BoardToOntology – Ontology Learning auf Basis der Wissensquelle Online-Diskussionsplattform

Andreas Volk

MASTERARBEIT

eingereicht am
Fachhochschul-Masterstudiengang

Interactive Media

in Hagenberg

im Juli 2012

Erklärung

Ich erkläre eidesstattlich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benutzt und die den benutzten Quellen entnommenen Stellen als solche gekennzeichnet habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Hagenberg, am 13. Juli 2012

Andreas Volk

Inhaltsverzeichnis

Kurzfassung

Im Zuge der Weiterentwicklung zum Semantic Web werden neue Datenstrukturen benötigt, die es ermöglichen, semantische Relationen abzubilden. Dies kann mit Ontologien erreicht werden. Die Problematik dabei ist, dass es sich beim Erstellen einer Ontologie um einen zeitaufwändigen, komplexen und fehleranfälligen Prozess handelt. Auf Grund dessen wurden Methoden entwickelt, die Ontologien automatisch anhand verschiedener Wissensquellen erstellen. In Rahmen dieser Arbeit wurde eine Methodik konzipiert, um Daten exemplarisch anhand des deutschsprachigen Board-Systems Android-Hilfe.de zu sammeln und diese anschließend zu einer Ontologie zu verarbeiten. Weiter wurde eine Text-Analyse auf Basis der geparsten Daten vorgenommen, um Terme zu extrahieren. Diese Wörter wurden mittels der TF-IDF Formel nach Relevanz sortiert und in ein XML-Dokument gespeichert. Die Termkandidaten können von einem Experten zu einer Taxonomie weiterverarbeitet werden. Mit Hilfe der gebildeten Taxonomie und der entstandenen Ontologie wäre es beispielsweise möglich, eine semantische Suche für das untersuchte Board-System zu implementieren.

Abstract

In the course of further development on the Semantic Web new data structures are required, which allow to represent semantic relations. This can be achieved with ontologies. The Problem here is that creating an ontology is a time consuming, complex and error susceptible task. Because of that, methods were developed to create ontologies automatically based on various sources of knowledge. In this thesis a methodology was developed to collect data exemplified by the German language board system Android-Hilfe.de and then process it to an ontology. Furthermore a text analysis based on the parsed data was performed to extract terms. These words have been sorted by relevance using the TF-IDF formula and stored in an XML. The term candidates can be further processed by an expert in a taxonomy. With the help of the resulting taxonomy and the resulting ontology, it would be possible to implement a semantic search for the considered board.

Kapitel 1

Einleitung

1.1 Problemstellung

> Ontologies play an important role in providing a controlled vocabulary of concepts, each with an explicitly defined and machine processable semantics. They are largely used in the next generation of the Semantic Web which focuses on supporting a better cooperation between humans and machines. [14]

Im Zuge der Entwicklung des Semantic Webs wurde es notwendig Datenbestände mit semantischen Informationen aufzubereiten. Da sich relationale Datenbanken nur begrenzt dazu eignen, wurde auf Ontologien als Wissensbasis zurückgegriffen. Eine Ontologie ermöglicht es, wie auch im einleitenden Zitat deutlich wird, maschinenlesbare semantische Inhalte zu erzeugen und damit dem Computer logische Schlussfolgerungen zu ermöglichen. Die Erstellung von Ontologien ist jedoch ein sehr zeitaufwendiger, teurer und fehleranfälliger Prozess [14]. Neben den angesprochenen Problemen gestaltet es sich sehr schwer, einen Experten für eine bestimmte Domäne zu finden. Gleichzeitig muss die fertiggestellte Ontologie immer auf dem aktuellsten Stand gehalten werden. Auf Grund dieser Komplexität des Entwicklungsprozesses einer Ontologie sind Ansätze entwickelt worden, um dieses Vorhaben zu automatisieren. Diese arbeiten mit verschiedenen Eingabeformaten als Wissensquelle, wie beispielsweise Datenbanken, HTML - oder XML-Dokumenten, oder auch natürlich sprachlichen Texten. Im Rahmen dieser Arbeit wurde das Board-System Android-Hilfe.de als Wissensquelle gewählt. Die Informationen, die in einem Forum - bzw. Board stecken, könnten durch Techniken des Semantic Webs besser zugänglich gemacht werden. Dafür wird jedoch wieder eine Ontologie mit den Board-Daten benötigt. Auf Grund dessen ergibt sich die Problemstellung für diese Arbeit mit der automatischen Generierung einer Ontologie auf Basis der Daten des Forschungsgegenstands.

1.2 Motivation

Die Motivation für die im Zuge dieser Arbeit durchgeführte Forschung gründet sich auf zwei zentralen Punkten. Der Erste bezieht sich auf das hohe Potential von Foren- bzw. Board-Systemen als Wissensquelle zur Erstellung einer Ontologie. Internetforen dienen hauptsächlich als Diskussionsplattform für spezifische Domänen. Durch diese Diskussionen findet jedoch auch ein Wissensaustausch statt und auf Grund dessen können Foren auch als Informationsquelle genutzt werden. Die Informationen zu spezifischen Problemen zu finden, gestaltet sich mitunter als sehr schwierig. Es werden zwar Suchoptionen bereitgestellt, jedoch kann man beispielsweise nicht nach bestimmten Kategorien suchen. Das Suchverhalten könnte durch Techniken des Semantic Webs verbessert werden und mittels Ontology Learning könnte der Aufwand für die semantische Auszeichnung der Daten vermindert werden. Dies ist der zweite zentrale Punkt für die Erstellung dieser Arbeit, denn Foren könnten sich durch ihre hierarchische Strukturierung sehr gut für die automatische Erstellung einer Ontologie bzw. einer Taxonomie eignen. Außerdem ist es denkbar, dass wegen der vielen Themen und Einträgen zu einer bestimmten Domäne genug Wissen in den Foren vorhanden ist, um ein Themengebiet nahezu komplett zu erfassen. Ein positives Ergebnis im Zuge der folgenden Untersuchungen könnte somit eine Verbesserung der Wissensgewinnung aus Foren zur Folge haben, ohne dabei viel Aufwand zu betreiben, woraus sich letztendlich die Motivation für diese Arbeit ergibt.

