglich dem Benutzer jene Informationen zu Beginn des Ergebnisses anzubieten, die am vertrauenswürdigsten sind. Die Vertrauenswürdigkeit basiert darauf, dass diese Informationen durch mehrere Datenquellen bestätigt werden.[41], S. 3

Für die Normalisierung der Prädikate wird der letzte Teil der Prädikat-URI genauer betrachtet. Dazu wird der Text durch Zeichenersetzungen (beispielsweise Unterstriche durch Leerzeichen oder Aufhebung der CamelCase-Schreibweise) und Anwendung von Heuristiken der englischen Sprache angepasst. [41], S. 3
In einem weiteren Schritt werden manuell erstellte Mappings auf den Text angewandt. So ist der Term „web page“ gleichbedeutend mit Termen wie „homepage“, „website“ oder „page“. Daher werden diese dem Benutzer auch nur unter dem bevorzugten Term im Ergebnis angezeigt.[41], S. 3
Die Prädikate werden dann noch gereiht. Prädikate, die eine große Bedeutung für Benutzer haben, wie „labels“, „despictions“, „short descriptions“ und „weblinks“, werden weit oben gereiht. Zusätzlich dazu werden aber auch Prädikate entfernt, die für den Benutzer keinen Mehrwert liefern (beispielsweise Informationen zum RDF Schema). [41], S. 3

Das Ranking der Prädikate basiert auf der Analyse wie viele der Datenquellen dieses Prädikat verwenden. Zudem werden jene Prädikate zusammengefasst, deren Aussagen in verschiedenen Datenquellen übereinstimmen. Somit wird die Ergebnisanzeige für den Benutzer übersichtlicher.[41], S. 3

Sind ***Objekte*** selbst wieder URIs, muss dem Benutzer ein passender Text als Linktext angezeigt werden. Dazu wird die URI abgefragt und nach Prädikaten wie dc:title, foaf:name, oder rdfs:label gesucht. Werden keine solchen Informationen

gefunden, wird der letzte Teil der URI ebenso normalisiert und die Prädikate und dem Benutzer so angezeigt.[41], S. 3f

Zur Verbesserung der Performance werden diese Informationen in einem Cache abgespeichert damit sie bei einer nächsten Abfrage wiederverwendet werden können. [41], S. 3

5.2.3 Technologie

Die LD Webapplikation ist in zwei Layern aufgebaut – das Backend und das Frontend. Das Backend ist eine Webapplikation mit Servlet-Technologie (im Tomcat) und stellt eine RESTful API zur Verfügung. Das Frontend ist vollständig in JavaScript mittels eines MVC-Pattern realisiert worden. [41], S. 4

Da das parallele Abfragen von mindestens 20 Datenquellen Zeit in Anspruch nimmt, wurde der Aufbau des Mashups durch AJAX-Abfragen des Backends realisiert. Der Benutzer/die Benutzerin erhält so schrittweise die Informationen (des „entity profile“) und kann den Fortschritt des Aufbaus des Ergebnisses verfolgen. [41], S. 4

Durch den Caching-Mechanismus wird es möglich, bei der Verarbeitung einer Suchanfrage eine durchschnittliche Performance von zehn Datenquellen pro Sekunde zu erzielen.[41], S. 4

5.2.4 Schlussfolgerungen

„Sig.ma“ verwendet als Consuming-Strategie das „On-the-fly dereferencing pattern“, also die Nachverfolgung von Links im „Web of data“. Dennoch wird dieses Pattern kombiniert mit der Verwendung einer semantischen Suchmaschine. Ergebnisse dieser semantischen Suche ermöglichen einen Einstiegspunkt in dieses „Web of data“.

Elemente der von Hartig et al. beschriebenen Software-Architektur (siehe Kapitel 4.2.3 ab Seite 50) können auch auf diese Applikation angewendet werden. Es wird zwar kein klassisches „Crawling pattern“ angewendet, jedoch werden die Daten adhoc ge-crawled und aggregiert. Bei diesem Prozess sind die Workflows zur Datenintegration aber sehr wohl zu berücksichtigen.

Insbesondere bei der Aggregation der Daten sind die Module „Web Data Access Module“, „Vocabulary Mapping Module“, „Identity Resolution Module“ und „Quality Evaluation Module“ zu beachten.

Der einheitliche Zugriff auf die unterschiedlichen Formate des LD-Web wird durch den any23-Parser realisiert.
Zum „Vocabulary Mapping“ existiert eine manuell erstelltes Mapping, das ähnliche Eigenschaften zu einer Gruppe zusammenfasst und diese Gruppen den Benutzern präsentiert.
Das „Identity Resolution Module“ wird insbesondere durch die Abfrage der Co-Reference Diensts „OKKAM“ realisiert.
Das „Quality Evaluation Module“, also die Überprüfung der Qualität, wird in Sig.ma über das Ranking realisiert. Informationen, die von mehreren Quellen stammen, sind vertrauenswürdiger als Informationen, die nur von einer Quelle stammen. Weitere Kriterien zur Überprüfung der Qualität der Ergebnisse ist die Prüfung, ob Suchterme in Literalen und Links der RDF-Daten der Ressourcen enthalten sind.

Das Userinterface ist mit JavaScript und AJAX-Abfragen implementiert, was wohl für Applikationen, die das „On-the-fly dereferencing pattern“ einsetzen, notwendig ist. Somit kann der Benutzer den Aufbau der Ergebnisliste mit verfolgen und muss nicht warten bis die Seite komplett aufgebaut ist. Das erhöht die „User experience“ (siehe Kapitel 4.3.2 ab Seite 55) für Applikationen mit längerer Antwortzeit.

Eine weitere wichtige Funktionalität dieser Applikation ist die Nachvollziehbarkeit der Quellen der angezeigten Informationen. Ein Benutzer hat die Möglichkeit bestimmte Quellen als vertrauenswürdig oder als nicht vertrauenswürdig einzustufen. Das System kann aus diesen Benutzerinteraktionen lernen.

5.3 RKB Explorer

RKBExplorer ist eine LD Webapplikation, die eine einheitliche Sicht für viele LD Datasets zur Verfügung stellt. Diese Applikation ist aus dem EU-Projekt ReSIST entstanden, das den Aufbau einer semantisch-unterstützten Wissensinfrastruktur befördert. Der Hauptaspekt des Projekts liegt auf dem Publizieren, Übernehmen von verteilten Ressourcen und der Verlinkung dieser Ressourcen. [44], S. 1

„RKB Explorer“ stellt Datenprovidern einen SPARQL-Endpoint, dereferenzierbare URIs, Datenbrowser, Datenexplorer, konsistente Verlinkungs-Dienste und eine keyword-basierte Suche zur Verfügung. [44], S. 2

5.3.1 Userinterface

Abbildung 16 zeigt das Userinterface der Applikation. Startpunkt für die Recherche ist immer eine URI. Die Idee dieser Applikation ist ähnliche URIs im Kontext „People“, „Organisations“, „Publications“ und „Courses & Materials“ anzuzeigen, um dem Benutzer ein Browsing darin zu ermöglichen. Sollte die URI nicht bekannt sein, kann über eine keyword-basierte Suche in den Literalen nach einer Ressource gesucht werden.

Im Kopf der Applikation befindet sich das Suchfeld für eine keyword-basierte Suche. Das große Fenster zeigt eine grafische Visualisierung der Beziehung zwischen den URIs. Das rechte Fenster, neben der Visualisierung, zeigt Details zu aktuellen URI. Die vier Fenster am unteren Rand der Applikation zeigen die verlinkten URIs der vier Bereiche „People“, „Organisations“, „Publications“ und „Courses & Materials“.

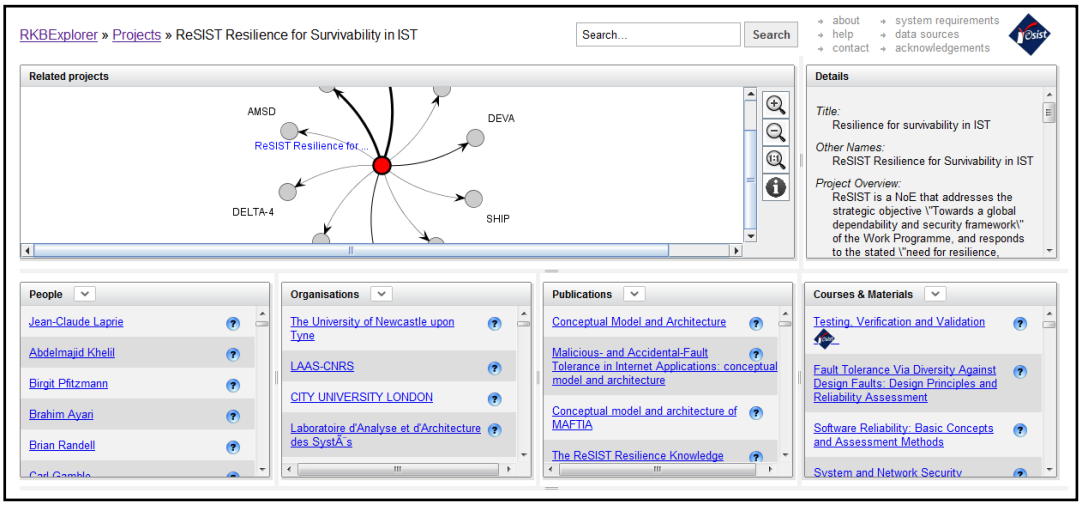

Abbildung 16: RKBExplorer GUI

5.3.2 Workflow der Integration neuer Datasets

Alle Datasets werden in separaten RDF Stores gespeichert und werden als eindeutige URIs im LD Web angeboten:

http://<Name des Datasets>.rkbexplorer.com

[44], S. 3

Alle verfügbaren Datasets können unter [45]abgerufen werden.

Die Integration neuer Datasets kann automatisiert geschehen. Dabei werden folgende Schritte durchlaufen:

- Konvertierung der Daten in RDF,
- Speicherung im RDF-Store,
- Daten in SPARQL-Endpoint bereitstellen,
- anbieten eines Databrowsers, UI und keyword-basierte Suche. [44], S. 2

Zusätzlich werden die Daten analysiert, um Links zu anderen bereits integrierten Dataset bilden zu können und auch die existierenden Datasets in beide Richtungen zu bilden. Dabei werden Beziehungen zwischen Publikationen und Personen ermittelt und in einer separaten Knowledgebase gespeichert. [44], S. 3f

5.3.3 Architektur

Diese separate Speicherung der Knowledgebase ermöglicht eine Skalierbarkeit des Systems und eine gute Query Performance.

„Consistent Reference Service“ (CRS) stellt eine Knowledgebase zur Verfügung, die Mappings zwischen URIs bestimmter Kategorien für die Applikation bereitstellt. Durch die separate Speicherung der Mappings in Knowledgebases kann die Trennung von Informationen erfolge die

- vom Provider bereitgestellt und
- von RKB Explorer ermittelt

wurden. Dadurch kann auf unterschiedliche Anforderungen der Datasets eingegangen werden. [44], S. 3f

Grundlagen für das Mapping der URIs sind:

- „publication place“ für Publikationen
- „funding body“ für Projekte und
- „place of work“ für Menschen [44], S. 4

5.3.4 Schlussfolgerungen

Als Consuming-Strategie verwendet “RKB Explorer” das „Crawling-Pattern“. Komplette Datasets des LD Web, die Daten zu Publikationen veröffentlichen, werden gecrawled und in separaten RDF-Stores gespeichert.

Für diesen Crawling-Aspekt kann die von Hartig et al. beschriebene Software-Architektur (siehe Kapitel 4.2.3 ab Seite 50) angewendet werden. Dadurch, dass alle Daten in lokalen RDF-Stores vorliegen, können die Module „Vocabulary Mapping“, „Identity Resolution“ und „Quality Evaluation“ leicht implementiert werden.

Der Aspekt der Performance von Abfragen auf RDF-Stores wird durch seperate Knowledgebases begegnet, die bestimmte Mappings enthalten, die performant abgefragt werden können.

„RKB Explorer“ stellt zudem eine re-publishung-Komponente bereit, die die Daten des Datasets auch mittels URIs und SPARQL-Interface bereitstellt.

Das Userinterface zeigt Ansätze der explorativen Navigation. So können Beziehungen zwischen Ressourcen, Organisationen und Publikationen erforscht werden. Eine keyword-basierte Suche ermöglicht den Einstieg in die Naviagtion.

5.4 Zusammenfassung

Die drei Fallstudien zeigen die Verwendung unterschiedlicher Consuming-Strategien. Abhängig von der gewählten Strategie sind bestimmte Module der Software-Architekturen mehr oder weniger ausgeprägt.

Die Fallstudien zeigen aber auch, dass das Thema Datenintegration mit Mappings von Vokabularien, Ressourcen als auch die Bewertung der Qualität essentiell wichtig ist. In Systemen, die die Daten vorab verarbeiten können, kann diese Analyse und Verarbeitung umfangreicher sein.
In Systemen, die auf die Daten adhoc zugreifen müssen Methoden der Benutzerinteraktion eingesetzt werden, um beispielsweise die Qualität eines Datasets zu bewerten.

Performance ist eine wichtige Anforderung einer Webapplikation. Werden Consuming-Strategien verwendet, die längere Antwortzeiten für Benutzeranfragen haben, müssen Caching-Methoden eingesetzt werden, um die „User Experience“ zu steigern.
Zentrale Elemente von Software-Architekturen für LD Webapplikationen ist der Zugriff auf die Daten durch einheitliche Schnittstellen auf lokale und entfernte Datenquellen, URIs, SPARQL-Interfaces, semantische Indexes und APIs.

Datenintegration ist die zentrale Aufgabe einer LD Webapplikation. So müssen:

- unterschiedliche Vokabulare auf ein einheitliches (intern verwendetes) Vokabular gemappt werden,
- Co-reference Beziehungen erkannt werden und
- Die Qualität der Informationen bei der Aggregation erkannt werden.

Datengrundlage für die Implementierung einer explorativen Suche ist die Auswertung der Links zwischen Ressourcen. Die Visualisierung dieser

Beziehungen kann von Applikation zu Applikation variieren, abhängig von den Anforderungen der Benutzer.

6 Prototyp

Die gewonnenen Erkenntnisse aus den Bereichen “Consuming Linked Data” und “Linked Data Webapplikationen” sollen nun ausgetestet werden. Dazu wird ein Prototyp implementiert, der es ermöglicht nach bibliografischen Informationen des LD Web zu recherchieren. Für diesen Prototyp sollen vorab ausgewählte Datasets verwendet werden. Daher wird zunächst die Domain „Library Linked Data“ untersucht, um daraus jene Datasets auszuwählen, die für den Prototyp verwendet werden können.

6.1 Library Linked Data

Wie in Abbildung 4 (auf Seite 20) gut zu erkennen, ist der Bereich der Publications (grün) bereits sehr aktiv bei der Bereitstellung von Daten als Linked (Open) Data.
In diese Gruppe fallen unterschiedlichste Initiativen: u.a. die Daten von Bibliothekskatalogen, Daten von Taxonomien und Thesauri, Daten zu wissenschaftlichen Artikeln von Zeitschriften und Konferenzen, Daten von Werkzeugen zur Erstellung von Literaturlisten und viele ähnliche Initiativen. [6], Kapitel 3.2.5

Um die Entwicklungen in diesem Bereich zu koordinieren wurde vom W3C in Zusammenarbeit mit vielen namhaften Bibliotheken und Herstellerfirmen von Bibliothekssoftware-Systemen die „Library Linked Data Incubator Group“ (LLD XG) gegründet. In diesem Zusammenhang wurde der Name „Library Linked Data“ etabliert, um Daten der LD Cloud zu adressieren, die im Bereich der Bibliotheken anzusiedeln sind.

Der Charter der LLD XG ist folgendes Ziel dieser Gruppe zu entnehmen:

> „*The group will explore how existing building blocks of librarianship, such as metadata models, metadata schemas, standards and protocols for building interoperability and library systems and networked environments, encourage libraries to bring their content, and generally re-orient their approaches to data interoperability towards the Web, also reaching to other communities. It will also envision these communities as a potential major provider of authoritative datasets (persons, topics...) for the Linked Data Web.*” [46]

Die Gruppe wurde durchaus mit dem Wissen gegründet, dass Bibliotheken und verwandte Einrichtungen qualitativ sehr hochwertige, strukturierte Daten verwalten. Diese Daten könnten eine Stütze des gesamten LD Webs sein. Um

das LD Web zu etablieren ist das W3C bemüht hier auch qualitativ hochwertige Daten als Teil dieses LD Web zu bekommen. Für die Verwirklichung/Umsetzung, bietet das W3C mit dieser Gruppe Hilfestellung zur technischen und organisatorischen Implementierung.

Ein erstes Ergebnis dieser Gruppe ist ein Dokument entstanden, das mögliche Anwendungsfälle für LD in der Domain der „Publications“ beschreibt. Diese werden im nächsten Kapitel 6.1.1nun genauer betrachtet.

6.1.1 Anwendungsfälle

Als eine der ersten Aktionen der LLD XG wurden Anwendungsfälle für die Domain der bibliografischen Daten im LD Web gesammelt zusammengestellt und unter [47] veröffentlicht.

Die gesammelten Anwendungsfälle wurden in Bereiche aufgeteilt:

- bibliografische Daten
- Normdaten
- Vokabular Mapping
- Archive und heterogene Daten
- Zitate
- Digitale Objekte
- Sammlungen
- Anwendungen im Bereich „social media“

[47]

Auf einige vielversprechende Bereiche soll nun näher eingegangen werden.

Der Bereich der ***bibliografischen Daten*** bietet einige Beispiele wie regionale oder sogar länderübergreifende Kataloge gebildet werden können. Die größte Herausforderung in diesem Bereich ist wohl die Vermeidung von Dubletten oder redundanten Daten, da viele Bibliothekskataloge ähnliche wenn nicht gleiche Informationen in ihren Katalogen verwalten.

Das Projekt „culturegraph.org“ stellt dazu einen Service bereit, der diese co-reference-Beziehungen unter einer zentralen URI anbietet. [48]Abbildung 17zeigt die Idee von culturegraph.org:

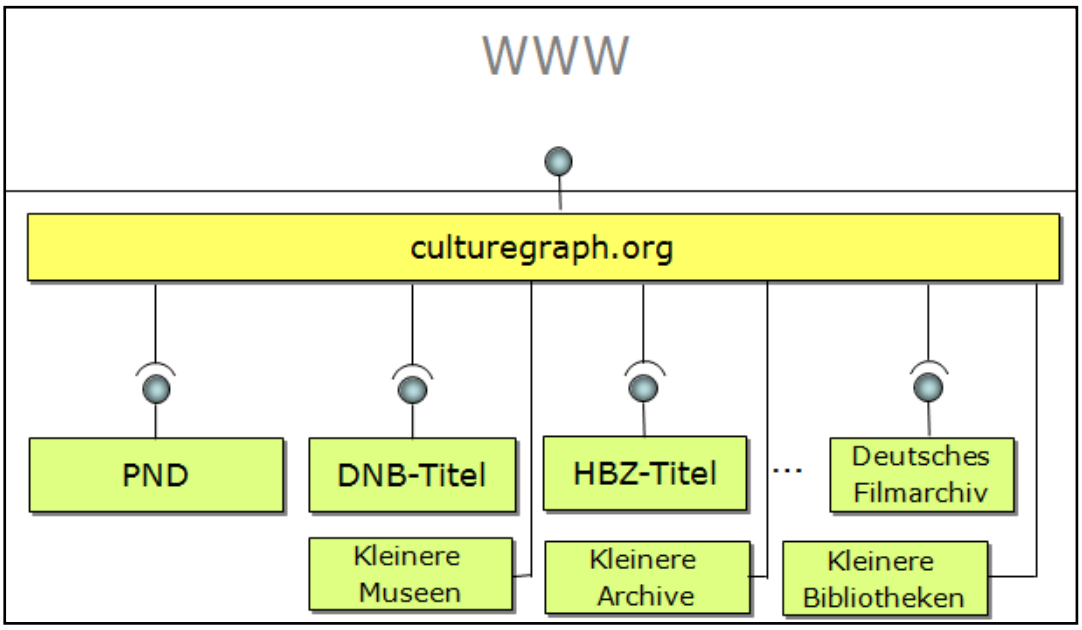

Abbildung 17: culturegraph als co-reference-Dienst[49]

Der Bereich der ***Normdaten***kann eine wichtige Grundlage für das LD Web bilden. Das Wikipedia-Projekt sammelt kollaborativ viele Informationen über Personen, Konzepte, Orte etc. Diese Informationen werden durch Millionen von Benutzern zusammengetragen. Durch die Vielzahl von Autoren kann der Benutzer davon ausgehen, dass jene Informationen der Wahrheit entsprechen. Jedoch darf sich ein Benutzer darauf nicht verlassen. Informationen zu Personen, Institutionen und Konzepten werden seit Jahrzehnten (wenn nicht schon länger) in Form von Normdaten durch Bibliothekare kontrolliert verfasst und verwaltet. Diese Informationen können als qualitativ hochwertiger angesehen werden, als Enzyklopädien, die im kollaborativen Ansatz verfasst werden.
Werden diese Informationen im LD Web veröffentlicht, gelten diese als vertrauenswürdiger und könnten von einer Vielzahl von Services (auch außerhalb der Domain der Bibliotheken) verlinkt und verwendet werden.

Diverse Untersuchungen der Erwartungen von Benutzern in bibliografischen Recherchesystemen haben ergeben, dass Benutzer mehr als nur bibliografische Metadaten erwarten – sie wollen den Content am liebsten sofort in Form von Volltexten oder zumindest des Inhaltsverzeichnisses. Da die Digitalisierung von Büchern und ähnlichen Werken sehr aufwendig und teuer ist, müssen Bibliotheken sich hier anderweitig behelfen. Somit ist es notwendig, dass die Bibliotheken untereinander ***Digitalisierungsprojekte*** koordinieren und Ergebnisse austauschen, um so Geld und Zeit zu sparen. Weiter hilfreich sind Mammutprojekte, wie sie beispielsweise die Österreichische Nationalbibliothek

unternimmt, indem sie eine Kooperation mit Google[49] eingegangen sind, um hunderttausende Bücher digitalisieren zu lassen. Würden Daten von solchen und ähnlichen Projekten über LD bereitgestellt werden, könnten Bibliothekskataloge durch diese Daten angereichert werden und würden sehr viel an Attraktivität gewinnen.

Literaturverwaltungsplattformen verwalten nicht nur Metadaten über die zitierten Aufsätze, sondern können auch Querverbindungen zwischen Autoren oder Publikationen herstellen, so dass ein Benutzer schnell einen leichten Überblick über ein Forschungsgebiet erhalten kann. Sollten solche Daten als LD angeboten werden bieten sie Grundlagen für den Einstieg in die Recherche eines Themas aber auch für Empfehlungsdienste und weitere Services.

6.1.2 Analyse LLD Datasets

Da die LOD-Cloud (siehe Abbildung 4 auf Seite 20) auf Informationen des CKAN-Portals basiert[13], kann CKAN eine Basis sein, um den Bereich der bereitgesellten Daten im Bereich „Publications“ zu untersuchen.

Abfragen in CKAN sind über die API möglich genauso wie über die Lucene Abfrage-Sprache[50] im Web. Für den Bereich der „Publications“ wurde von [14]die Abfrage verwendet:

tags:publications AND
groups:lodcloud AND
-tags:lodcloud.needsfixingAND
-tags:lodcloud.nolinks AND
-tags:lodcloud.unconnected

Diese Suchanfrage liefert (Stand 11.03.2011) 72[51] Pakete. Darin enthalten sind Datasets, die Informationen zu Publikationen als auch Datasets von Bibliotheken, die bibliografische Metadaten publizieren. Im Prototyp soll aber nur LLD betrachtet werden. Daher wurde die Suchanfrage speziell auf die Gruppe LLD eingeschränkt. Eine Einschränkung der Suchanfrage auf die Gruppe LLD („Library Linked Data“) liefert 15 Pakete. In CKAN sind aber noch ähnliche Tags zu

[49]http://googlepolicyeurope.blogspot.com/2010/06/unlocking-our-shared-cultural-heritage_4958.html[zuletzt angesehen 17.05.2011]
[50]http://wiki.ckan.net/Searching_Packages[zuletzt angesehen 17.05.2011]
[51]Anmerkung: In der Tabelle „Linked Data by Domain“ wird gezeigt, dass dieser Bereich 68 Datasets inkludiert. Eine entsprechende Suche (Verfolgung des Links) bracht am 11.03.2011 aber 72 Datasets als Ergebnis.

bibliografischen Daten vorhanden wie „library“ und „bibliographic“. Um alle relevanten Datasets aus diesem Bereich zu ermitteln, wurde die CKAN-Query wie folgt abgeändert:

(tags:lld OR tags:bibliographic OR tags:library) AND
groups:lodcloud AND
-tags:lodcloud.needsfixing AND
-tags:lodcloud.nolinks AND
-tags:lodcloud.unconnected

Diese Anfrage liefert 21 Datasets, die nach den Kriterien

- Anzahl der Triple pro Dataset
- Vorhandensein einer voiD Beschreibung des Dataset
- Vorhandensein und Verfügbarkeit eines SPARQL-Interfaces
- Anzahl der Verlinkungen in andere Datasets
- Lizenzen

nachfolgend genauer untersucht werden. Für die vorliegende Untersuchung wurden die Daten der CKAN-Plattform verwenden. In dieser Untersuchung werden nur die ursprünglichen Data-Provider berücksichtigt und keine „Re-Publisher“. Aus diesem Grund wird das Dataset („Open Library data mirror in the Talis Plattform“) nicht betrachtet. Zahlenmaterial der Untersuchung ist in Anhang A (auf Seite 116) zu finden.

Datasets

Insgesamt sind im untersuchten Bereich 858.081.526 Triples verfügbar. Das entspricht einer durchschnittlichen Anzahl von 42.904.076,30 Triples/Dataset. Lediglich 35 % der untersuchten Datasets bieten ein SPARQL-Interface an und nur 25% der Datasets bieten eine voiD-Beschreibung an.

In den Datasets können nach Art der Informationen, die bereitgestellt werden, klassifiziert werden:

- Normdaten und Thesauri
- Titeldaten, Publikationen und Bestandsnachweise in Bibliotheken
- Digitale Objekte
- Institutionen

Abbildung 18(auf Seite 78) zeigt die Anzahl der Datasets nach dieser Klassifikation und Abbildung 19 (auf Seite 78) zeigt jene der Triples entsprechend dieser Klassifikation.

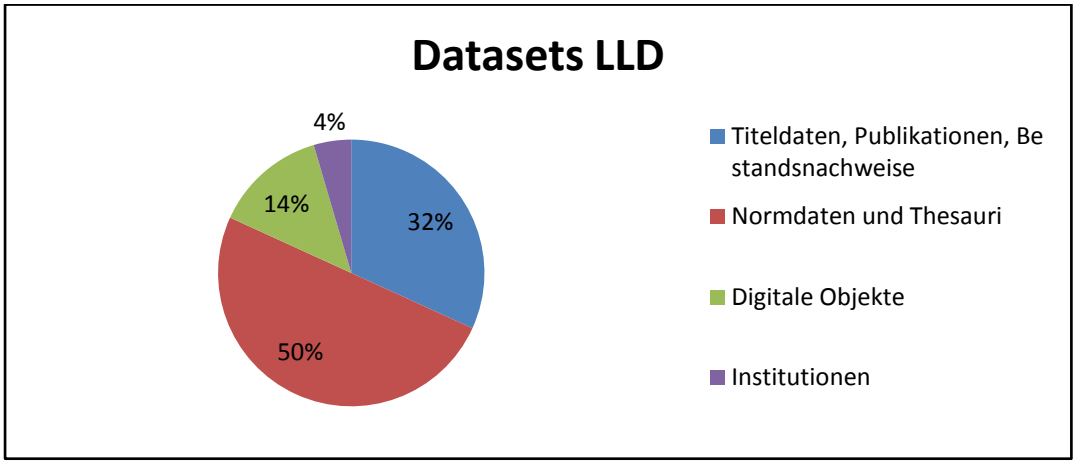

Abbildung 18: Klassifikation der Datasets LLD

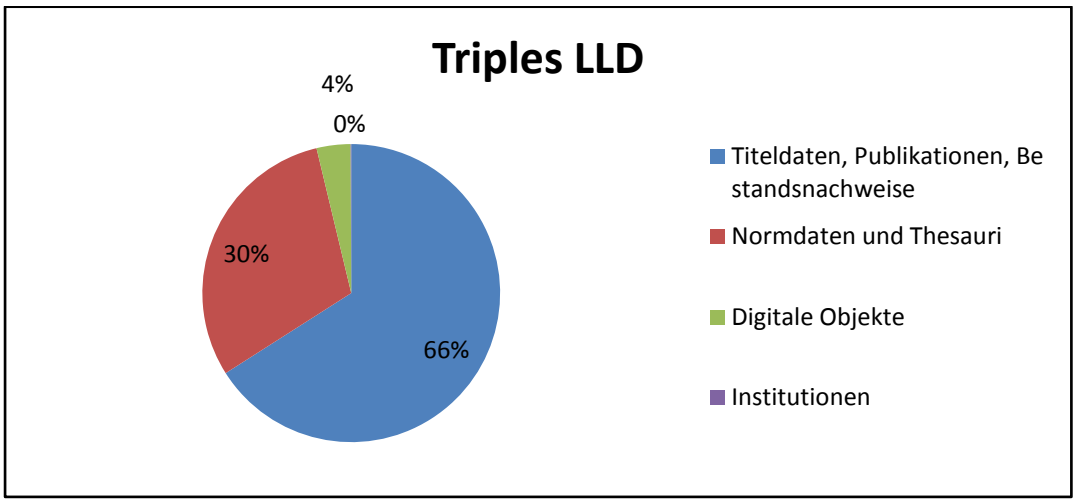

Abbildung 19: Klassifikation der Triples LLD

Bei genauerer Betrachtung zeigen die Abbildungen, dass zwar die meisten LLD Datasets Normdaten oder Thesauri bereitstellen, aber absolut an Triples gemessen werden am häufigsten Titeldaten, Publikationen und Bestandsnachweise publiziert.

Erfreulich ist auch, dass Metadaten und Links zum Zugriff auf digitale Objekte veröffentlicht werden. Die Integration dieser Daten ermöglicht einen wirklichen Mehrwert für bibliografische Anwendungen.

Verlinkung

Interessant ist auch die Verlinkung zu anderen Datasets. So sind insgesamt 19.546.410 Links vorhanden.In Abbildung 20(auf Seite 79) ist die Verteilung dieser Links nach Ziel-Datasets dargestellt.

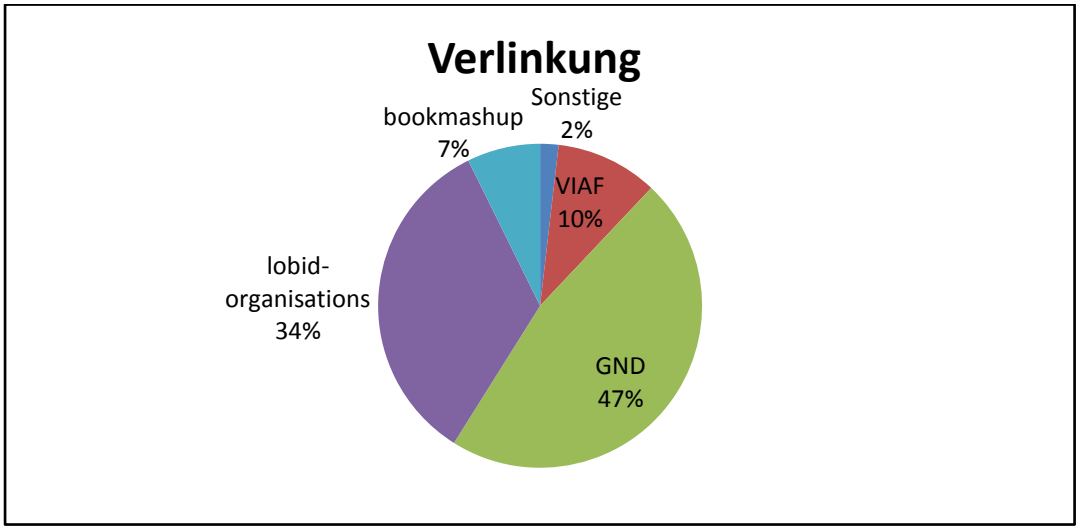

Abbildung 20: Verteilung der Links im Bereich LLD nach Ziel-Datasets

Die große Verwendung von lobid-organisations und GND ist durch die Initiativen in Deutschland zu erklären. Die LD Dienste der Deutschen Nationalbibliothek, des Hochschulzentrums Nordrhein-Westfalen und der Universität Mannheim stellen Links zu diesen Daten bereit. Da bei dieser Art der Auswertung nur die absolute Anzahl der Links zählt, werden kleinere Datasets nicht genügend berücksichtigt.