1.3 Zielsetzung

Im Zuge dieser Arbeit wird einerseits das Feld Ontology Learning theoretisch abgehandelt und andererseits eine Methodik beschrieben, die dazu dienen soll eine Ontologie automatisch anhand der Daten eines Board-Systems zu erstellen. Außerdem soll erläutert werden, wie auf Basis einer textuellen Analyse mögliche Terme für den Aufbau einer Taxonomie extrahiert werden können. Daraus ergeben sich zwei Ziele für die in dieser Arbeit durchgeführte Forschung. Einerseits soll ein System entwickelt werden, mit dem es möglich ist, die Daten des Forschungsgegenstands Android-Hilfe.de zu erfassen und zu einer Ontologie zu verarbeiten. Weiter soll dieses System auf Basis der Themen und Beiträge eine textuelle Analyse vornehmen, mit der eine Rangliste erstellt wird, in der mögliche Terme für die Erstellung einer Taxonomie in der Domäne Android App-Entwicklung dargestellt werden.

1.4 Aufbau der Arbeit

Nach der Einleitung folgt ein Kapitel über die in dieser Arbeit benötigten Grundlagen. Dabei werden das Konzept einer Ontologie, sowie das For-

schungsfeld Ontology Learning definiert und die Grundlagen der Beschreibungssprache OWL erläutert. Daraufhin folgt in Kapitel 3 ein Überblick über die bereits geleistete Arbeit im Forschungsbereich Ontology Learning mit einer Beschreibung der daraus resultierenden Ansätze und Frameworks. Auf Basis dieser Erkenntnisse wird eine detaillierte Erläuterung der entwickelten Methodik vorgenommen. Diese beinhaltet eine Spezifizierung des Forschungsgegenstands als Wissensquelle für Ontology Learning. Weiter wird in diesem Kapitel der Unterschied zwischen einem Foren- und Board-System, sowie deren Aufbau thematisiert. Außerdem folgt die Beschreibung der Systemarchitektur der entwickelten Software, wobei der Aufbau der drei zentralen Module Datenerfassung, Datenhaltung und Datenverarbeitung in den Fokus der Ausführungen gestellt werden. In Kapitel 5 wird auf die Implementierung des Systems eingegangen. Dabei wird zunächst auf den Internet-Crawler, die Aufbereitung der Rohdaten und das Datenmodell eingegangen. Daraufhin folgt eine Beschreibung des Ontology Mapping Prozesses und der textuellen Analyse mittels Methoden aus dem Natural Language Processing. Im darauf folgenden Kapitel sechs wird eine Evaluierung des Systems vorgenommen, in dem die Ergebnisse dargestellt werden. Den Abschluss der Arbeit bildet eine Conclusio, in der eine Zusammenfassung, ein Ausblick und ein Fazit über die entwickelte Methodik gegeben werden.

Kapitel 2

Grundlagen Ontology Learning

2.1 Ontologien

> People can't share knowledge if they don't speak a common language. [25]

Der Wissensaustausch zwischen Kommunikationspartnern kann nur erfolgen, wenn sich diese im selben sprachlichen Diskurs bewegen und die gleiche Vorstellung über die Entitäten in einer Welt besitzen [31]. Dieser Zusammenhang kann anhand des semiotischen Dreiecks (siehe Abb. 2.1), entwickelt von Charles K. Odgen und Ivor A. Richards, nachvollzogen werden. Dieses Modell beschreibt, dass ein Sprecher einen bestimmten sprachlichen Ausdruck immer mit einer bestimmten Vorstellung des Begriffs verknüpft und diese Vorstellung wiederum an ein konkretes Abbild (Referent) in der realen Welt gekoppelt ist. Die theoretischen Grundlagen des semiotischen Dreiecks können anhand folgenden Beispiels nachvollzogen werden:

> Wenn ich also frage[:] Papa, kann ich heute Abend dein Auto haben?, so verwende ich die Lautfolge [auto] für einen Gegenstand, von dem ich eine bestimmte Idee habe: Ein Auto ist eine Fahrzeug, in das man einsteigen kann und das normalerweise mit Benzin oder Diesel fährt etc. Außerdem aber verweise ich in meiner Frage auf ein ganz konkretes Auto, das in der außersprachlichen Wirklichkeit existiert (=Referent, nämlich das Auto in unserer Garage, z. B. ein blauer VW-Passat . . .). [12]

Die Konzepte einer Welt und die Relationen untereinander können innerhalb einer Ontologie beschrieben werden. Mit Hilfe dieser Ontologien kann die Kommunikation zwischen Computern, Menschen und Computern, sowie zwischen Menschen verbessert werden, da dadurch eine gemeinsame Wissens- bzw. Vorstellungsbasis geschaffen wird [23]. Die ersten Ansätze zur Beschreibung der Welt in Form einer Ontologie lassen sich auf die antike

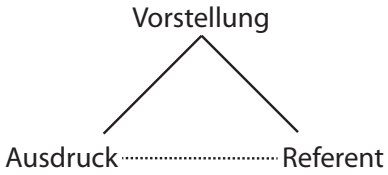

Abbildung 2.1: Semiotisches Dreieck [12].

griechische Philosophie zurückführen. Konkret stellte Aristoteles, auf Basis der Erkenntnisse seines Lehrers Platon, ein System zur Klassifizierung aller beschreibbaren Dinge auf. Im Zuge dieser Untersuchungen hinsichtlich des Gebiets *Metaphysik* definierte Aristoteles Ontologie als die Wissenschaft des Seienden als Seiendes [23]. Im philosophischen Kontext kann die Ontologie als die Lehre des Seienden bezeichnet werden. Ontologien werden jedoch nicht nur im Rahmen der Philosophie verwendet, sondern auch in der Informationstechnik. Dau, Mugnier und Stumme haben die unterschiedlichen Anwendungen in [18] folgendermaßen beschrieben:

> Where philosophical ontology has been concerned with the furniture and entities of reality, i.e., with the study of „being qua being“, computer scientists have been occupied with the development of formalized, semantic, and logic-based models, which can easily be implemented in computer systems.