Eine weitaus bessere Kennzahl zur Bewertung der ausgehenden Links ist ihr Vergleich zur Größe des Datasets (Anzahl der Triple). Abbildung 21(S. 80) zeigt diese Auswertung. Darin ist zu erkennen, dass durchaus kleine Datasets wie die marccodes-list eine sehr gute Verlinkungshäufigkeit aufweisen.

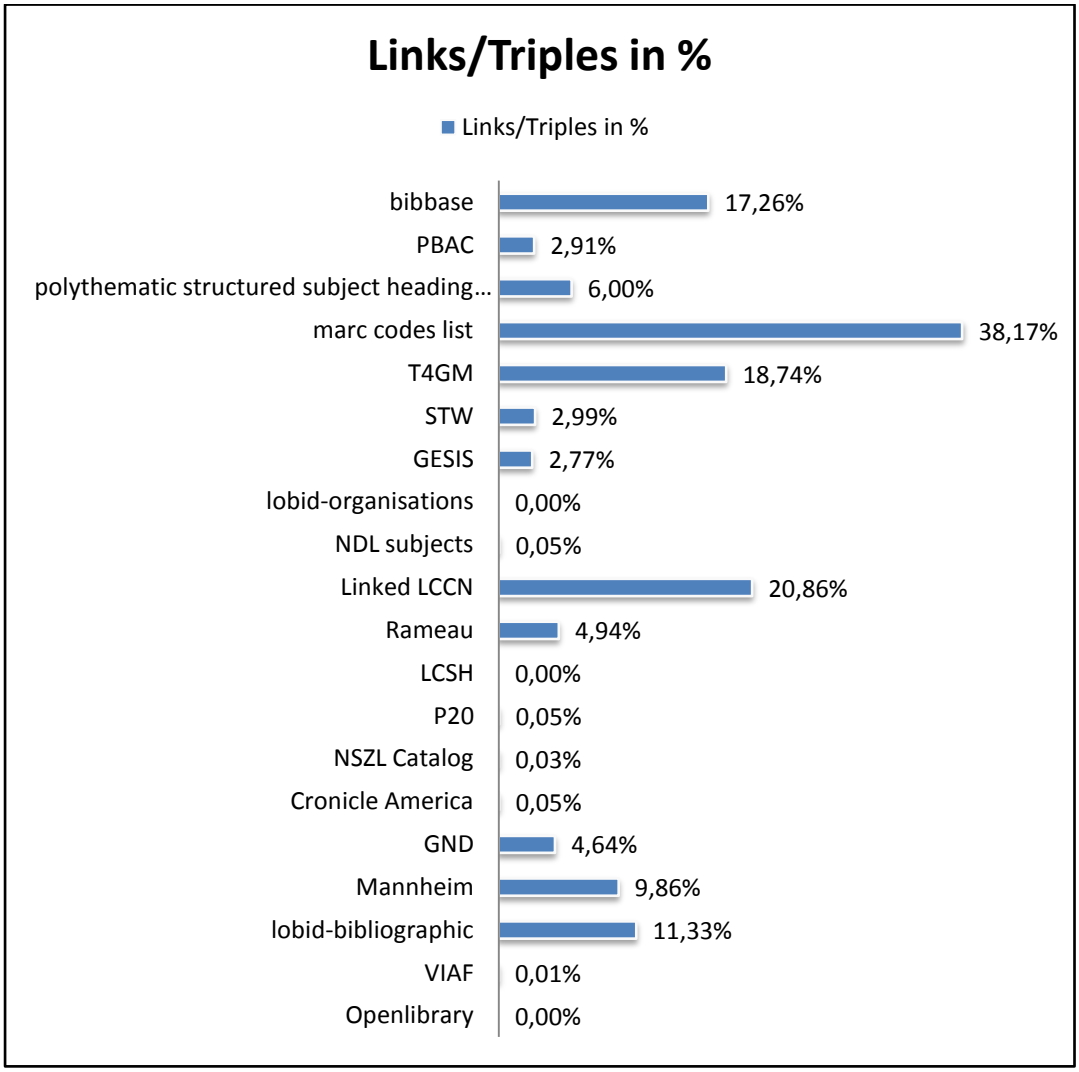

Abbildung 21: Ausgehende Links/Dataset für LLD-Datasets

Eine noch bessere Kennzahl für die Verlinkung wäre die Anzahl der ausgehenden Links pro Ressource. Leider war es nicht möglich, diese Daten aus den CKAN-Daten zu ermitteln.

Wird die Art der Verlinkung in diesem Bereich genauer betrachtet, fällt auf, dass unterschiedliche Arten von Links verwendet werden:

- Links zu Normdaten wie GND, marccodes, VIAF, etc.
- Links zu bibliografischen Titeldaten (co-reference Verlinkungen)

Auffällig ist, dass weitere bekannte Datasets aus dem bibliothekarischen Bereich wie LIBRIS[52] oder JISC[53] nicht in dieser Statistik enthalten sind. Diese Datasets müssten in CKAN mit entsprechenden Tags versehen werden.

[52] http://ckan.net/package/libris [zuletzt angesehen 17.05.2011]

6.2 LLD Search Prototyp

Basierend auf der Analyse

- der *Consuming*-Strategien von LD (Kapitel3),
- der Software-Architekturen von LD Webapplikationen (Kapitel 4) und
- der vorhandenen Datasets im Bereich von LLD (Kapitel 6.1.2)

wird an dieser Stelle das Konzept eines Prototyps präsentiert, der die Daten des LLD Webs sowohl für einen menschlichen Anwender als auch in Teilen für Computerprogramme recherchierbar machen soll.

Die Webapplikation „LLD Search“ stellt ein Recherche-Instrument für (derzeit ausgewählte) Datasets des LLD Web bereit. Zudem werden Daten in Form eines „LD Browsers“ zum Navigieren in den Daten angeboten.

6.2.1 Zielgruppen und Anwendungsfälle

Die Applikation kann von Benutzern mit unterschiedlichen Anforderungen genutzt werden. Dabei werden die drei Ebenen unterstützt:

1. Benutzer wollen Informationen nach dem Prinzip der klassischen Suchmaschine auffinden durch Eingabe eines oder mehrerer Suchbegriffe.
2. Benutzer wollen Querverbindungen folgen, um so möglichst auch neue, noch nicht bekannte Werke zu finden.
3. Benutzer wollen gezielt Abfragen mit der Abfragesprache SPARQL durchführen, um komplexe Bedingungen in der Abfrage abzubilden.

Zur Umsetzung dieser Anforderungen werden in „LLD Search“ folgende Recherche-Instrumente unterstützt:

- Keyword-basierte Suche
- Explorative Navigation in Themengebieten
- Formulierung von Suchanfragen mit SPARQL

Die drei Anwendungsfälle werden damit abgedeckt:

Fall A:

Ein Benutzer recherchiert nach Titeldaten mit dem Ziel zu erfahren in welcher Bibliothek das Buch zur Verfügung steht. Der Benutzer weiß in diesem Fall schon ziemlich genau wonach er sucht.

[53] http://ckan.net/package/jiscopenbib-bl_bnb-1 [zuletzt angesehen 17.05.2011]

Fall B:

Ein Benutzer sucht nach wissenschaftlichen Publikationen, um das Fachgebiet explorativ zu erforschen, beispielsweise um zu erfahren:

- Welche weiteren Autoren publizieren in diesem Fachgebiet?
- Welche weiteren Arbeiten wurden im gleichen Zeitraum veröffentlicht?
- Welche Unterthemen oder verwandte Themen existieren in diesem Fachgebiet?

Fall C:

Ein Benutzer möchte bibliografische Daten detailliert durch Bedingungen oder semantische Beziehungen abfragen. Hierfür steht die Abfragesprache SPARQL bereit.

Wie „LLD Search“ für die geschilderten Anwendungsfälle konkret funktioniert, wird in Kapitel 6.2.6 beispielhaft ab Seite 91 aufgezeigt. Zunächst wird aber das technische Konzept nachfolgend erklärt.

6.2.2 Konzept

„LLD Search“ besteht im Wesentlichen aus den drei Teilbereich:

- Suchmaschine zum effizienten Recherchen nach bibliografischen Inhalten des LLD Webs
- LD Browser zum Anzeigen dieser Daten für den menschlichen Nutzer
- Exploratives Erforschen des Kontexts dieser Daten

Abbildung 22 zeigt diese drei Bestandteile:

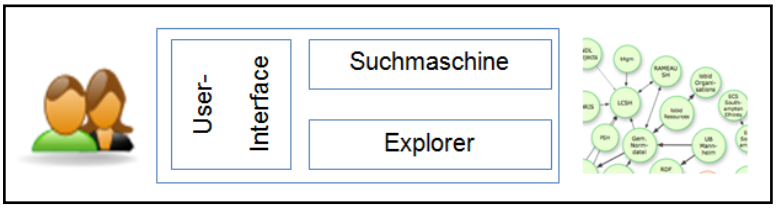

Abbildung 22: Module von LLD Search

Datenbasis

Um die Funktionalität dieses Prototyps aufzuzeigen, wurden zunächst einige LLD Dataset manuell ausgewählt. Als Grundlage dazu diente die Analyse der Datasets in Kapitel 6.1.2 ab Seite 76. Da es in den analysierten Datasets leider noch keine Daten von österreichischen Bibliotheken gibt, wurde entschieden, dass andere deutschsprachige Datasets für die Auswahl in Frage kommen. Hier ist die

Auswahl auf eine Kombination von lobid.org und den Datasets der deutschen Nationalbibliothek (Dataset „GND“) gefallen.

Im Lobid.org-Dataset sind jene bibliografischen Titeldaten und Bestandsnachweise von Bibliotheken des Hochschulbibliothekenzentrums des Landes Nordrhein-Westfalen (hbz), die unter opendata-Prinzipien veröffentlicht sind. Dieses Dataset kann zur Demonstration des Anwendungsfalls A verwendet werden.

Glücklicherweise sind viele Ressourcen des Lobid.org-Datasets bereits mit URI der „GND“ verlinkt, d.h. zu Einträgen der Personen- und Schlagwortnormdatei. Über diese Verlinkungen kann der Anwendungsfall B aufgezeigt werden, die explorative Erforschung einen Fachgebiets.

Für den Anwendungsfall C wird den Benutzern ermöglicht in öffentlich zugänglichen SPARQL-Interfaces zu recherchieren. Dazu wurden exemplarisch SPARQL-Interfaces, ausgewählt, die bibliografische Informationen zur Abfrage anbieten:

- http://lobid.org/sparql/
- http://data.bib.uni-mannheim.de/sparql
- http://lod.gesis.org/thesoz/sparql
- http://jisc.rkbexplorer.com/sparql/
- http://dblp.rkbexplorer.com/sparql/

Workflow einer Recherche

Ein Benutzer startet mit einer keyword-basierten Suche. Als Ergebnis dieser Anfrage wird eine Ergebnisliste mit Facettenklassifikation angeboten. Diese Ergebnisliste enthält bereits Elemente des explorativen Navigierens in Form von grafischen Visualisierungen (Tagclouds und Timelines).

Die Vollanzeige des Treffers stellt ein Mashup dar, dass alle Informationen dieser Ressource aggregiert. Dabei wird sowohl auf Informationen der Suchmaschine als auch auf Online verfügbare Daten zugegriffen. Methoden der explorativen Navigation werden hier eingesetzt, um möglicherweise relevante andere Ressourcen zu entdecken. Um das zu ermöglichen werden hier bibliografische Titeldaten mit Normdaten kombiniert und versucht über Personen und Schlagwortnormdateien weitere Links zu verfolgen und somit möglicherweise zusätzlich relevante Ressourcen anzubieten.

Technologie

Die Webapplikation wird mithilfe des JSF2-Frameworks realisiert und wird in einem Tomcat Version 7.0 laufen. Die Suchmaschine wird mithilfe des Lucene-Frameworks realisiert. Für das Parsen der RDF-Inhalte wird das Jena-Framework verwendet.

6.2.3 Consuming-Strategien

Im Kapitel 3.3(ab Seite 27) wurden die unterschiedlichen Strategien zur Abfragen der Daten des „Web of Data“ detailliert diskutiert. Verteilte SPARQL-Queries (siehe „Query Federation Pattern“) erweisen sich aufgrund der geringen Performance und Zuverlässigkeit der Schnittstellen nicht als praktikabel für Abfragen, die dem Benutzer sofort und schnell ein Ergebnis liefern können. Der Gegenansatz dazu, das Crawling Pattern, ist ebenso ungeeignet, da hier enorme Kosten für die Aktualität und Wartung der Daten anfallen.

Daher wurde für die Realisierung dieser Applikation eine Mischform beider Ansätze gewählt. Für die Realisierung der Suchmaschine wird das „**Crawling pattern**“ implementiert, für den LD Browser wird das „**On-the-fly dereferencing pattern**“ verwendet und für den explorativen Ansatz werden beide Pattern kombiniert.

Die Daten werden indexiert, um Benutzern eine performante Recherche anbieten zu können. Der Index selbst wird lokal gespeichert, jedoch werden die Daten, die für die Indexierung verwendet werden nicht in einem RDF-Store gespeichert. Somit kann der Bedarf an Ressourcen reduziert werden. Details zur Indexierung werden im Kapitel 6.2.4 ab Seite 85beschrieben.

Der Index ist nur für das effiziente Auffinden von Informationen notwendig. Das Anzeigen der Daten mithilfe des LD Browsers erledigt dann durch den Online-Zugriff auf die Daten über die entsprechenden URIs. Details zum LD Browser werden ebenfalls in Kapitel 6.2.4 (ab Seite 85) beschrieben.

Abbildung 23zeigt die unterschiedlichen Arten des Zugriffs auf die Daten. Die Applikation nutzt drei unterschiedliche Arten von Schnittstellen zum LLD Web:

- URIs für den Direktzugriff für den LD Browser
- RDF-Dumps als Datenbasis für die Initial-Indexierung und URIs für die Ongoing-Indexierung
- SPARQL für die Formulierung von Suchanfragen im Frontend durch versierte Benutzer

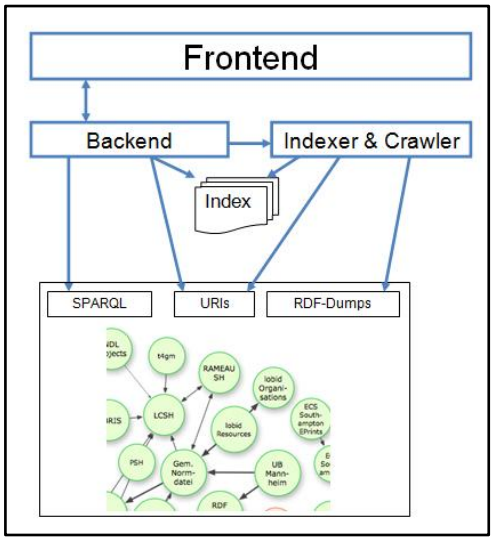

Abbildung 23: Schnittstellen LLD Search zur LD-Web

Aspekte der Datenintegration

URIs unterschiedlicher Datasets, die die gleiche Ressource beschreiben, müssen erkannt und dem Benutzer entsprechend präsentiert werden. In diesem Prototyp wurde „Co-reference-mapping“ noch nicht implementiert. Hier könnten Dienste wie sameAs.org verwendet werden. Bis jetzt sind noch keine der überprüften Datasets verfügbar. Culturegraph.org möchte dieser Dienst für LLD sein. Allerdings existiert zu diesem Service noch keine API, die bei einer maschinellen Verarbeitung berücksichtigt werden könnte.

Die ausgewählten Datasets verwenden unterschiedliche Vokabulare und Ontologien. Einige der Eigenschaften meinen wohl den gleichen Sachverhalt. Das muss den Benutzern der Applikation transparent vermitteln werden. Dazu wurde manuell ein Mapping erstellt, das sowohl bei der Indexierung als auch bei der Darstellung der Informationen im Frontend berücksichtigt wird. Details zum Mapping sind Anhang B (ab S. 119) zu entnehmen.

6.2.4 Software-Architektur

Prinzipiell werden für den Prototypen drei Komponenten unterschieden: „Frontend“, „Backend“ und „Indexer“.