Grundsätzlich handelt es sich um formalisierte Modelle, die es dem Computer ermöglichen, durch logische Verknüpfungen, Schlussfolgerungen zu ziehen. Dabei lassen sich fächerübergreifende, die allgemeine Beziehungen beschreiben, und domänenspezifische Ontologien, die ein bestimmtes Fachgebiet wie beispielsweise Physik oder Chemie beinhalten, unterscheiden [31]. In Abb. 2.2 ist exemplarisch eine domänenspezifische Ontologie aus dem Bereich Politik illustriert, anhand welcher man die grundlegenden Merkmale einer formalen Ontologie erkennen kann. Die grau hinterlegten Rechtecke (Lebewesen, Politiker) repräsentieren Konzepte, welche abstrakte Objekte der Welt darstellen und zusätzlich eine hierarchische Kategorisierung zulassen. Die Verbindungen zwischen den Ovalen bzw. Rechtecken illustrieren Beziehungen zwischen den Konzepten. Die ovalen Elemente stellen Instanzen dar, die das konkrete Auftreten der Konzepte verkörpern [31]. Eine detaillierte Beschreibung des formalen Aufbaus einer Ontologie folgt unter den Ausführungen zu der Beschreibungssprache OWL 2.3. Abschließend soll versucht werden, eine für diese Arbeit gültige Definition des Begriffs Ontologie zu finden. Studer et al. hat 1998, basierend auf den Begriffserklärungen von

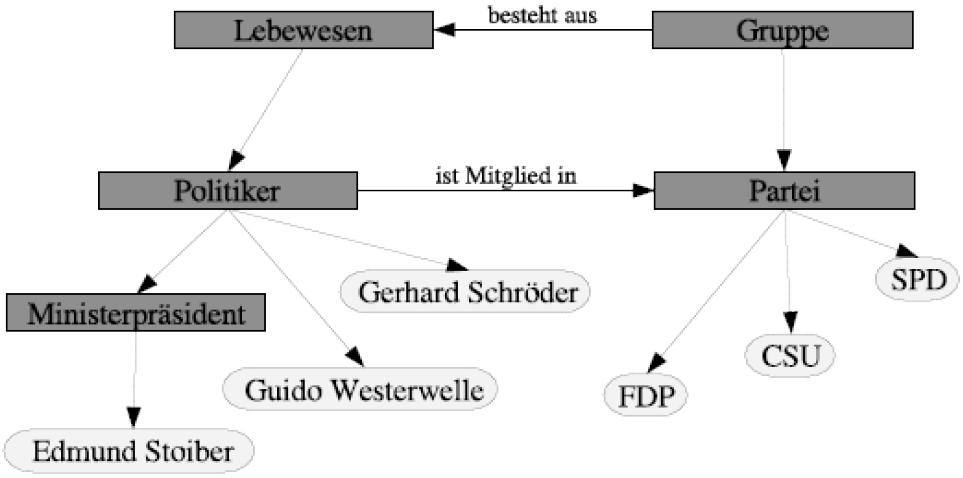

Abbildung 2.2: Beispiel Ontologie [31].

Gruber und Borst, folgende Definition, entnommen aus [23], aufgestellt:

> An ontology is a formal, explicit specification of a shared conceptualization.

Zum besseren Verständnis der oben genannten Definition müssen die Begriffe „formal, explicit specification“ und „conceptualization“ näher erläutert werden. Auf Grund des beschränkten Umfang dieser Arbeit können diese Termini nur kurz erklärt werden. Für eine detaillierte Ausführung soll an dieser Stelle auf das Buch Handbook of Ontologies [23] verwiesen werden. Durch die formale und explizite Spezifikation soll garantiert werden, dass auch Personen neben dem Entwickler die Ontologie korrekt interpretieren und erweitern können. Die Konzeptualisierung kann wie folgt definiert werden:

> A body of formally represented knowledge is based on a conceptualization: the objects, concepts, and other entities that are assumed to exist in some area of interest and the relationships that hold among them. A conceptualization is an abstract, simplified view of the world that we wish to represent for some purpose. Every knowledge base, knowledge-based system, or knowledge-level agent is committed to some conceptualization, explicitly or implicitly. [23]

Genesereth und Nillson legen die Konzeptualisierung als abstrakte, vereinfachte Darstellung der Welt dar, welche alle Objekte, Konzepte, Entitäten und deren Beziehungen untereinander beinhaltet. Dadurch ergibt sich, dass

im informationstechnischen Kontext Ontologien als hierarchisch strukturierte vereinfachte Darstellung eines Weltauschnitts zu definieren sind.

Durch die vorangegangenen Ausführungen wurde deutlich, was der Begriff Ontologie im philosophischen, sowie im informationstechnischen Kontext bedeutet. Andererseits ist deutlich geworden, dass die Erstellung einer Ontologie ein sehr komplexer und unter Umständen langwieriger Prozess ist. Hinsichtlich der Weiterentwicklungen im Bereich Semantic Web wächst der Bedarf an Ontologien stetig. Auf Grund dessen ist es erforderlich die Erstellung von Ontologien zu vereinfachen, wobei die Methoden von Ontology Learning eingesetzt werden könnten. Im Folgenden soll der Begriff Ontology Learning definiert und erläutert werden.

2.2 Ontology Learning

> Though ontology engineering tools have become mature over the last decade (cf. [2]), the manual acquisition of ontologies still remains a tedious, cumbersome task resulting easily in a knowledge acquisition bottleneck. [16]

Die Erfassung und Aufbereitung von Wissen in Form einer Ontologie ist trotz der dafür entwickelten Programme, wie beispielsweise Protègè, mit großem Aufwand verbunden. Bei der manuellen Erstellung dieser Wissensbasen werden sogenannte Wissensingenieure benötigt, die meist von einem Fachexperten unterstützt werden. Diese Experten sind jedoch schwer zu finden und auf Grund der schnellen Weiterentwicklungen in den jeweiligen Gebieten kann es auch dazu kommen, dass das Expertenwissen subjektiv, unvollständig oder nicht mehr aktuell ist. Auf Grund dessen werden andere Medien, wie Texte, Bücher, das World Wide Web oder vorhandene Datenbanken als Wissensquellen für Ontologien verwendet. Dabei kann der Bereich Ontology Learning den Aufwand bei der Erfassung und Aufbereitung dieser Daten reduzieren [30]. Ontology Learning basiert auf data-mining Techniken, die durch maschinelles Lernen aus dem Feld Künstliche Intelligenz unterstützt werden und kann wie folgt definiert werden:

> Ontology learning (OL) is an emerging field aimed at assisting a knowledge engineer in ontology construction and semantic page annotation with the help of machine learning (ML) techniques. [19]