Elemente des Frontends (also des Userinterfaces) werden detailliert in Kapitel Userinterface6.2.6 (ab Seite 91) beschrieben.

Details zum Backend und zur Indexierung werden im Kapitel 6.2.5 ab Seite 87 beschrieben. Zunächst soll aber ein Überblick über die Software-Architektur des gesamten Systems mithilfe der in Kapitel 4.2 (ab Seite 45) beschriebenen Modelle gegeben werden.

Heitmanns Referenzmodell für semantische Webapplikationen (Kapitel 4.2.1 ab Seite 46) kann gut zur Beschreibung der Komponenten von „LLD Search“ und deren Anordnung verwendet werden. Abbildung 24 zeigt die Darstellung der Software-Architektur von „LLD Search“ nach dem Vorbild von Heitmanns Modell. Gegenüber dem Original wurden einige Anpassungen vorgenommen, die nachfolgend beschrieben werden.

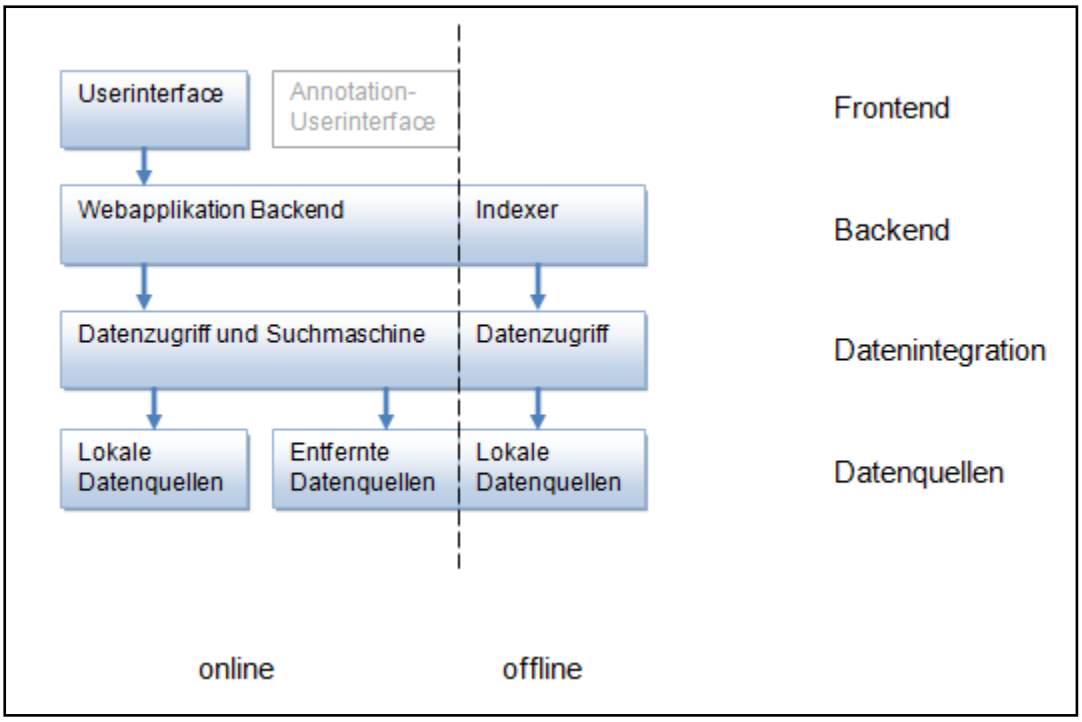

Abbildung 24: Software-Architektur LLD Search (Heitmann Modell)

Zunächst wird zwischen „online“ und „offline“ unterschieden. „Offline“ meint die Vorverarbeitung (bei LLD Search die Indexierung ausgewählten Datenquellen) der Daten. Mit „online“ ist die Verwendung der Webapplikation gemeint. Zu diesem Zeitpunkt stehen schon die Ergebnisse der Vorverarbeitung zur Verfügung und können verwendet werden.

Weiter sind die Komponenten in Schichten angeordnet. Es werden die vier Schichten „Frontend“, „Backend“, „Datenintegration“ und die „Datenquellen“ selbst unterschieden.

In der Schicht „Frontend“ befinden sich die Komponenten „Userinterface“ und „Annotation-Userinterface“, wobei letztere grau dargestellt ist, um zu zeigen, dass diese Komponente derzeit noch nicht verwendet wird. Alle Daten werden bislang nur angezeigt, der Benutzer hat noch keine Möglichkeit Daten zu bearbeiten.

In der Schicht „Backend“ befindet sich das Backend der Webapplikation und der Indexer zur Indexierung der ausgewählten Daten. Beide Komponenten greifen über die Schicht „Datenintegration“ auf die lokalen oder entfernten Daten zu. Diese Schicht übernimmt Aufgaben zum Zugriff auf Daten im RDF-Format als auch auf die Suchmaschine und andere lokale Daten. Zusätzlich zum reinen Zugriff übernimmt diese Schicht zusätzlich Aufgaben der Datenintegration wie Mapping von Vokabularien oder Bewertung der Datenqualität.

Die unterste Schicht der „Datenquellen“ gehört nicht mehr direkt zur Applikation. Sie zeigt nur auf welche Daten zugegriffen wird. Auf lokale Datenquellen wird ausschließlich über APIs zugegriffen. Das betrifft heruntergeladenen RDF-Dumps zur Indexierung als auch auf den Index der Suchmaschine. Auf entfernte Daten wird über APIs (Jena für Direktzugriff der URIs) und über SPARQL zugegriffen.

6.2.5 Backend und Indexierung

Das Backend ist eine JSF2-Applikation, die in einem Tomcat (Version 7) als Servlet-Container läuft. Zusätzlich dazu werden in der Komponente einige Servlets angeboten, die wesentliche Elemente der explorativen Navigation bereitstellen.

Das Backend von „LLD Search“ stellt grundlegende Dienste zur Recherche und Anzeige der Daten im Frontend zur Verfügung. Dazu gehören der Zugriff auf die Indexes der Suchmaschine als auch auf die Daten des LD-Web zur Anzeige der Daten des LD Browsers.Abbildung 25zeigt die grundlegenden Elemente des Backends.

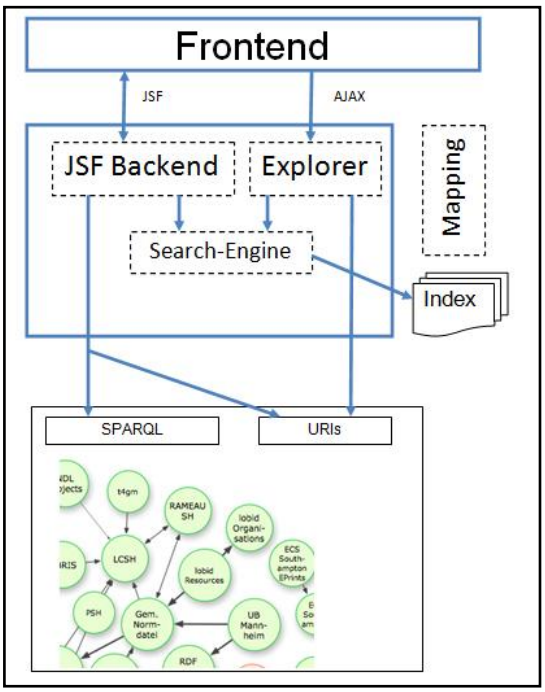

Abbildung 25: LLD Search - Backend-Komponenten

Die Komponente „**Search-Engine**“ stellt Funktionalitäten zu einer Lucene-Suche zur Verfügung. Dazu zählen sowohl das Suchen in bestimmten Feldern eines Indexes als auch eine Suche über alle Felder eines Indexes. Weiter kalkuliert diese Komponente die Facettenklassifikation anhang der ersten 100 Treffer des Suchresultats. Dazu werden alle Werte definierter Felder analysiert und ermittelt wie viele Treffer diese Einschränkung des Ergebnisses auf diesen Wert haben würde.

Die Komponente „**JSF-Backend**“ stellt grundlegende Funktionalitäten der Webapplikation bereit. Dazu gehören die Aufbereitung der Daten für das Frontend, die Weiterleitung der Suchanfragen an die Suchmaschine und der Zugriff auf die LD-Cloud.

Die Komponente „**Explorer**“ stellt weitere Funktionalitäten über zusätzliche Servlets bereit. Diese Servlets liefern JSON als Response-Format aus und sind somit gut für Anfragen via AJAX geeignet. Folgende Anfragen sind derzeit über den Explorer möglich:

- /UriLabel: zur Ermittlung der bevorzugten Bezeichnung einer URI
- /LobidBySubject: zur Ermittlung aller lobid.org Titel anhand eines Schlagworts
- /Narrower: zur Ermittlung aller Unterbegriffe eines Begriffs in der Schlagwortnormdatei

- /NarrowerTagcloud: zur Generierung einer Tagcloud aller Unterbegriffe eines Begriffs der Schlagwortnormdatei
- /NarrowerRVK: zur Ermittlung aller Unterklassen einer Notation der RVK
- /DBPedia: zur Ermittlung des Abstracts und der Kategorien einer Person in der DBPedia
- /BackLinks: zur Ermittlung aller Titeldaten, die dieses Schlagwort verwenden

Die von „Explorer“ bereitgestellten Services können von Applikationen dazu verwendet werden um mit JavaScript zusätzliche relevante Informationen in einer Webseite zu integrieren. Über diese Komponente werden wesentliche Teile der explorativen Navigation in LLD Search realisiert.

Indexierung

Für die Initial-Indexierung wurden RDF-Dumps verwendet. Diese lagen in N-Triples bei lobid.org und in RDF/XML bei GND vor.
Zum Einlesen der Daten wurde das Jena-Framework verwendet. Dazu wird die ganze RDF-Datei in den Arbeitsspeicher eingelesen und dann die Triples verarbeitet. Bei sehr großen Dateien ist diese Arbeitsweise nicht möglich, da große Dateien aufgrund von begrenzten Arbeitsspeicher-Ressourcen nicht verarbeitet werden können. Daher wurden große RDF/XML-Dateien vorab mithilfe eines Perl-Skripts aufgesplittet, so dass jede Ressource-Description in einer einzelnen kleinen XML-Datei gespeichert wurden.

Als Indexierungsart wird eine Schema-Level Indexierung verwendet (siehe Kapitel 3.4.4 ab Seite 37). Alle Aussagen zu einem Subjekt werden somit in einem Lucene „Document“ gespeichert. Die URI des Subjekts wird im Feld „URI“ indexiert. Die Prädikate der Aussagen werden als weitere Felder und die zugehörigen Objekte als Werte dieser Felder indexiert. Ist das Objekt kein Literal, beispielsweise eine Liste, wird diese aufgelöst und in mehreren Instanzen der Felder indexiert. Literale werden mit dem Lucene StandardAnalyzer analysiert und indexiert, URIs werden als „not-analyzed“ indexiert, so dass die URI nicht zerlegt wird.

Zu jedem indexierten Dokument werden Provenance-Informationen gespeichert (Herkunft – Name des Datasets, Datum der erstmaligen Indexierung, Datum des Updates des Dokuments im Index).

Das nachfolgende Beispiel zeigt schematisch diese Indexierung:

Ressource *http://lobid.org/resource/BT000000626 aus lobid.org-Dataset:*

```
@prefix bibo:    <http://purl.org/ontology/bibo/> .
@prefix dcterms:  <http://purl.org/dc/terms/> .
@prefix rdf:      <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix foaf:     <http://xmlns.com/foaf/0.1/> .
@prefix geo:      <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix geonames: <http://www.geonames.org/ontology#> .
@prefix owl:      <http://www.w3.org/2002/07/owl#> .
@prefix frbr:     <http://purl.org/vocab/frbr/core#> .

<http://lobid.org/resource/BT000000626>
dcterms:language"ger" ;
rdf:type dcterms:BibliographicResource ;
rdf:type bibo:Book ;
dcterms:title "Freizeitkarte, leisure map, carte loisirs Ennepe-
Ruhr-Kreis" ;
dcterms:issued "1993" ;
dcterms:extent "1 Kt. : mehrfarb. ; 47 x 55 cm, gefaltet" ;
bibo:isbn "3-8164-0500-2" ;
dcterms:subject <http://d-nb.info/gnd/4014819-1> ;
dcterms:subject <http://d-nb.info/gnd/4155353-6> ;
dcterms:publisher [
rdf:type foaf:Organisation ;
foaf:name "St\u00E4dte-Verl. v. Wagner u. Mitterhuber" ;
geo:location [
rdf:type geo:SpatialThing ;
geonames:name "Fellbach b. Stuttgart" ;
]
];
rdf:type frbr:Manifestation .
```

Dieser Datensatz wird in Lucene als „Document“ wie folgt gespeichert:

Feld	***Wert***	***analyzed?***
URI	*http://lobid.org/resource/BT00000626*	*Nein*
dcterms:language	*ger*	*Ja*
rdf:type	*http://purl.org/dc/terms/BibliographicResource*	*Nein*
rdf:type	*http://purl.org/ontology/bibo/Book*	*Nein*
dcterms:title	*Freizeitkarte, leisure map, carte loisirs Ennepe-Ruhr-Kreis*	*Ja*
dcterms:issued	*1993*	*Ja*
dcterms:extent	*1 Kt. : mehrfarb. ; 47 x 55 cm, gefaltet*	*Ja*

bibo:isbn	*3-8164-0500-2*	*Ja*
dcterms:subject	*http://d-nb.info/gnd/4014819-1*	*Nein*
dcterms:subject	*http://d-nb.info/gnd/4155353-6*	*Nein*
dcterms:publisher	*St\u00E4dte-Verl. v. Wagner u. Mitterhuber, Fellbach b. Stuttgart*	*Ja*
rdf:type	*http://purl.org/vocab/frbr/core#Manifestation*	*Nein*
prv:create	*Datum der Initialindexierung*	*Nein*
prv:update	*Datum des letzten Updates*	*Nein*

Jedes Dataset wird in einem eigenen Index gespeichert. So können Besonderheiten einzelner Datenquellen besser berücksichtigt werden.

Dataset Selection und Discovery

Die Datasets für diesen Prototyp wurden manuell aufgrund der Analyse der derzeit existierenden Datasets in diesem Bereich ausgewählt.

Im LD Browser der Applikation werden alle „neuen" URIs vermerkt, über die der Benutzer während der Navigation „stolpert". Diese neuen URIs werden dann für die nächste Indexierung via LDSpider ge-crawled und bei der Indexierung berücksichtigt.
Somit lernt das System durch Benutzerverhalten und erweitert ständig den Suchrahmen auf eventuell zusätzlich relevante Ressourcen (oder Datasets).

SPARQL-Query

Zur Durchführung der verteilten SPARQL-Queries der Anwender wird das Jena Framework verwendet.

Datenqualität

Aspekte der Datenqualität, z.B. zum Ranking der Ergebnisse, werden derzeit nicht beachtet – könnten aber zukünftig berücksichtigt werden. Zusätzlich zu automatischen Verfahren, z.B. der Komponente trdf des Jena-Frameworks, könnten Interaktionen mit den Anwendern dazu verwendet werden, um zu entscheiden, welche Datasets die „richtigen" Ergebnisse liefern und somit am Vertrauenswürdigsten sind.

6.2.6 Userinterface

Im Folgenden werden die Features des Frontends anhand der drei in Kapitel 6.2.1 (ab Seite 81) geschilderten Anwendungsfälle dargestellt.

Das Userinterface besteht grundlegend aus drei Bereichen:

- Startseite
- Kurztrefferlist
- Vollanzeige

Die „Startseite“ bildet den Einstiegspunkt in eine keyword-basierte Suche oder eine Suche mittels SPARQL. Die „Kurztrefferliste“ zeigt jeweils die Ergebnisse der Suche an und in der „Vollanzeige“ werden sämtliche Informationen zu einer Ressource in Form eines Mashups angezeigt.

Anwendungsfall A

Die Startseite gilt als Einstiegpunkt in die Webapplikation. Die keyword-basierte Suche wird als Default-Einstellung angeboten. Gleichsam ist es aber möglich in den „Experten“-Modus umzuschalten. Abbildung 26 (auf der nächsten Seite) zeigt ein Screenshot dieser Startseite, in der die Hauptbereiche markiert sind.
Bereich „1“ ist die Navigationsleiste, mit der zwischen den unterschiedlichen Modi gewechselt werden kann.
Bereich „2“ ist das eigentliche Suchformular. Hier kann sowohl der Index als auch das Feld in dem gesucht werden soll, ausgewählt werden.
Bereich „3“ ist ein Informationsbereich, wo dem Benutzer allgemeine Informationen angezeigt werden können, z.B. in welchen Datenquellen gesucht werden kann.