Im Zuge der Verbesserung des Entwicklungsprozesses von Ontologien sind zahlreiche Frameworks entstanden, wie TextToOnto und OntoLT, die im nachfolgenden Kapitel näher erläutert werden sollen. Bei den Wissensquellen bzw. Eingabedaten für das Ontology Learning lassen sich drei verschiedene Arten unterscheiden:

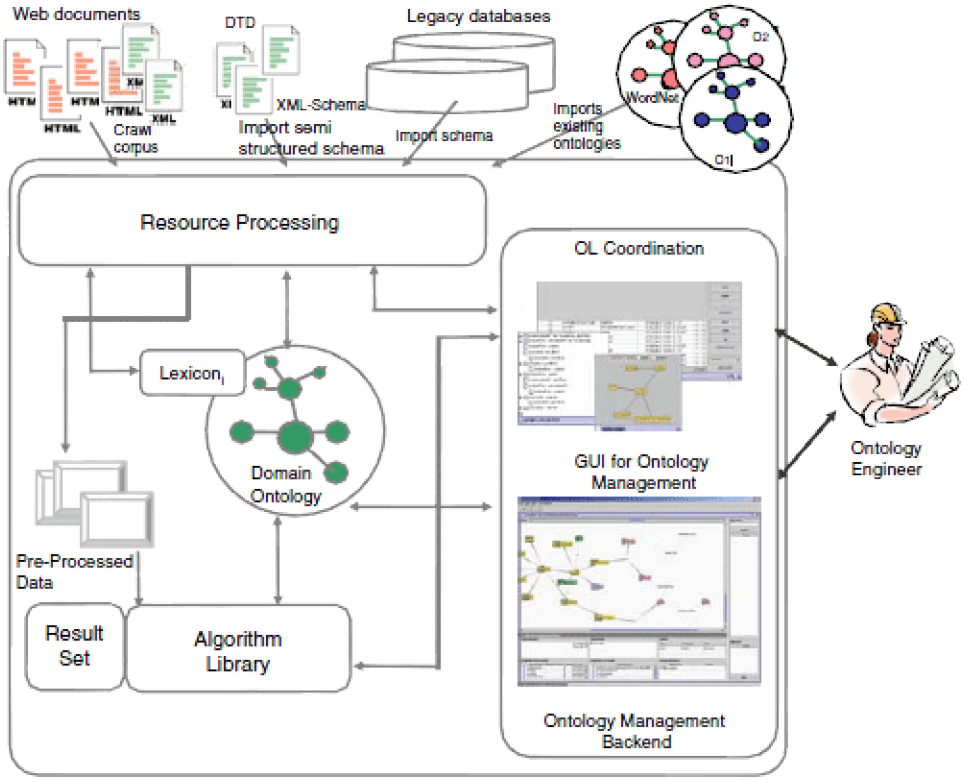

Abbildung 2.3: Architektur des generischen Ontology Learning Framework [23].

- strukturiert (Datenbanken)
- semi-strukturiert (HTML oder XML)
- unstrukturiert (bspw. Textdokumente)

Die verschiedenen Methoden für Ontology Learning lassen sich anhand dieser Unterschiede in der Strukturierung unterteilen. Während für die strukturierten Daten eher eine Art Mapping der Daten mit vermindertem Einsatz von maschinellem Lernen vorgenommen werden kann, müssen die semi- und unstrukturierten Daten zumeist unter Einsatz von Natural Language Processing aufbereitet werden [23]. Detaillierte Ausführungen zu den Methoden für die jeweilige Strukturierungsebene folgen im nächsten Kapitel. Bevor jedoch diese Systeme erläutert werden sollen, folgt an dieser Stelle die Beschreibung einer generischen Architektur für ein Ontology Learning System nach Staab et al. Die zentralen Komponenten, wie in Abb. 2.3 ersichtlich für ein Ontology Learning Framework sind Verwaltung, Koordination, Ressourcenverarbeitung und eine Library mit verschiedenen Learning Algorithmen. Die Verwaltungskomponente definiert eine Schnittstelle zwischen der Ontologie und den Algorithmen. Dabei enthält dieser Systemteil einerseits Methoden,

um die Ontologie mit neuem Wissen (Klassen, Relationen, Individuals etc.) anzureichern bzw. bei Änderungen zu aktualisieren. Andererseits muss diese Komponente Funktionen enthalten, die die Konsistenz der jeweiligen Ontologie gewährleistet.

Tabelle 2.1: Lernende Algorithmen und deren Verwendung im Ontology Learning [23].

Algorithm	*Generic use*	*Use in ontology learning*
Association rule discovery	Discovery of "interesting" transactions in itemsets (e.g., customer data)	Discovery of interesting associations between words
(Hierarchical) Clustering	Discovery of groups in data (unsupervised)	Clustering of words
Classification (e.g., SVMs, Naive Bayes, kNN, etc.)	Prediction (supervised)	Classification of new concepts into an existing hierarchy
Inductive logic programming	Induction of rules from data (supervised)	Classification of new concepts from extensional data
Conceptual clustering (e.g., FCA)	Concept discovery (extension and intension)	Learning concepts and concept hierarchies

Mithilfe der Koordinationskomponente kann der Ontology Engineer mit der Ressourcenverarbeitung und den Learning Algorithmen interagieren. Die Ressourcenverarbeitung enthält Techniken für das Einpflegen, Analysieren und Aufbereiten der jeweiligen Wissensquellen, wobei einer der wichtigsten Bestandteile das Modul für das Natural Language Processing darstellt. Zentrale Aufgabe ist dabei die Daten für die weitere Verarbeitung mit einem der Learning Algorithmen aufzubereiten. Die Bibliothek mit den Learning Algorithmen enthält nach Staab et al. die in Tab. 2.1 ersichtlichen Algorithmen [23]. Auf eine detailliertere Beschreibung des hier vorgestellten Frameworks muss auf Grund des begrenzten Umfangs dieser Arbeit verzichtet werden und anstatt dessen wird auf [23] verwiesen.

2.3 OWL – Grundlagen

> OWL is designed for use by applications that need to process the content of information instead of just presenting information to humans. [13]

Traditionelle (Web-)Applikationen dienen dazu, dem Benutzer Wissen in aufbereiteter Form zu präsentieren. Mit größerer Verbreitung des Semantic Webs und in Bereichen wie der Künstlichen Intelligenz ist es notwendig, dass

die Informationsinhalte nicht nur dargestellt werden, sondern verarbeitet bzw. interpretiert werden, um beispielsweise Schlussfolgerungen zu ziehen. Um Informationen für die maschinelle Interpretation aufzubereiten wird die Beschreibungssprache OWL eingesetzt. Diese bietet, wie in nachfolgendem Zitat deutlich wird, mittels zusätzlicher Sprachkonstrukten größere Möglichkeiten als in XML, RDF oder RDF-Schema, Informationen zu verarbeiten. Auf Grund dessen wird diese Beschreibungssprache auch im Feld Ontology Learning häufig angewandt. Deswegen soll im Folgenden ein Überblick über die wichtigsten Konzepte der Web Ontology Language gegeben werden.

> OWL facilitates greater machine interpretability of web content than that supported by XML, RDF, and RDF Schema (RDFS) by providing additional vocabulary along with a formal semantics. [13]

Dokumente, die mittels OWL erstellt wurden, lassen sich in den Kopfbereich und den Bereich mit der eigentlichen Ontologie unterteilen. Der Kopfbereich setzt sich aus XML-Namespace Deklarationen, die vor allem zu einer einfacheren Lesbarkeit der Ontologie führen, und dem Ontologie-Header zusammen. In diesem werden allgemeine Informationen, wie Kommentare über den Inhalt, die Versionierung, die Importierung anderer Ontologien etc., dargestellt. Danach folgt die Beschreibung der eigentlichen Ontologie, wofür verschiedene Sprachkonstrukte zur Verfügung stehen, die im Folgenden erläutert werden sollen. Die zentralen Elemente sind dabei Klassen, Individuen und Rollen.[1]

> A class defines a group of individuals that belong together because they share some properties. [32]

Klassen sind eine Gruppierung von Individuen, die gleiche Eigenschaften vorweisen und werden in einer hierarchischen Struktur abgebildet. Zum Aufbau solcher Taxonomien ist das Sprachkonstrukt *rdfs:subClassOf* vorgesehen, das angibt, dass eine bestimmte Klasse einer allgemeineren Klasse untergeordnet ist. Daraus ergibt sich, dass, wenn X eine Subklasse von Y ist, jede Instanz von X auch eine Instanz von Y sein muss. Weiter ist diese Beziehung transitiv, wodurch festgelegt ist, dass wenn X eine Subklasse von Y ist und Y eine Subklasse von Z, ist X auch eine Subklasse von Z [33]. Weitere Klassenbeziehungen sind *owl:disjointWith*, welche zwei Klassen als disjunkt deklariert, was bedeutet, dass kein Individuum der beiden Klassen gleich ist und *owl:equivalentClass*, wodurch festgelegt wird, dass die beiden Klassen gleich sind [10]. In OWL existiert die Basisklasse *Thing*, der jede andere Klasse untergeordnet ist und eine Klasse *Nothing*, der jeder Klasse übergeordnet ist. Individuen (Individuals) beschreiben dabei die konkreten

[1] Die Begriffe Rollen und OWL-Propertys werden im Zuge dieser Arbeit synonym verwendet.

Mitglieder der jeweiligen Klassen. Beispielsweise kann Bill Gates als Mitglied der Klasse *Person* zugeordnet werden. Zusätzlich zur Erstellung von Taxonomien ist es möglich, mittels Rollen (Properties) die Verknüpfungen zwischen den Klassen bzw. Individuen untereinander zu beschreiben. Dabei lassen sich abstrakte Rollen (object properties) und konkrete Rollen (datatype properties) unterscheiden. Abstrakte Rollen definieren eine Beziehung zwischen zwei Individuen von Klassen, konkrete verknüpfen Individuen mit Elementen von Datentypen [10]. Die verfügbaren Datentypen in OWL werden in Tab. 2.2 ersichtlich.

Tabelle 2.2: OWL Datentypen [10].

xsd:string	xsd:boolean	xsd:decimal
xsd:float	xsd:double	xsd:dateTime
xsd:time	xsd:date	xsd:gYearMonth
xsd:gYear	xsd:gMonthDay	xsd:gDay
xsd:gMonth	xsd:hexBinary	xsd:base64Binary
xsd:anyURI	xsd:token	xsd:normalizedString
xsd:language	xsd:NMTOKEN	xsd:positiveInteger
xsd:NCName	xsd:Name	xsd:nonPositiveInteger
xsd:long	xsd:int	xsd:negativeInteger
xsd:short	xsd:byte	xsd:nonNegativeInteger
xsd:unsignedLong	xsd:unsignedInt	xsd:unsignedShort
xsd:unsignedByte	xsd:integer	

Die Rollen können mit *rdfs:subPropertyOf* wiederum in einer hierarchische Struktur abgebildet werden. Zusätzlich ist es möglich globale Restriktionen mit den Sprachkonstrukten *rdfs:domain* und *rdfs:range* auf die jeweiligen Rollen zu setzen.

> A domain of a property limits the individuals to which the property can be applied. If a property relates an individual to another individual, and the property has a class as one of its domains, then the individual must belong to the class. [...] The range of a property limits the individuals that the property may have as its value. If a property relates an individual to another individual, and the property has a class as its range, then the other individual must belong to the range class. [32]

Die Domäne schränkt die Individuen hinsichtlich dessen ein, welcher Klasse die Restriktion zugeordnet werden kann, wohingegen die *range* den

Wertebereich des Individuums einschränkt. Dieser Zusammenhang kann anhand des nachfolgend ersichtlichen Beispiels, dass von [10] entnommen wurde, nachvollzogen werden.

```
<owl:ObjectProperty rdf:about="Zugehörigkeit">
  <rdfs:domain rdf:resource="Person"/>
  <rdfs:range rdf:resource="Organisation"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:about="Vorname">
  <rdfs:domain rdf:resource="Person" />
  <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
```

Es wird festgelegt, dass ein Individuum der Klasse *Person* eine Zugehörigkeit zu einem Individuum aus der Klasse *Organisation* hat. Weiter wird bestimmt, dass der Vorname eines Individuums aus der Klasse *Person* immer vom Datentyp *String* sein muss. Um bessere Möglichkeiten für die Schlussfolgerungsmechanismen zu bieten, können Rollen weiter charakterisiert werden. Folgende Regeln können aufgestellt werden [33]:

- Symmetrie: r(a,b) impliziert r(b,a)
- Transitivität: r(a,b) und r(b,c) impliziert r(a,c)
- Funktionalität: r(a,b) und r(a,c) impliziert b=c
- inverse Funktionalität: r(a,b) und r(a,c) impliziert a=c

Symmetrie bedeutet, dass, wenn A und B in Beziehung stehen, auch B zu A in Beziehung steht. Aus dem unten stehenden OWL-Beispiel von [10] ergibt sich, dass *YorkSure* zu *AnupriyaAnkolekar* in der Beziehung *hatKollegen* steht, woraus sich aus der Symmetrie ergibt, dass *AnupriyaAnkolekar* *YorkSure* auch zum Kollegen hat [10].

```
<owl:ObjectProperty rdf:about="hatKollegen">
  <rdf:type rdf:resource="&owl;TransitiveProperty"/>
  <rdf:type rdf:resource="&owl;SymmetricProperty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="hatProjektleiter">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="istProjektleiterFuer">
  <rdf:type rdf:resource="&owl;InverseFunctionalProperty"/>
</owl:ObjectProperty>
<Person rdf:about="YorkSure">
<hatKollegen rdf:resource="PascalHitzler"/>
<hatKollegen rdf:resource="AnupriyaAnkolekar"/>
<istProjektleiterFuer rdf:resource="SEKT"/>
</Person>
<Projekt rdf:about="SmartWeb">
<hatProjektleiter rdf:resource="PascalHitzler"/>
<hatProjektleiter rdf:resource="HitzlerPascal"/>
</Projekt>
```

Transitivität bedeutet, dass wenn A zu B und B zu C in Beziehung steht, dann steht auch A zu C in Beziehung. Da *AnupriyaAnkolekar hatKollegen*

YorkSure und *YorkSure hatKollegen PascalHitzler* gilt kann gefolgert werden, dass auch *AnupriyaAnkolekar hatKollegen PascalHitzler* gilt. Die Regel der Funktionalität besagt, dass wenn A zu B und A auch zu C in Beziehung steht, B und C identisch sind. Da *hatProjektleiter* funktional ist folgt, dass *PascalHitzler* und *HitzlerPascal* das gleiche Individuum beschreiben. Inverse Funktionalität bedeutet, dass eine inverse Rolle funktional ist [10].

Neben der detaillierten Charakterisierung der Rollen können auf diese weitere Restriktionen erhoben werden. Dabei kann einerseits der Wertebereich eines Properties eingeschränkt werden, andererseits ist es möglich, eine bestimmte Kardinalität anzugeben. Diese Zusammenhänge sollen anhand der folgenden Beispiele erläutert werden. Das erste der folgenden Beispiele, entnommen aus [10], beschreibt die *owl:allValuesFrom* Restriktion. Diese besagt, dass alle Prüfer einer Prüfung Professoren sein müssen, wodurch sich ergibt, dass sich diese Einschränkung auf alle Individuen innerhalb dieser Rolle bezieht. Das zweite der nachfolgenden OWL-Beispiele beschreibt das Sprachkonstrukt *owl:someValuesFrom*, welches spezifiziert, dass jede Prüfung mindestens einen Prüfer haben muss. Durch die Kardinalität lässt sich ein bestimmter Wertebereich für die inkludierten Individuen festlegen. In den angeführten Beispielen wird durch owl:minCardinality bestimmt, dass eine Prüfung mindestens drei Themengebiete haben muss und durch owl:maxCardinality höchstens zwei Prüfer pro Prüfung zugelassen sind [10].

```
<owl:Class rdf:about="Pruefung">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="hatPruefer"/>
      <owl:allValuesFrom rdf:resource="Professor"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="Pruefung">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="hatPruefer"/>
      <owl:someValuesFrom rdf:resource="Person"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="Pruefung">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="hatThema"/>
      <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">
      3
      </owl:minCardinality>
    </owl:Restriction>
```

```
27   </rdfs:subClassOf>
28 </owl:Class>
29
30 <owl:Class rdf:about="'Pruefung"'>
31   <rdfs:subClassOf>
32     <owl:Restriction>
33       <owl:onProperty rdf:resource="hatPruefer"/>
34       <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">
35       2
36       </owl:maxCardinality>
37     </owl:Restriction>
38   </rdfs:subClassOf>
39 </owl:Class>
```

Durch die vorausgegangenen Ausführungen ist ein Überblick über die Web Ontology Language geschafft worden, mit dem für die im weiteren Verlauf dieser Arbeit wichtigsten Konzepte dieser Beschreibungssprache. Für weitere Ausführungen zu diesem Thema wird auf [10], [32] und [33] verwiesen. Auf Grund der Ausführungen zu den Begriffen Ontologie und Ontology Learning, sowie der Erläuterung der Grundkonzepte der Beschreibungssprache OWL ist es möglich ausführlich im nächsten Kapitel auf die bereits vorhandenen Forschungsbeiträgen bezüglich Ontology Learning einzugehen. Es wurde in den vorangegangenen Ausführungen eine Einteilung der Methoden für Ontology Learning anhand der Strukturierung der vorhandenen Daten vorgenommen. Anhand dieser Einteilung sollen im Folgenden die existierenden Techniken für das automatische Erfassen von Wissen innerhalb einer Ontologie vorgestellt und diskutiert werden.

Kapitel 3

Bestehende Arbeiten

Aus den vorangegangenen Ausführungen wurde deutlich, dass man die Wissensquellen für Ontology Learning in die Gruppen strukturiert, semi-strukturiert und unstrukturiert einteilen kann. Nachfolgend soll exemplarisch für jeden dieser Bereiche eine Methodik vorgestellt werden, wie diese Datenformate zu einer Ontologie verarbeitet werden können.

3.1 Wissensquelle Strukturierte Daten

3.1.1 Datenbanken

> Due to the wide use of relational database in information management, a large amount of data about various domains are organized and stored in relational database. Data in relational database may be used as a kind of important resource for ontology learning. [13]

Die große Verbreitung und der häufige Einsatz von relationalen Datenbanken in den verschiedensten Bereichen der Informationstechnologie haben zur Folge, dass ein hohes Maß an Wissen dort abgebildet ist. Diesen Zusammenhang beschreiben Li et. al in [13], in dem sie eine Methode vorstellen, wie man mit Hilfe der von W3C standardisierten formalen Beschreibungssprache OWL eine Ontologie auf Basis von einer oder mehreren relationalen Datenbanken aufbauen kann. Nachfolgend soll die von Li et al. vorgestellte Methode für Ontology Learning erläutert und diskutiert werden. Die Erstellung der Ontologie anhand einer Datenbank folgt nach Li et. al einem regelbasierten Ansatz. Eine ausführliche Erklärung zu den von Li et. al erstellten Regeln findet sich im Anhang A A. Nachfolgend wird der Aufbau des entwickelten Systems beschrieben.