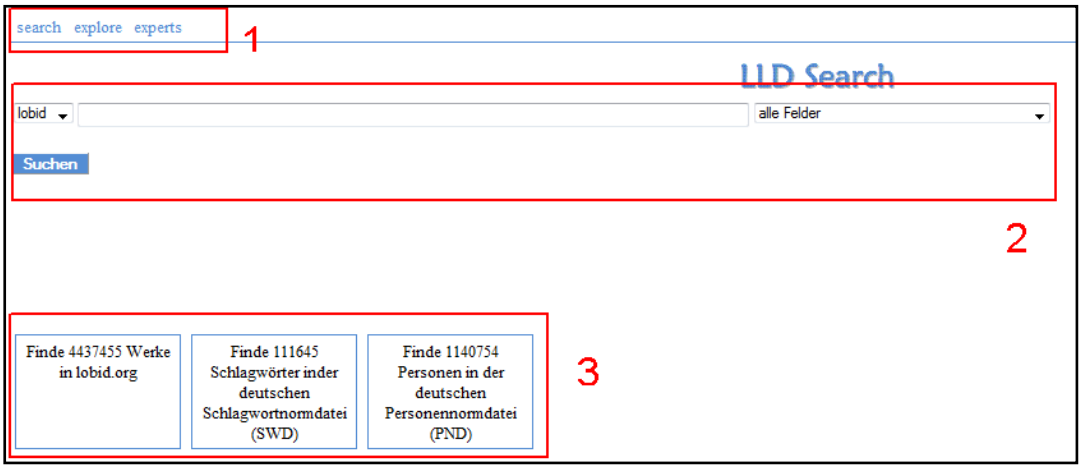

Abbildung 26: Screenshot LLD Seach - Startseite

Kann die vom Benutzer gestellte Suchanfrage erfolgreich durchgeführt werden, wird eine Kurztrefferliste generiert. In dieser wird eine Ergebnisliste von 20 Resultaten (Ressourcen) pro Seite mit der Möglichkeit zum vor- und zurückblättern angeboten. Die angezeigten Felder pro Treffer sind: Autor, Titel, Erscheinungsjahr und Schlagwörter. Zudem wird eine Facettenklassifikation vordefinierter Felder Autor, Erscheinungsjahr, Sprache, Verlag und Schlagwort

angeboten. Diese Facetten werden basierend auf den ersten 100 Treffern der Ergebnisliste kalkuliert.[54]

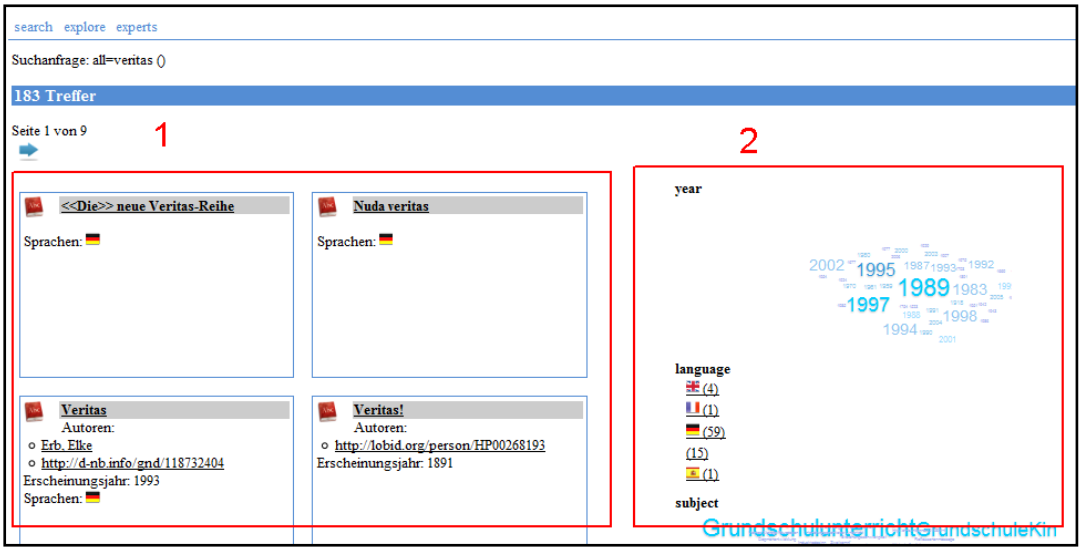

Abbildung 27: LLD Search Kurzanzeige

Ein Klick auf einen Treffer führt zur Vollanzeige, die dann im Stile eines LD Browsers die Informationen der Ressource live holt, aggregiert und anzeigt.Abbildung 28 zeigt beispielhaft eine Vollanzeige.

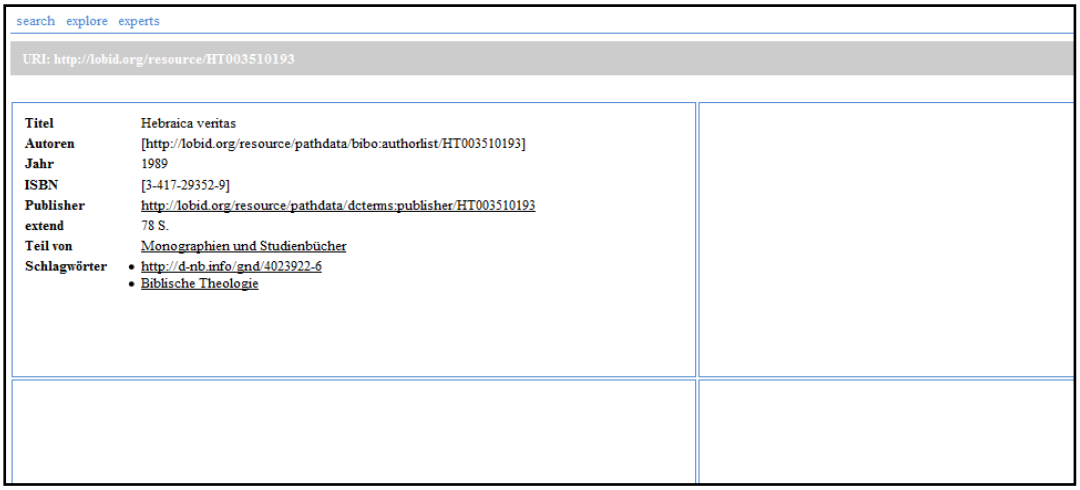

Abbildung 28: LLD Search Vollanzeige

Diese enthält neben den Informationen zum Titel auch weitere Informationen zur explorativen Navigation: Orte, mehr Informationen zu den Autoren und mehr Informationen im Kontext der Schlagwörter.

[54]Inspiriert vom Algorithmus der Facettenbildung des Exlibris-Produkts Primo.

Für die Vollanzeige werden die anzuzeigenden Informationen live geholt, aggregiert und präsentiert. Zunächst werden die Informationen der URI geholt. Sind in der Beschreibung der URI weitere RDF-Links enthalten werden diese maximal zwei Ebenen weiter verfolgt. Dadurch können womöglich relevante Informationen zur angezeigten Ressource aufgrund der Verlinkung des LLD automatisch mitberücksichtigt werden.
Für die Integration von explorativen Elementen werden die Explorer-Services des Backends verwendet, um zusätzliche Querverbindungen zwischen den Daten zu ermitteln.

Anwendungsfall B

Die Schritte keyword-basierte Suche, Kurzanzeige und Vollanzeige sind auch in diesem Fall identisch. In der Beschreibung wird nur noch auf Unterschiede zum Anwendungsfall A eingegangen.

Beim Startpunkt der keyword-basierten Suche muss bei diesem Anwendungsfall im Index „pnd“ oder „swd“ gesucht werden. Abbildung 29 zeigt die Kurzanzeige bei der Suche nach Personen. Für jede Ressource werden Name der Person, Lebensdaten, Berufe und weiterführende Links angezeigt.

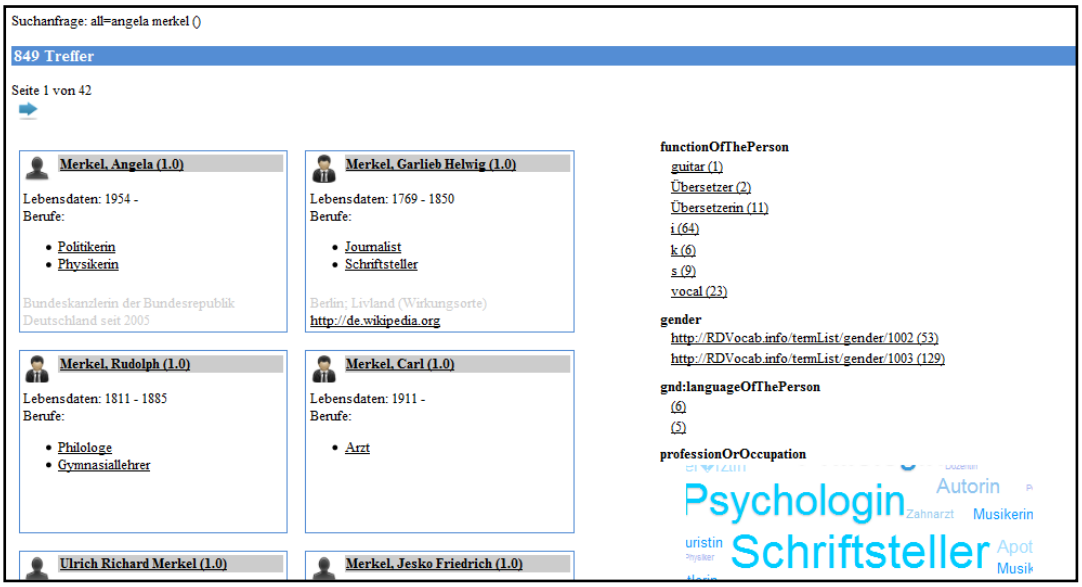

Abbildung 29: LLD Search Kurzanzeige - Suche in PND

Abbildung 30(auf Seite 95) zeigt die Vollanzeige der Ressource „Angela Merkel“ in der PND.

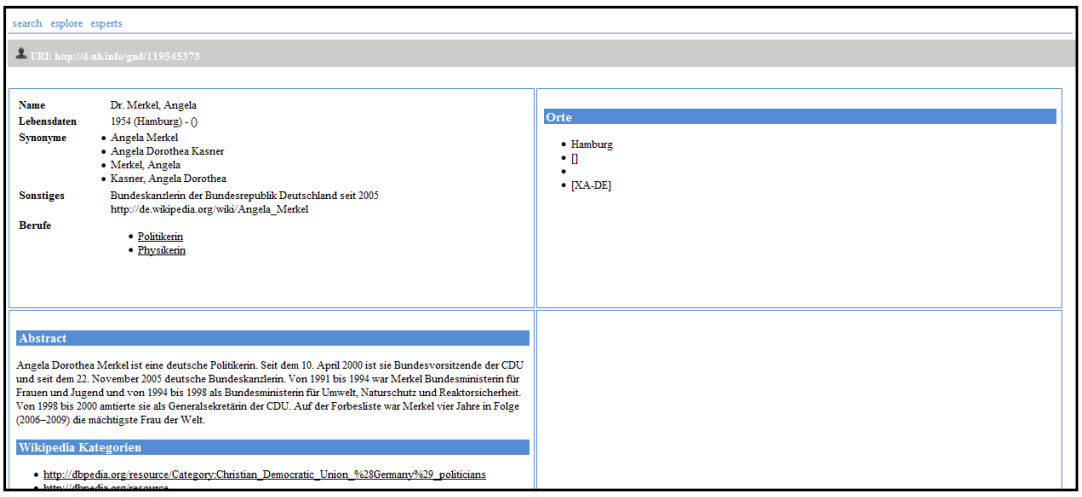

Abbildung 30: LLD Search Vollanzeige "Angela Merkel"

Die Vollanzeige, ähnlich wie bei der Vollanzeige der Titeldaten, ist in vier Bereiche unterteilt:

- Informationen zur Ressource
- Orte
- Verwandte Themen
- Publikationen der Person

Die Bereiche „Informationen zur Ressource“, „Orte“ und „Publikationen der Person“ werden aus dem Datensatz der Ressource direkt durch Zugriff auf die URI geholt. Die Informationen „verwandte Themen“ wird durch Verfolgung weiterführender Links generiert. In diesem Beispiel werden Informationen der DBPedia abgeholt:

- deutschsprachiges Abstract der Wikipedia und
- Kategorien

Das Abstract ermöglicht eine Kurzeinführung in diese Ressource. Über die Kategorien kann diese Ressource besser im Kontext eingeordnet werden. So können Personen in Erfahrung gebracht werden, die beispielsweise auch in Hamburg geboren sind, oder Personen, die ebenfalls den Karlspreis erhalten haben.

Analog zu Personen kann auch nach Schlagwörtern der SWD gesucht werden. Abbildung 31(auf Seite 96) zeigt die Kurzanzeige einer Suche nach „Informatik“ im Index „swd“.

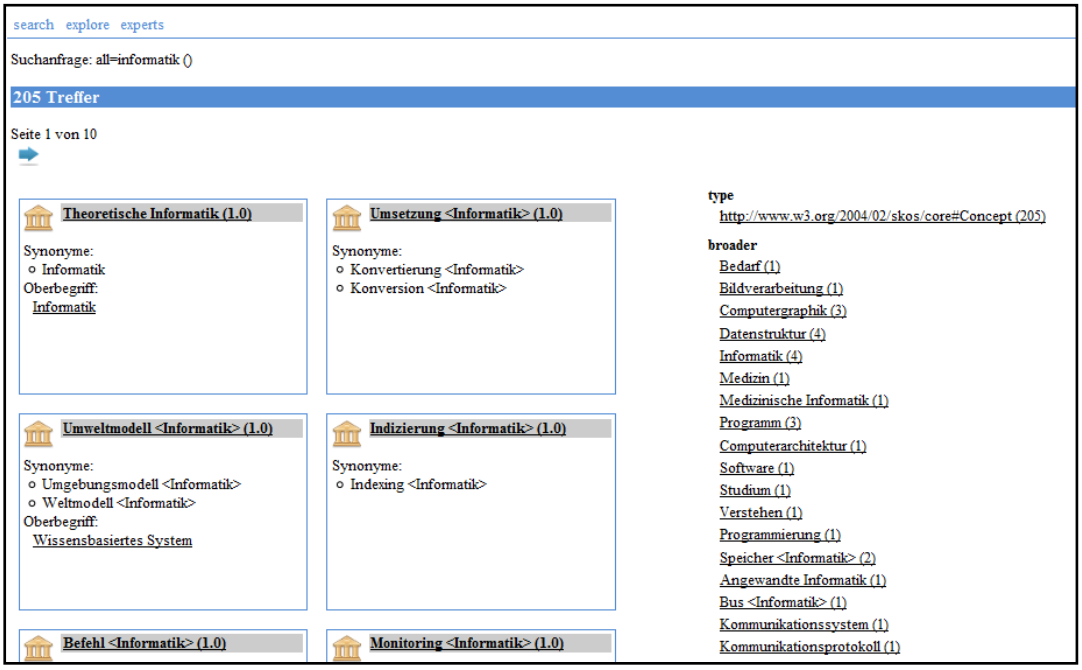

Abbildung 31: LLD Search Kurzanzeige "Informatik"

Hier werden neben dem bevorzugten Namen der Ressource auch Synonyme und Oberbegriffe angezeigt. Die Vollanzeige der Ressource ist wieder in die vier bekannten Bereiche aufgeteilt. Abbildung 32 zeigt die Vollanzeige der Ressource „Informatik“. Im Bereich „verwandte Themen“ werden Ober- und Unterbegriffe angezeigt. Damit kann ein Benutzer leicht verwandte Themen erforschen. Unterbegriffe werden als Tagcloud angezeigt. Bei jedem Begriff wird in Klammern angegeben, wie viele Ressourcen in lobid.org mit diesem Schlagwort verlinkt sind.

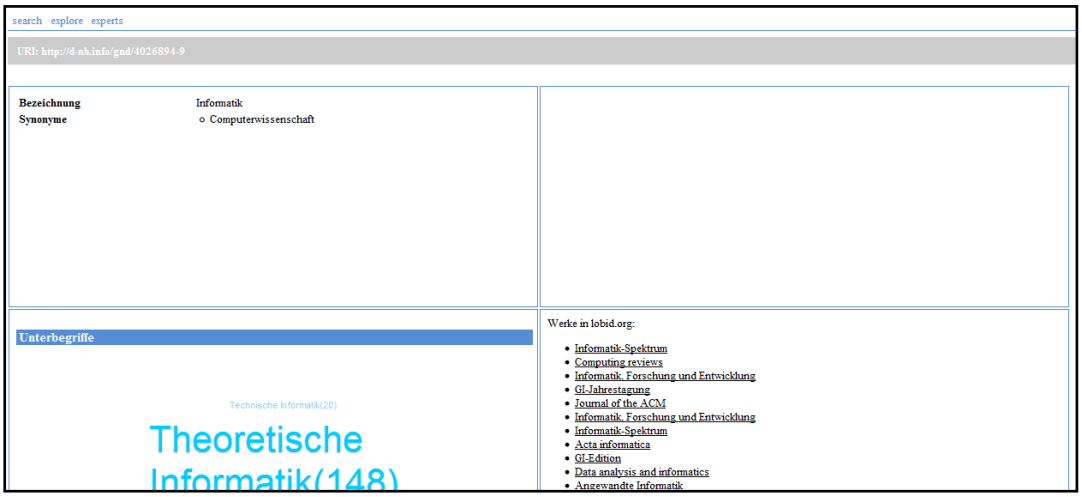

Abbildung 32: LLD Vollanzeige "Informatik"

Der vierte Bereich zeigt eine Auswahl von Titeln in lobid.org, die mit diesem Schlagwort verlinkt sind.

Anwendungsfall C

Die SPARQL-Suche kennzeichnet sich durch ein großes Eingabefester zur Eingabe einer SPARQL-Query. Abbildung 33 zeigt diese Startseite. (Künftig können hier auch SPARQL-Builder verwendet werden, um den Anwendern die Eingabe zu erleichtern.)

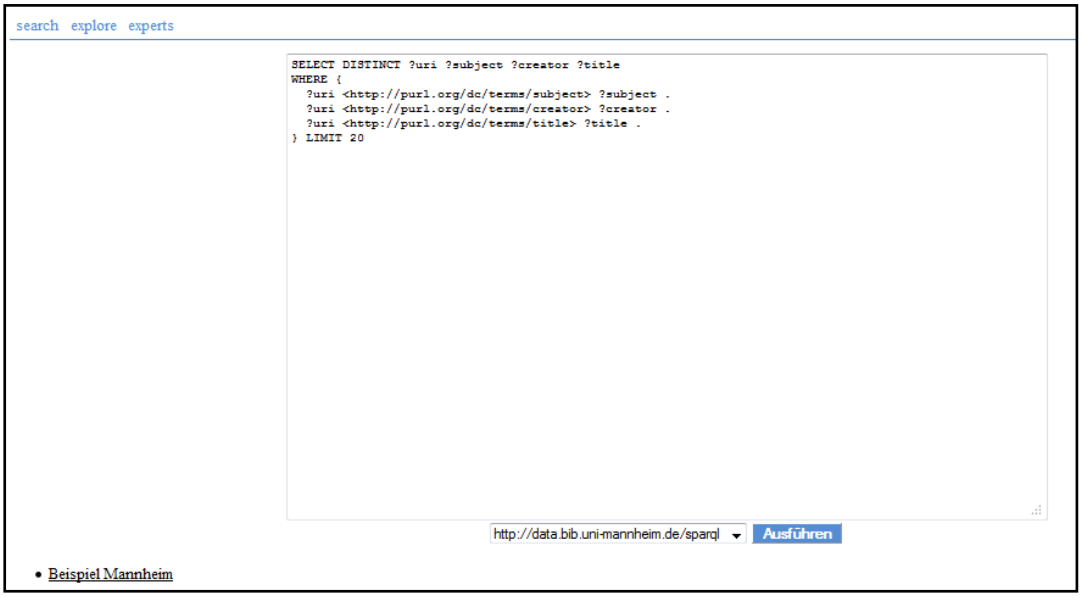

Abbildung 33: LLD Search Expertensuche

In der Liste unterhalb der Textbox sind einige SPARQL-Interfaces angeführt, die bibliografische Daten anbieten. Der Link „Beispiel Mannheim“ fügt eine Beispielabfrage in die Textbox ein, um dem Benutzer zu vermitteln, wie diese Suche verwendet werden kann.

Die Vollanzeige diese SPARQL-Query ist in Abbildung 34 (auf Seite 97) zu sehen. Sind URIs in der Ergebnismenge enthalten, wird der bevorzugte Name dieser URIs aufgelöst und die Links werden mit der Vollanzeige von LLD Search verlinkt.

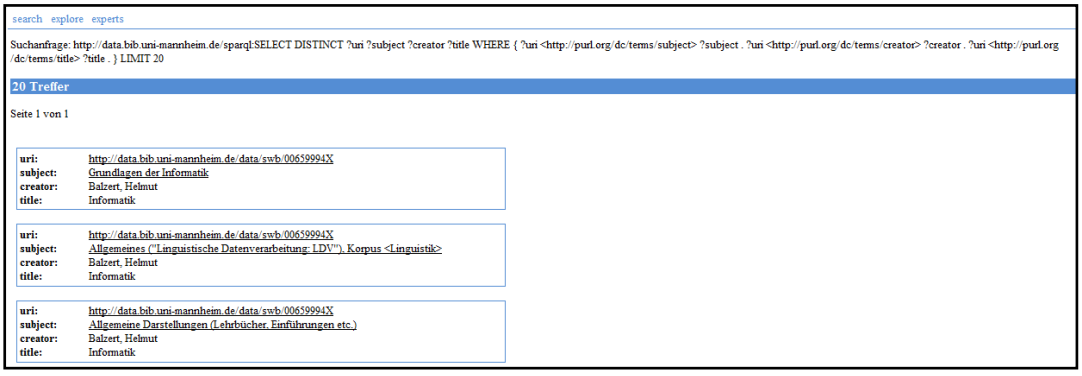

Abbildung 34: LLD Search Kurzanzeige einer SPARQL-Query

LD Browser

Die Funktionalität der Vollanzeige wurde bisher nur an Beispielen gezeigt, die über die Suchmaschine gefunden wurden. In dieser Vollanzeige werden dabei alle Informationen einer URI geholt und angezeigt. Sind in diesen Informationen selbst wieder URIs enthalten, wird dieser Link wiederum in der Vollanzeige angezeigt. Diese Daten werden tabellarisch, wie in einem typischen LD Browser angezeigt. Ein Beispiel zeigt Abbildung 35:

Abbildung 35: LLD Search Vollanzeige (LD Browser)

6.2.7 Fazit und Future Work

In diesem Prototyp wurden einige Aspekte der Implementierung einer LD Webapplikation ausgetestet und mögliche Lösungswege aufgezeigt. Diese Implementierung zeigt aber auch eindeutig, dass aus den vorhandenen Daten eine bedienungsfreundliche Applikation nicht auf Knopfdruck entwickelt werden kann. Derzeit sind noch einige Schritte notwendig, die einiger intellektueller Vorarbeit bedürfen. Im Großen und Ganzen kann aber gesagt werden, dass die semantische Reichhaltigkeit der Daten einen enormen Mehrwert bietet, der ohne diese Daten nur kaum zu erreichen ist.

Dieser Prototyp soll die Möglichkeit aufzeigen, wie es möglich ist eine LD Webapplikation durch die Abfrage des „Web of Data“ zu realisieren. Er soll bestehenden integrierten Bibliotheksmanagement-Systemen keine Konkurrenz machen. Vielmehr ist es aber möglich die Vorteile der Reichhaltigkeit der Daten des LLD Web auch in diesen Applikationen zu nutzen, beispielsweise durch (bessere) explorative Userinterfaces.

Future Work

Diverse Verbesserungen können an „LLD Search“ noch durchgeführt werden, um daraus eine produktionsreife Webapplikation zu machen.

Indexierung:

So müsste zum Beispiel der ganze Prozess des crawlen, aufbereiten und Indexierens verbessert und automatisiert werden. Lucene SiREN und SOLR für RDF-Daten sollten dabei evaluiert werden.

Verlinkung:

Anreicherungsmethoden und Verlinkungen könnten dadurch verbessert werden, dass lokale Daten analysiert werden, beispielsweise mit Tools wie dem SILK-Server oder der Verwendung von co-reference-Diensten.

Datenquellen:

Sollten sich semantische Indexes wie sindice.com weiter etablieren und mehr Daten (also auch des LLD) beinhalten, können diese für die Recherche verwendet werden. Somit müssten die Daten nicht mehr selbst ge-crawled und indexiert werden.

Grundlage für die Auswahl weiterer SPARQL-Interfaces könnte [50] sein. Diese Webseite zeigt die Verfügbarkeit öffentlich zugänglicher SPARQL-Endpoints an.

Userinterface:

Explorative Methoden wurden nur in kleinem Ausmaß ausgetestet. Diese könnten noch sehr viel umfangreicher und vielfältiger implementiert werden.

Explorative Suchmethoden könnten noch um einiges mehr an Mehrwert bringen, wenn Reasoning und Inference bei der Kalkulation berücksichtigt werden.

Eine weitere Art der Benutzer-Interaktion ist die Mitbestimmung der Anwender darüber, welche Datasets am relevantesten sind. Der Benutzer hat die Möglichkeit bestimmte Treffer in der Ergebnisliste als „nicht relevant“ zu markieren. Das Dataset erhält somit Negativ-Punkte und wird bei der nächsten Indexierung im Boosting-Faktor berücksichtigt.

Letztlich stellt sich die Frage wie sich das LLD Web weiter entwickeln wird. Um wirklich gute Applikationen zu implementieren ist es wichtig, qualitativ hochwertige Daten und Beziehungen zu haben und zu nutzen. Erst dann wird es möglich einen wirklichen Mehrwert aus diesen Daten zu ziehen.

7 Zusammenfassung

Das Web ist in vielen Bereichen des privaten und beruflichen Umfelds ein essentieller Informationslieferant geworden. Es werden immer mehr Informationen im Web publiziert. Jedoch fällt es zunehmend schwerer die Informationen zu finden, die gesucht werden. Klassische Volltextsuchmaschinen gelangen dabei immer mehr an ihre Grenzen, da hier existierende Dokumente lediglich nach Begriffen „gescannt“ werden.

Semantic Web und Linked Data

Um bessere Suchresultate zu liefern, ist es notwendig, dass ein Computerprogramm (hier Suchmaschine) den Suchbegriff in einen Kontext einordnen kann.

Beispiel: Ist mit der Suchanfrage „Java“ die Programmiersprache, die Insel oder der Modetanz der 1920er Jahre gemeint?

Soll ein Computerprogramm diesen Kontext einordnen, ist es notwendig die Informationen mithilfe von Metainformationen zu beschreiben. Die Technologien und Protokolle des Semantic Web stellen diverse Möglichkeiten zur semantischen Beschreibung der Informationen bereit.

Ein Teilbereich des Semantic Web, Linked Data, stellt Richtlinien zum Publizieren semantisch aufbereiteter Daten im Web bereit. Informationsprovider stellen Informationen zu einem Konzept über eine eindeutig dereferenzierbare HTTP URI bereit. Die Informationen werden mithilfe von Vokabularien und Ontologien in RDF semantisch beschrieben. Der zentrale Aspekt dieses „Konzepts“ ist die typisierte Verlinkung zwischen Informationen. Dadurch kann ein wirklicher Mehrwert gewonnen werden, der ohne die semantische Beschreibung der Beziehungen nicht möglich wäre.

Beispiel: Recherchiert ein Anwender nach einer Publikation eines Autors, sind Publikationen der Co-Autoren dieser Publikation wahrscheinlich auch relevant. Es ist davon auszugehen, dass die Autoren einen ähnlichen Forschungsbereich haben, wenn sie eine Publikation gemeinsam schreiben.

Bibliotheken und Linked Data = Library Linked Data

Bibliotheken hatten in der Vergangenheit weniger mit der Informationsvielfalt im Web, sondern eher mit der mangelnden Sichtbarkeit ihrer Dienstleistungen im Web zu kämpfen. [51], S. 3f

Aufgrund der Probleme proprietärer Bibliotheksmanagementsysteme war es in der Vergangenheit schwierig, bei der Recherche nach bibliografischen Daten in den Google-Ergebnislisten auf den ersten Seiten vertreten zu sein. Hier dominieren zumeist Amazon und andere Webshops. Webseiten von Bibliotheken oder gar Bibliothekskataloge scheinen hier nicht auf und werden von potenziellen Anwendern somit nicht gefunden.

Bibliotheken erwarten sich durch die Bereitstellung ihrer Daten als Linked Data eine bessere Sichtbarkeit im Web und somit eine bessere Nutzung ihrer Dienstleistungen. Zudem können durch die Reichhaltigkeit der semantisch-beschriebenen Daten bessere User-Interfaces zur Recherche bibliografischer Daten entwickelt werden.[51], S. 3f[52], S. 10f

In der Arbeit wurde der aktuelle Stand (11.03.2011) der Domain *Library Linked Data* untersucht. Von den rund 860.000.000 Aussagen dieser Domain entfallen rund 66% auf Titeldaten und Bestandsnachweise, 30% auf Normdaten und Thesauri und lediglich 4% auf Sammlungen digitaler Objekte.

Gerade im Bereich des Bibliothekswesens werden ähnliche, wenn nicht gleiche bibliografische Informationen bei vielen Bibliotheken (oder Bibliotheksverbünden) separat gespeichert. Würde der Großteil dieser Bibliotheken ihre Daten als Linked Data veröffentlichen, stellt sich das Problem des „co-reference mappings“. Nach den Richtlinien von Linked Data müssten alle diese Ressourcen untereinander verlinkt werden.
Möchte nun eine Applikation oder ein Anwender diese Information verlinken stellt sich die Frage, welche der identischen Ressourcen verlinkt werden soll. Daher ist es notwendig einen einheitlichen Einstiegspunkt in dieses „Netz“ zu bieten. „Culture Graph[55]“ möchte dieser Einstiegspunkt sein.

Consuming Linked Data

Linked Data kann über unterschiedliche Schnittstellen bereitgestellt werden:

- direkt via HTTP URI
- via SPARQL-Interface
- als Dump
- via APIs

Abhängig von den Anforderungen einer Anwendung, die die so bereitgestellten Daten nutzen möchten, werden mehrere Consuming-Strategien unterschieden:

- Crawling pattern

[55] http://culturegraph.org[zuletzt angesehen 17.05.2011]

- On-the-fly dereferencing pattern
- Query federation pattern

Das „Crawling pattern“ beschreibt einen Datawarehousing-Ansatz. Alle Quellen werden vorab gecrawled und in einer oder mehreren Datenbanken lokal gespeichert. Abfragen werden dann gegen diese Datenbank gemacht.

Das „On-the-fly-dereferencing pattern“ beschreibt eine Strategie, bei der zur Zeit der Benutzeranfrage eine URI aufgerufen wird. Dann können manuell oder automatisch die Links verfolgt werden.

Das „Query Federation Pattern“ beschreibt eine verteilte SPARQL-Abfrage, die mehreren Datenquellen parallel abfragt.

Jede der drei Strategien hat Vor- und Nachteile. Je nach Anwendungsfall ist die beste Strategie zu wählen. In den meisten Fällen wird es wohl zur Kombination der Strategien kommen müssen. Kriterien für die Auswahl sind:

- Anzahl der Datenquellen,
- Grad der Aktualität,
- Antwortzeiten,
- Auffinden neuer Datenquellen zur Laufzeit

Weitere Aspekte der Datenintegration sind Mapping von Vokabularien, Mapping von Ressourcen, Bewertung der Datenqualität und Ranking.

Es ist zu erwarten, dass zusätzlich zu den Linked Data Produzenten (und Providern) und den Linked Data „Consumer“ auch noch eine Schicht der „Aggregatoren“ etabliert wird. Diese könnten Daten unterschiedlicher Quellen als SPARQL-Service oder als Suchmaschinen-Index anbieten, um Consumern eine performantere Schnittstelle zum „Web of Data“ bereitzustellen.

Linked Data Webapplikationen

Zu Beginn der Linked Data Bewegung wurden Linked Data Browser präsentiert, die es ermöglichen Informationen der Linked Data Datasets für einen menschlichen Leser lesbar zu machen. Diese Art von Webapplikationen verlangt aber ein bestimmtes Vorwissen der Anwender in Bezug auf die Datenstruktur und der verwendeten Vokabularien.