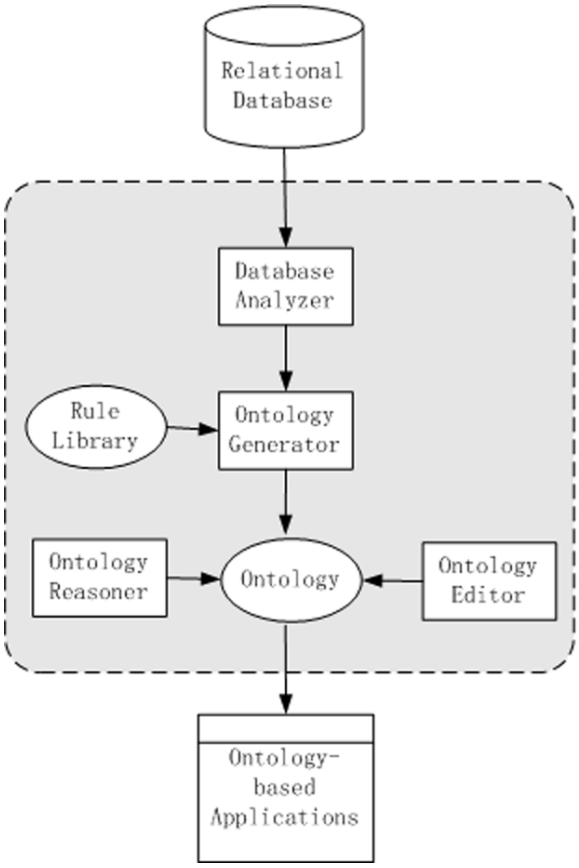

Abbildung 3.1: Ontology Learning Framework [13].

Aufbau des Frameworks

Das entwickelte Framework ist wie in Abb. 3.1 ersichtlich aufgebaut. Das Datenbankschema wird mittels des Datenbankanalysemoduls bezüglich Primär- und Fremdschlüssel, sowie Beziehungen etc. ausgewertet. Anschließend wird mit Hilfe der vorangehend erläuterten Regeln die Ontologie erstellt und kann von einem Benutzer noch weiter modifiziert bzw. auf Korrektheit überprüft werden.

Das vorgestellte System hat den Vorteil, dass es ermöglicht, eine OWL-Ontologie vollautomatisch anhand einer Datenbank zu erstellen. Dabei wird aus den Relationen der Datenbank eine Taxonomie aufgebaut und die Daten anhand von Rollen und Kardinalitäten weiter spezifiziert. Dadurch wird es ermöglicht, maschinelle Schlussfolgerungen aus den erfassten Informationen zu schließen, und auch die weitere Verwendung in Applikationen, beispielsweise für das Semantic Web, ist gewährleistet. Die einzige Einschränkung, die das System hat, ist, dass die Datenbank mindestens in der dritten Nor-

malform vorliegen muss. Dies kann jedoch vernachlässigt werden, da es zuverlässig arbeitende Algorithmen gibt, die diese Umwandlung vornehmen. Nachdem eine Methode aufgezeigt wurde, wie man aus strukturierten Daten eine Ontologie erstellen kann, soll nachfolgend Learning Ontology Strategien für semi-strukturierte Informationen erläutert werden.

3.2 Wissensquelle: Semi-Strukturierte Daten

Semi-strukturierte Daten können einerseits im Format HTML und andererseits in XML vorliegen. Nachfolgend sollen Methoden vorgestellt und diskutiert werden, um anhand von Webdokumenten eine Ontologie zu erstellen. Anschließend werden Ontology Learning Strategien für Daten, die im XML-Format vorliegen, vorgestellt.

3.2.1 HTML

Mit der wachsenden Bedeutung des World Wide Webs in den letzten Jahren wurde auch die strukturierte Aufbereitung bzw. Speicherung von Daten immer essentieller. Für eine bessere maschinelle Verarbeitung und eine hierarchische Darstellung, wie auch aus den vorangegangenen Ausführungen deutlich wurde, ist der Einsatz von Ontologien erforderlich. Eine manuelle Aufbereitung der im Internet verfügbaren Daten ist jedoch mit einem immensen Aufwand verbunden. Auf Grund dessen wurde im Bereich Ontology Learning von Webdokumenten bereits ein hohes Maß an Forschung betrieben. Nachfolgend sollen exemplarisch drei Methoden vorgestellt werden, um anhand von Daten aus dem Internet eine Ontologie zu erstellen. Im Fokus der drei Systeme steht automatisch eine Taxonomie aus verschiedenen Webdokumenten aufzubauen. Der erste Ansatz stammt von Sanchez und Moreno [22] und extrahiert Informationen von verschiedenen Webseiten anhand eines vorher festgelegten Schlagwortes. Davon ausgehend wird einen Taxonomie aufgebaut, die einen allgemeinen Überblick zu dem angegebenen Stichwort und die dazugehörigen Webseiten enthält [22]. In Abb. 3.2 ist ein Überblick über die Funktionsweise des Algorithmus von Sanchez und Moreno [22] gegeben. Der Algorithmus soll im Folgenden schrittweise erläutert werden. Zu Beginn der Ausführung muss ein Schlagwort angegeben werden, das für eine bestimmte Domäne steht. Anschließend wird über die Suchmaschine Google nach relevanten Webseiten gesucht. Dabei wird die Suche nach folgenden Kriterien eingeschränkt:

- Maximale Ergebnisanzahl: Sollten eine große Anzahl an Suchergebnissen auftreten (>= 10000), werden fünf bis zehn Prozent der relevantesten Ergebnisse verwendet.
- Filter für gleiche Seiten: Für generelle Schlagwörter werden durch diesen Filter Webseiten mit der gleichen Web-Domain vernachlässigt. Für