Daher ist es notwendig benutzerfreundliche Webapplikationen für den Enduser zu entwickeln, um die Informationen aus dem Linked Data Web für jedermann einfach zugänglich zu machen. Um das zu erreichen sollten lediglich relevante

Informationen aggregiert und angezeigt und vernachlässigbare Informationen verborgen werden.

Für die Implementierung neuer Userinterfaces, ist es wichtig die Entwicklung von Suchanforderungen zu kennen. Diese reichen von einer umfangreichen keyword-basierten Volltextsuche bis hin zu einer explorativen Suche.
Damit wird dem Benutzer ermöglicht, nicht nur Informationen zu Begriffen, die er oder sie bereits kannte, zu finden, sondern es werden zusätzlich Methoden des „Learning“ und der „Investigation“ einbezogen. Ein erster Schritt in diese Richtung ist die facettierte Navigation, wie sie mittlerweile schon in vielen Suchmaschinen angeboten wird. Facetten werden aber bisher nur aus den Ergebnissen der Suchtreffer zu einem Begriff gebildet. Die Reichhaltigkeit des LD Webs bildet die Grundlage zu einer wirklichen explorativen Suche. Die Beziehungen zwischen Ressourcen (durch Links) ermöglichen es, dem Benutzer verwandte Begriffe anzubieten und so alternative Recherchepfade bereitzustellen.

Bei der Implementierung von Linked Data Webapplikationen können Pattern der Entwicklung von Webanwendungen genutzt werden. Zusätzlich dazu sind eine Menge von Herausforderungen zu bewältigen, die die Architektur von Linked Data mit sich bringt – zusammengefasst ein paar Beispiele:

- Auffinden und Auswählen von relevanten Datasets in einem sich ständig ändernden Netz von Informationen
- Bewertung der Datenqualität von Datasets (oder einzelnen Ressourcen) zur Bewertung der Relevanz
- Unterschiedliche Anforderungen der Benutzer an Recherchesysteme
- Performanz der Suchanfragen
- Verfügbarkeit und Zuverlässigkeit der Schnittstellen
- Erkennung von ähnlichen Informationen zur Aggregation (co-reference-mapping und Mapping von Vokabularien)

Zusätzlich zu etablierten Pattern für die Implementierung von Webapplikationen werden für Linked Data insbesondere Komponenten zur Datenintegration benötigt. Diese ermöglichen einen einheitlichen Zugriff auf die in unterschiedlichen Formaten und Technologien publizierten Daten im Linked Data Web. Abhängig davon, welche Consuming-Strategie gewählt wird, sind hier Lösungen zu folgenden Aspekten zu implementieren:

- Lokale (temporäre oder persistente) Speicherung der Daten
- Mapping von Vokabularien, damit über ein Schema auf die Daten zugegriffen werden kann
- Mapping von Ressourcen (co-reference-mapping), damit die Daten gut aggregiert werden können

- Indexierung der Daten, damit performant auf die Daten zugegriffen werden kann
- Bewertung der Datenqualität, damit ein Ranking der Ergebnisse ermöglicht werden kann und eventuell weniger vertrauenswürdige Daten dem Benutzer verborgen werden

Die Anzahl der Verlinkungen im Linked Data Web ist in vielen Bereichen noch recht schwach. Daher ist es oftmals notwendig, beim Consuming der Daten eine lokale Linking-Komponente zu implementieren.

Sollten in der Linked Data Applikation neue Informationen (beispielsweise durch Verlinkung oder Reasoning) generiert werden, empfiehlt es sich, diese Daten selbst auch wieder als Linked Data Dataset zu veröffentlichen.

Prototyp

Es wurde eine Linked Data Webapplikation entwickelt, die ausgewählte Datasets der Library Linked Data Domain zur Recherche anbietet. Die Applikation wurde mit Hilfe folgender Technologien realisiert:

- Jena Framework zum Verarbeiten von RDF Daten
- Lucene zum Entwickeln einer Suchmaschine
- JSF als Framework zur Entwicklung der Webapplikation

Die Webapplikation („LLD Search“) ermöglicht dem Benutzer (vorerst manuell ausgewählte) Linked Data Datasets der Domain Library Linked Data zu durchsuchen und explorativ zu „erforschen“.

Im Backend der Applikation wurde das Crawling-Pattern (Crawling von RDF-Dumps) implementiert, um so einen Lucene-Index der Triples der ausgewählten Daten aufzubauen. Teile des Backends sind so realisiert worden, dass es auch möglich ist Client-Abfragen via AJAX asynchron zu stellen.

Das Mapping von Vokabularien für die Indexierung wurde manuell für die ausgewählten Datasets durchgeführt und in einer Konfigurationsdatei gespeichert.

Das Mapping von Ressourcen konnte nicht durchgeführt werden, da Mapping-Dienste wie CultureGraph noch über keine API verfügen, die maschinell verwendet werden kann. Somit werden „Dubletten“ weder auf Indexierungsebene noch auf User-Ebene erkannt.

Das Ranking basierend auf der Datenqualität wurde im Prototyp bisher noch nicht realisiert. Die in der Arbeit diskutierten Aspekte der Datenqualität könnten aber dazu verwendet werden, um ein Ranking zu implementieren.

Das Recherche-Interface ermöglicht Benutzern mit unterschiedlichen Anforderungen in LLD Search zu recherchieren.

- Auffinden von bekannten Informationen (lookup durch keyword-basierte Suchen)
- Exploratives Auffinden von Informationen (Browsing in Tagclouds, Timelines und Maps)
- Verteilte SPARQL-Queries, um entfernte SPARQL-Interfaces abzufragen

Die Ergebnisse werden klassisch in Form einer Ergebnisliste mit Facettenklassifikation angezeigt.

Die Vollanzeige eines Treffers (also eine Ressource) wird in Form eines Linked Data Browsers mit explorativen Ansätzen realisiert. Alle weiterführenden RDF-Links werden somit auch in dieser Umgebung angezeigt und entsprechend aufbereitet.

Die durch Anwender verfolgten Links werden mitgeloggt. Diese Logdateien dienen als Grundlage für die Backend-Komponente und werden beim nächsten Indexierungsprozess mit indexiert. Somit lernt das System mit der Verwendung durch die Benutzer weitere (eventuell) relevante neue Datasets automatisch.

Die Indexierung wird vorab (nicht zum Zeitpunkt der Recherche durch die Benutzer) durchgeführt. Durch die Vorab-Indexierung kann den Benutzern eine performante Anwendung zur Verfügung gestellt werden.
Zur Anzeige der Daten in der Vollanzeige werden die Daten online abgeholt (mit kleinen Caching-Mechanismen zur Reduktion des Datentransfers) und dem Benutzer präsentiert. Während der Implementierung ist aufgefallen, dass die online verfügbaren Daten meist reichhaltiger sind als die indexierten. Daher wird jeder vom Anwender aufgerufene Satz bei der nächsten Indexierung gecrawled und erneut indexiert.

Fazit

Die Fragestellung nach der geeignetsten Strategie für die Nutzung von Linked Data in Webapplikationen kann nicht eindeutig beantwortet werden. Abhängig von den Anforderungen der Anwendungen kann eine andere Strategie gewählt werden. Hier wird zwischen Strategien unterschieden, die sämtliche Daten lokal

speichern und Strategien, die auf die derzeit verfügbaren Daten im Linked Data Web zugreifen.
Aufgrund von Performance-Problemen und der Verfügbarkeit von Schnittstellen zu den Linked Data Datasets im Web werden oft crawling Strategien verfolgt, um Abfragen der Daten möglichst performant anbieten zu können.
Wenn künftig Indexes oder Aggregator-Services für Linked Datasets angeboten werden, ist gut vorstellbar, dass in Zukunft eher Strategien eingesetzt werden die Live auf die Daten im Web zugreifen.

Die zweite Fragestellung nach der Struktur von Linked Data Webapplikationen kann hingegen leichter beantwortet werden. Abhängig von der gewählten Strategie zum Consuming der Daten des Linked Data Web wird für Linked Data Webapplikationen ein Datenintegrations-Layer (oder Komponente) benötigt. Diese Schicht oder Komponente muss Themen wie das Mapping von Ressourcen und Vokabularien als auch die Bewertung der Datenqualität zur Implementierung eines Ranking-Verfahrens berücksichtigen.
Werden Daten neu aggregiert, wird empfohlen, diese „neuen“ Daten wieder als Datasets im Linked Data Web zur Verfügung zu stellen.

Ausblick

In der Vergangenheit wurden sehr viele Anstrengungen unternommen, um Daten als Linked Data zu veröffentlichen. Aus den Erfahrungen sind sogenannte. „Best Practices“ entstanden, die als eine Art Fahrplan zum Publizieren von Linked Data zu verstehen sind. Dennoch gibt es hier sehr viele unterschiedliche Tools und Vokabularien. Die Zukunft wird zeigen, ob sich hier bestimmte Technologien und Vokabularien durchsetzen.

Derzeit werden sehr viele Anstrengungen unternommen, um die Daten noch „reicher“ zu machen, indem Verlinkungen zwischen Ressourcen vermehrt generiert werden. Hier ist zu erwarten, dass es künftig zentrale Services gibt, die Co-reference-mappings und andere Verlinkungen anbieten.

Die Reichhaltigkeit von Linked Data ermöglicht die Entwicklung neuer Webapplikationen. Derzeit wird noch viel Aufwand in die Bereitstellung und Verlinkung von Linked Data gesteckt. Künftig könnten diese Ressourcen genutzt werden, um Linked Data Applikationen zu entwickeln, die bessere Userinterfaces ermöglichen.

Zentrale Services (z.B. aggregierte SPARQL-Interfaces, semantische Indexes etc.) könnten eine performante Abfrage des Linked Data Web ermöglichen und

somit eine Grundlage zur Entwicklung von performanten und benutzerfreundlichen Linked Data Webapplikationen bilden.

Die meisten bibliografischen Daten werden derzeit im Web durch proprietäre oder freie integrierte Bibliotheksmanagementsysteme (ohne Linked Data) zugänglich gemacht. Zur Etablierung vom Library Linked Data müssen diese Systeme mit einbezogen werden. Ein derzeit häufiges Format zur Publikation bibliografischer Daten ist OAI-PMH[56]. OAI2LOD[53]ist ein Wrapper, der OAI-PMH-Daten als Linked Data bereitstellt. Das neue Protokoll OAI-ORE[57] definiert einen Standard ähnlich den Linked Data Prinzipien. Daher muss es langfristig betrachtet das Ziel sein, dass integrierte Bibliotheksmanagement-Systeme statt OAI-PHM OAI-ORE Formate zur Verfügung stellen.

[56] Open Archives Initiative Protocol for Metadata Harvesting: http://www.openarchives.org/pmh/[zuletzt angesehen 17.05.2011]

[57] Open Archives Initiative Object Reuse and Exchange: http://www.openarchives.org/ore/[zuletzt angesehen 17.05.2011]

8 Literaturverzeichnis und Quellen

[1] Pascal Hitzler, Markus Krötzsch, Sebastian Rudolph, and York Sure, *Semantic Web*, 1st ed. Berlin, Heidelberg, Deutschland: Springer, 2008.

[2] Tim Berners-Lee, James Hendler, and Ora Lassila, "The Semantic Web," *Scientific American*, Mai 2001.

[3] Pascal Hitzler, Markus Krötzsch, and Sebastian Rudolph, *Foundations of Semantic Web Technologies*. Boca Raton: Chapman & Hall / CRC textbooks in computing, 2010.

[4] The World Wide Web Consortium. (2007, März) http://www.w3c.org. [Online]. http://www.w3.org/2007/03/layerCake.png

[5] Google. (2009, Mai) Introducing Rich Snipplets. [Online]. http://googlewebmastercentral.blogspot.com/2009/05/introducing-rich-snippets.html

[6] Tom Heath and Christian Bizer, *Linked Data: Evolving the Web into a Global Data Space*, 1st ed.: Morgan & Claypool, 2011.

[7] Tim Berners-Lee. (2006) Linked Data - Design Issues. [Online]. http://www.w3c.org/DesignIssues/LinkedData.html

[8] Toby Segaran, Colin Evans, and Jamie Taylor, *Programming the Semantic Web*, 1st ed. Sebatopol, USA: O'Reilly, 2009.

[9] Leo Sauermann and Richard Cyganiak. (2008, Dezember) Cool URIs for the semantic web - W3C interest group note. [Online]. http://www.w3.org/TR/cooluris/

[10] Kendall Grant Clark, Lee Feigenbaum, and Elias Torres. (2008, Januar) SPARQL PRotocol for RDF. [Online]. http://www.w3.org/TR/2008/REC-rdf-sparql-protocol-20080115/

[11] Eric Prud'hommeaux and Andy Seaborne. (2008, Januar) SPARQL Query Language for RDF. [Online]. http://www.w3.org/TR/rdf-sparql-query/

[12] Olaf Görlitz and Steffen Staab, "Federated Data Management and Query Optimization for Linked Open Data," in *New Directions in Web Data Management*, Athena Vakali and Lakhmi Jain, Eds. Berlin, Heidelberg:

Springer-Verlag, 2011, ch. 5, pp. 109-137.

[13] Christian Bizer, Anja Jentzsch, and Richard Cyganiak. (2011, Februar) State of the LOD Cloud. [Online]. http://lod-cloud.net/state

[14] Richard Cyganiak and Anja Jentzsch. (2010, September) The Linked Open Data cloud diagram. [Online]. http://richard.cyganiak.de/2007/10/lod/lod-datasets_2010-09-22_colored.html

[15] Michael Hausenblas. (2009, Juli) Linked Data Applications. Dokument.

[16] Kevin R. Page, David C. De Roure, and Kirk Martinez, "REST and Linked Data: a match made fo domain driven development?," in *Second International Workshop on RESTful Design at WWW 2011*, Hyderabad (Indien), 2011.

[17] Erik Wilde and Michael Hausenblas, "RESTful SPARQL? You Name It!," in *4th Workshop on Emerging Web Services Technology WEWST 2009*, Eindhoven (Niederlande), 2009.

[18] Andreas Harth, Katja Hose, Marcel Karnstedt, Axel Polleres, and Kai-Uwe, Umbrich, Jürgen Sattler, "Data Summaries for On-Demand Queries over Linked Data," in *10 Proceedings of the 19th international conference on World wide web*, New York, 2010, pp. 411-420.

[19] Olaf Hartig and Jun Zhao, "Publishing and Consuming Provenance Metadata on the Web of Linked Data," in *Proc of 3rd Int Provenance and Annotation Workshop*, New York, 2010, p. 14.

[20] Olaf Hartig and Andreas Langenegger. (2010, Oktober) A Database Perspective on Consuming Linked Data on the Web. Dokument.

[21] Olaf Hartig, Christian Bizer, and Johann-Christoph Freytag. (2009) Executing SPARQL Queries over the Web of Linked Data. Dokument.

[22] Ian C. Millard, Hugh Glaser, Manuel Salvadores, and Nigel Shadbolt. (2010) Consuming multiple linked data sources: Challenges and Experiences. Dokument.

[23] Olaf Hartig, "Towards a Data-Centric Notion of Trust in the Semantic Web," in *2nd Workshop on Trust and Privacy on the Social and Semantic Web SPOT2010*, Heraklion (Greece), 2010.

[24] Olaf Hartig. (2010, Dezember) trdf - Trust-Komponente für Jena Framework. [Online]. http://sourceforge.net/apps/mediawiki/trdf/index.php?title=Quality_Criteria_for_Linked_Data_sources

[25] Marcus Cobden, Jennifer Black, Nicholas Gibbins, and Nigel Shadbolt. (2010, November) Consuming Linked Closed Data. Dokument.

[26] Robert Isele, Jürgen Umbrich, Christian Bier, and Andreas Harth, "LDSpider: An open-source crawling framework for the Web of Linked Data," in *9th International Semantic Web Conference (ISWC2010)*, Shanghai, November 2010.

[27] Jena Community. (2010) Jena Semantic Web Framework. [Online]. http://jena.sourceforge.net

[28] Chris Bizer, Tobias Gauß, Richard Cyganiak, and Olaf Hartig. (2009, November) Semantic Web Client Library. [Online]. http://www4.wiwiss.fu-berlin.de/bizer/ng4j/semwebclient/

[29] Joshua Shinavier, "Functional Programs as Linked Data," in *3rd Workshop on Scripting for the Semantic Web Innsbruck Austria*, Innsbruck, 2007, pp. 1-10.

[30] Peter Mika, "Distributed Indexing for Semantic Web," in *SEMSEARCH 10*, Raleigh, NC, USA, 2010, pp. 1-4.

[31] SWEO Community. (2010, Mai) SWEO Community Project: Linking Open Data on the Semantic Web - Applikationen. [Online]. http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData/Applications

[32] BenjaminM Hayes, Conor Heitmann and Eyal Oren. (2009, Februar) semwebapp-components. [Online]. http://semwebapp-components.dabbledb.com

[33] Christian Bizer, Tom Heath, and Tim Berners-Lee, "Linked Data - The Story So Far," *International Journal on Semantic Web and Information Systems*, vol. 5, no. 3, pp. 1-22, 2009.

[34] W3C Community. (2010, November) SWEO Community Project: Linking Open Data on the Semantic Web - Client Applications. [Online].

http://www.w3.org/wiki/TaskForces/CommunityProjects/LinkingOpenData/SemWebClients

[35] Benjamin Heitmann, Conor Hayes, and Eyal Oren, "Towards a reference architecture for Semantic Web applications," in *Proceedings of the 1st International Web Science Conference*, Athen, 2009, p. 5.

[36] Jörg Waitelonis, Magnus Knuth, Johannes Hercher, and Harald Sach, "The Path is the Destination - Enabling a New Search Paradigm with Linked Data," in *Linked Data in the Future Internet at the Future Internet Assembly*, Ghent, 2010, p. 8.

[37] Zhenning Shangguan and Debroah L. McGuiniss. (2009) Towards Faceted Browsing over Linked Data. Dokument.

[38] Max L. Wilson, Bill Kules, M.C. Schraefel, and Ben Shneiderman, "From Keyword Search to Exploration: Designing Future Search Interfaces for the Web," *Foundations and Trends in Web Sience*, vol. 2, no. 1, pp. 1-97, 2010.

[39] Michael Tvarožek and Mária Bieliková, "Personalized Exploratory Search in the Semantic Web," in *Proceedings of the 10th international conference on Web engineering*, Wien, 2010, pp. 527-530.

[40] Lynda Hardman, Jacco van Ossenbruggen, Raphael Troncy, Alia Amin, and Michiel Hildebrand, "Interactive Information Access," in *Proceedings of the WebSci'09*, Athen, 2009, p. 5.

[41] Giovanni Tummarello et al., "Sig.ma: Live Views on the Web of Data," in *International World Wide Web Conference Committee (IW3C2)*, Raleigh, 2010.

[42] Samur F. C. de Araúja and Daniel Schwabe, "Explorator : a tool for exploring RDF data through direct manipulation.," in *Proceedings of the WWW2009 Workshop on Linked Data on the Web*, Madrid, 2009, p. 11.

[43] Roberto Mirizzi and Tommaso Di Noia, "From Exploratory Search to Web Search and back," in *Proceedings of the 3rd workshop on PhD students in information and knowledge management*, Toronto, 2010, pp. 39-46.

[44] Hugh Glaser, Ian C. Millard, and Afraz Jaffri, "RKBExplorer.com: A Knowledge Driven Infrastructure for Linked Data Providers," in

Proceedings of the 5th European Semantic Web Conference ESWC 2008, Teneriffa, Spanien, 2008, pp. 797-801.

[45] The RKB Explorer Initiative. (2011, März) RKB Explorer. [Online]. http://www.rkbexplorer.com/data/

[46] Tom Baker, Emmanuelle Bermes, and Antoine Isaac. (2010, Juni) Library Linked Data Incubator Group. [Online]. http://www.w3.org/2005/Incubator/lld/charter

[47] Teilnehmer der LLD XG. (2011, März) Use Cases - Library Linked Data. [Online]. http://www.w3.org/2005/Incubator/lld/wiki/UseCases

[48] Jürgen Kett. (2010, November) culturegraph.org - Basisinfrastruktur für Gedächtnisinstitutionen im Semantic Web. [Online]. http://www.hbz-nrw.de/dokumentencenter/presse/pm/culturegraph_de

[49] Richard Altenhöhner. (2010, November) Kollaboration durch das Semantic Web: Strategie und Aktivitäten der Deutschen Nationalbibliothek. [Online]. http://swib.org/swib10/vortraege/swib10_altenhoener.ppt

[50] Pierre-Yves Vandenbussche. (2011, Mai) SPARQL Endpoints Status. [Online]. http://labs.mondeca.com/sparqlEndpointsStatus/index.html

[51] Adrian Pohl. (2011, Oktober) Linked Data und die Bibliothekswelt. Dokument.

[52] Jan Hannemann, "Linked Data for Libraries," in *WORLD LIBRARY AND INFORMATION CONGRESS: 76TH IFLA GENERAL CONFERENCE AND ASSEMBLY*, Gothenburg, Juni 2010, p. 12.

[53] Bernhard Haslhofer and Bernhard Schandl, "Interweaving OAI-PMH Data Sources with the Linked Data Cloud," *International Journal of Metadata Semantics and Ontologies*, vol. 5, no. 1, pp. 17-31, 2010.

Abbildungsverzeichnis

Abbildung 1: Protocol Stack des Semantic Web [4] 9
Abbildung 2: SPARQL Query 17
Abbildung 3: verteilte SPARQL-Query (Auszug aus [12], S. 127) 18
Abbildung 4: Linking Open Data Cloud, Stand: September 2010 [14] 20
Abbildung 5: Übersicht "Publishing Linked Data" [6], Abbildung 5.1 22
Abbildung 6: semantic sitemaps und voiD [15] 23
Abbildung 7: Prozess "Linked Data Consumption" [25], S. 8 39
Abbildung 8: Referenz-Architektur nach Heitmann et al. [35] 47
Abbildung 9: Architektur einer LD Webapplikation [15], S. 17 49
Abbildung 10: Architektur einer LD-driven Applikation (mit crawlinng pattern) 51
Abbildung 11: exploratory search von openlibrary.org 57
Abbildung 12: LED Screenshot 60
Abbildung 13: LED Architektur [43] 61
Abbildung 14: Sig.ma Mashup 64
Abbildung 15: Workflow Recherche in sig.ma [41], S. 2 65
Abbildung 16: RKBExplorer GUI 69
Abbildung 17: culturegraph als co-reference-Dienst [49] 75
Abbildung 18: Klassifikation der Datasets LLD 78
Abbildung 19: Klassifikation der Triples LLD 78
Abbildung 20: Verteilung der Links im Bereich LLD nach Ziel-Datasets 79
Abbildung 21: Ausgehende Links/Dataset für LLD-Datasets 80
Abbildung 22: Module von LLD Search 82
Abbildung 23: Schnittstellen LLD Search zur LD-Web 85
Abbildung 24: Software-Architektur LLD Search (Heitmann Modell) 86
Abbildung 25: LLD Search - Backend-Komponenten 88
Abbildung 26: Screenshot LLD Seach - Startseite 92
Abbildung 27: LLD Search Kurzanzeige 93
Abbildung 28: LLD Search Vollanzeige 93
Abbildung 29: LLD Search Kurzanzeige - Suche in PND 94
Abbildung 30: LLD Search Vollanzeige "Angela Merkel" 95
Abbildung 31: LLD Search Kurzanzeige "Informatik" 96
Abbildung 32: LLD Vollanzeige "Informatik" 96
Abbildung 33: LLD Search Expertensuche 97
Abbildung 34: LLD Search Kurzanzeige einer SPARQL-Query 97
Abbildung 35: LLD Search Vollanzeige (LD Browser) 98

Tabellenverzeichnis

Tabelle 1: LOD Cloud Datasets 19

Abkürzungsverzeichnis

CKAN	Comprehensive Knowledge Archive Network
DOI	Digital Object Identifier
DQP	Distributed query processing approaches
FYN	Follow-your-nose: Prinzip des Browsens von Links im Web
HTTP	Hypertext Transfer Protocol
IQ	Information Quality
JSON	JavaScript Object Notation
LCD	Linked Closed Data
LD	Linked Data
LLD	Library Linked Data
LLD XG	W3C Library Linked Data Incubator Group
LOD	Linked Open Data
MAT	Materialisation-based approaches
OWL	Web Ontology Language
RDF	Resource Description Framework
RDFS	RDF Schema
RIF	Rule Interchange Format
SKOS	Simple Knowledge Organisation System
SW	Semantic Web
URI	Uniform Resource Identifier
URN	Uniform Resource Name
voiD	Vocabulary of Interlinked Datasets
W3C	World Wide Web Consortium

Anhang A: LLD Datasets

Zur Klassifikation der Datasets

Vertreter der Gruppe Titeldaten, Publikationen und Bestandsnachweise umfasst folgende Datasets: Bibbase, Openlibrary, LD Dienst hbz (lobid.org), Mannheim LD Dienst, Ungarische Nationalbibliothek (NSZL), Library of Congress LCCN und English Language Books listed in Printed Book Auction Catalogues from 17th Century Holland (PBAC).

Vertreter der Gruppe Normdaten und Thesauri sind VIAF, GND (der deutschen Nationalbibliothek), Ungarische Nationalbibliothek (NSZL), Library of Congress Subject Headings, Rameau Subjects Headings, National Diet Library of Japan Subject Headings, Thesaurus for Social Sciences (GESIS), Thesaurus for Economics (STW), Thesaurus for Graphic Materials (T4GM), MARC Codes Lists und Polythematic Structured Subject Heading System.

Vertreter der Gruppe Digitale Objekte sind Openlibrary, Cronicle America und 20th Century Press Archive (P20).

Vertreter der Gruppe Institutionen ist der Dienst des hbz (lobid.org organizations).

Anmerkung: Einige Datasets sind in mehreren Gruppen enthalten, wenn sie nicht eindeutig zuzuordnen waren.

Untersuchung der Datasets (Stand 11.03.2011)

#	Dataset	Klasse	Lizenz	voiD	SPARQL	Triples
1	Openlibrary	T, D	OKD Compliant::Other	nein	nein	400000000
2	VIAF	N	-	nein	nein	200000000
3	lob.id-organizations	I	CC-Zero	nein	nein	105000000
4	Mannheim	T	-	nein	ja	53415605
5	GND	N	Non-OKD Compliant::Other	nein	nein	40182561
6	Cronicle America	D	OKD Compliant::Other	nein	nein	20000000
7	NSZL Catalog	T, N	-	nein	nein	17500000

8	P20	D	Special	Ja	nein	12000000
9	LCSH	N	OKD Compliant::Other	Nein	nein	4151586
10	Rameau	N	-	nein	nein	1619918
11	Linked LCCN	T	OKD Compliant::Other	nein	ja	1573663
12	NDL subjects	N	-	nein	ja	1294669
13	Lob.id-organizations	I	-	ja	nein	587936
14	GESIS	N	by-nc-nd	ja	ja	181558
15	STW	N	by-nc	ja	nein	107000
16	T4GM	N	cc-by-sa	nein	nein	103000
17	Marc codes list	N	OKD Compliant::Other	nein	nein	8816
18	polythematic structured subject heading system	N	by-nc	nein	nein	100000
19	PBAC	T	odc-pddl	ja	Ja	55214
20	BibBase	T	cc-by	nein	ja	200000
Summe:						858081526

Klassifikation:

T: Titeldaten, Bestandsnachweise und Publikationen
D: Digitale Objekte
N: Normdaten, Thesauri
I: Institutionen

Untersuchung der Verlinkung (Stand 11.03.2011)

Ausgehende Links zu:

A: dbpedia, B: STW, C: LCSH, D: VIAF,
E: marccodes, F: book-isbn, G: MusicBrains H: LinkedMDB,
I: openlibrary, J: geonames, K: GND, L: lob.id-organizations,
M: bookmashup, N: DBLP, O: IEEE, P: ACM,
Q: Citeseer

	A	B	C	D	E	F	G	H	I
1									
2	10000								
3									
4									
5	40136		37547	1786975					
6	10000								
7	4852								
8	2500								
9									
10			60000						
11	10911			209422	80275	4468	6909	883	15331
12			588						
13									
14		5024							
15	3200								
16			19300						
17	599		2232						
18	3000							3000	
19	1607								
20	53								
Summe	86858	5024	119667	1996397	80275	4468	6909	3883	15331

	J	K	L	M	N	O	P	Q	
1									0
2									10000
3		5300000	6600000						11900000
4		3839380		1429424					5268804
5									1864658
6									10000
7									4852
8		3800							6300
9									0
10		20000							80000
11									328199
12									588

13									0
14									5024
15									3200
16									19300
17	534								3365
18									6000
19									1607
20					5000	75	21513	7872	34513
Summe	534	9163180	6600000	1429424	5000	75	21513	7872	

Anhang B: Ausschnitte aus Sourcecode

Hier werden relevante Teile des Sourcecodes des Prototypen zur Indexierung, Umgang mit dem Jena-Framework und ausgewählte Aspekte der Webanwendung gezeigt.

RDF-Daten mit Jena einlesen:

Download Jena-Framework: http://jena.sourceforge.net/downloads.html

```
//rdf-Model erzeugen
Model rdfModel = ModelFactory.createDefaultModel();

//wenn keine komplette XML-Datei bei RDF/XML – z.B. für GND-Dump benötigt:
//rdfModel.getReader().setProperty("embedding", "true");

//Datei öffnen und in Model einlesen
InputStream in = FileManager.get().open(file.toString());
rdfModel.read(in, baseURI, ”RDF/XML”);

//alle Triples holen
StmtIterator itModel = rdfModel.listStatements();

//über Triples iterieren
while (itModel.hasNext()) {
Statement st = itModel.next();
String subject = st.getSubject().toString();
String predicate = st.getPredicate().toString();
String object = st.getObject().toString();
```

// etwas damit tun
}

Indexierung:

```
//IndexWriter öffnen
IndexWriter writer = newIndexWriter(
        FSDirectory.open(INDEX_DIR),
        new StandardAnalyzer(Version.LUCENE_31),
        true,
        MaxFieldLength.UNLIMITED);

//neues Lucene Dokument erzeugen
Document doc = new Document();

//neues Feld dem Dokument hinzufügen
doc.add(new Field("URI","http://lobid.org/resource/BT00000626",
Field.Store.YES,Field.Index.NOT_ANALYZED));

//Dokument dem Index hinzufügen
writer.addDocument(doc);

//Index optimieren
writer.optimize();

//Index schließen
writer.close();
```

Mapping

Jedes Prädikat in einer RDF-Beschreibung, die von LLD Search indexiert oder angezeigt werden soll wird in einer Mapping-Konfiguration gespeichert. Für jedes Prädikat werden somit die Einstellungen name, label, prefix und occurence gespeichert. Weiter werden die Attribute show, sort und browsable verwendet.
Diese Konfiguration wird von der Indexierungskomponente und der Backend-Komponente verwendet, um die Daten zu lesen, indexieren und aufbereiten.

Name	**Beschreibung**
Tag „name“	URI des Prädikats
Tag „prefix“	Name des Feldes im Lucene-Index

Tag „label“	Name des Feldes, wie er dem Benutzer angezeigt werden soll
Tag „occurence“	Angabe, ob das Feld in einem Dokument mehrfach vorkommen darf
Attribute „show“	Angabe, ob dieses Feld als Facette verwendet werden soll
Attribute „sort“	Verwendet zur Sortierung der Felder in der Anzeige im Frontend
Attribute „browseable“	Angabe, ob dieses Feld auch zum Browsen erlaubt werden soll (d.h. bei der Suche in diesem Feld muss kein Suchwert eingegeben werden)

```
<?xml version="1.0" encoding="UTF-8"?>
<facets>
	<facet show="false" sort="1" browseable="false">
		<name>URI</name>
		<label>identifier</label>
		<prefix>URI</prefix>
		<occurrence>single</occurrence>
	</facet>
	<facet show="true" sort="1" searchShow="true" browseable="true">
		<name>http://purl.org/dc/terms/identifier</name>
		<label>dcterms:identifier</label>
		<prefix>dcterms:identifier</prefix>
		<occurrence>single</occurrence>
	</facet>
	<facet show="false" sort="2" browseable="false">
		<name>http://purl.org/ontology/bibo/isbn</name>
		<label>isbn</label>
		<prefix>dcterms:isbn</prefix>
		<occurrence>multiple</occurrence>
	</facet>
	<facet show="false" sort="2" browseable="false">
		<name>http://purl.org/dc/terms/title</name>
		<label>title</label>
		<prefix>dcterms:title</prefix>
		<occurrence>single</occurrence>
	</facet>
…
</facets>
```