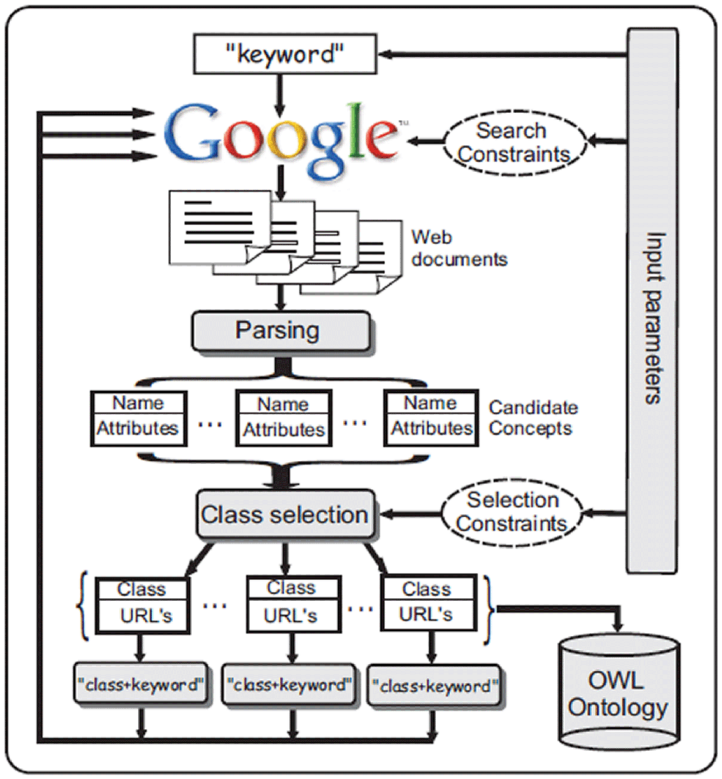

Abbildung 3.2: Ontology Learning von Webdokumenten: Aufbau Framework [22].

spezifischere Schlagwörter mit einer geringeren Ergebnismenge wird dieser Filter nicht angewandt.

Jede gefundene Webseite wird exakt analysiert, was bedeutet, dass bei Formaten, die nicht in HTML vorliegen (pdf, ps etc.) die HTML-Version aus dem Google Cache verwendet wird. Für „Not found“ oder „Unable to show“ Seiten wird wiederum versucht die Daten vom Google Cache zu erhalten. Weiter werden alle Weiterleitungen solange verfolgt bis letztendlich die Webseite gefunden wird und framebasierte Seiten werden auch berücksichtigt. Der Parser liefert anschließend den entdeckten Text zurück, in dem nach dem angegebenen Schlagwort gesucht wurde. Wird dieser gefunden, wird zusätzlich das vorangegangene Wort analysiert. Sollten diese keine Stoppwörter[1], eine Mindestzahl an Zeichen (Standard-ASCII) haben, dann werden die Begriffe als sogenanntes „candidate concept“ gekennzeichnet. Es folgt eine statistische Auswertung der gefundenen Konzepte, woraufhin die Re-

[1]Stoppwörter sind Wortarten, wie beispielsweise Konjunktionen, Disjunktionen, Präpositionen etc..

levanz festgelegt wird. Die Kriterien für diese Beurteilung sind unter anderem die Auftretenshäufigkeit des Begriffs, die Anzahl der Webseiten, die den Ausdruck enthalten, die Anzahl der Suchergebnisse für diesen Begriff etc.. Nachdem die wichtigsten Konzepte herausgefiltert wurden, wird der Prozess rekursiv wiederholt, bis kein Suchergebnis mehr auftritt und es somit keine Subklasse mehr gibt. Daraus ergibt sich eine Taxonomie in Form einer Ontologie. Die für die Ontologie relevanten Konzepte werden in der morphologischen Grundform gespeichert, beispielsweise bei „optical“ würde „optic“ verwendet werden. Am Ende des Algorithmus werden Klassen und Subklassen die die gleichen URLs assoziieren zusammengefasst, um Redundanzen zu vermeiden [22]. Diese Methode hat den Vorteil, dass eine große Anzahl an Informationen aus Webdokumenten zu einer bestimmten Domäne automatisch in einer hierarchischen Struktur abgebildet werden können. Da die Ontologie im OWL-Format erstellt wird, können Inkonsistenzen und Redundanzen mittels Programmen wie Protègè schnell erkannt und beseitigt werden. Problematisch wird es jedoch bei der Evaluationder, sowie der Überprüfung auf die korrekte Zuordnung bzw. den korrekten Aufbau der Taxonomie. Dies müsste ein Experte übernehmen, was mit einem hohen Aufwand verbunden ist. Diese Methode hat daher eher eine unterstützende Funktion beim Erstellen einer solchen Taxonomie, bringt aber auch ein hohes Maß an Zeitersparnis mit sich und kann vor allem im Bereich Information Retrieval eingesetzt werden. Eine weitere Methode eine Ontologie anhand von semi-strukturierten Daten im HTML-Format zu erstellen, wird in Hazman et al. [34] vorgestellt.

Auch im Zuge dieser Methode wird eine Taxonomie aus den gewonnenen Informationen aus den Webdokumenten erstellt. Im Gegensatz zu dem System von [22] werden jedoch nicht mehrere Webseiten automatisch anhand von den anhand eines Schlagwortes gefundenen Suchergebnissen analysiert, sondern die HTML-Struktur und die Texte in den Überschriften der Webseite. In Abb. 3.3 kann der Aufbau des Frameworks nachvollzogen werden. Es besteht aus insgesamt sieben Modulen, die, in der folgenden Aufzählung, entommen aus [34], kurz beschrieben werden sollen:

- Heading Extractor: Dieses Modul extrahiert die Überschriften aus dem HTML-Dokument und speichert diese in eine Datenbank
- Heading Processor: In diesem Bereich werden die textuellen Daten der Überschriften durch Stoppwortentfernung aufbereitet
- N-gram based Ontology Learner: In diesem Modul wird die Taxonomie anhand der Überschriften aufgebaut.
- N-gram ontology refiner: Dieser Teil des Frameworks überarbeitet die aufgebaute Ontologie und entfernt Konzepte, dessen Gewichtung unter einem gegebenen Wert liegen.
- HTML structure based Ontology learner: Dieses Modul baut eine Ontologie anhand der Hierarchie der Überschriften des Webdokuments auf

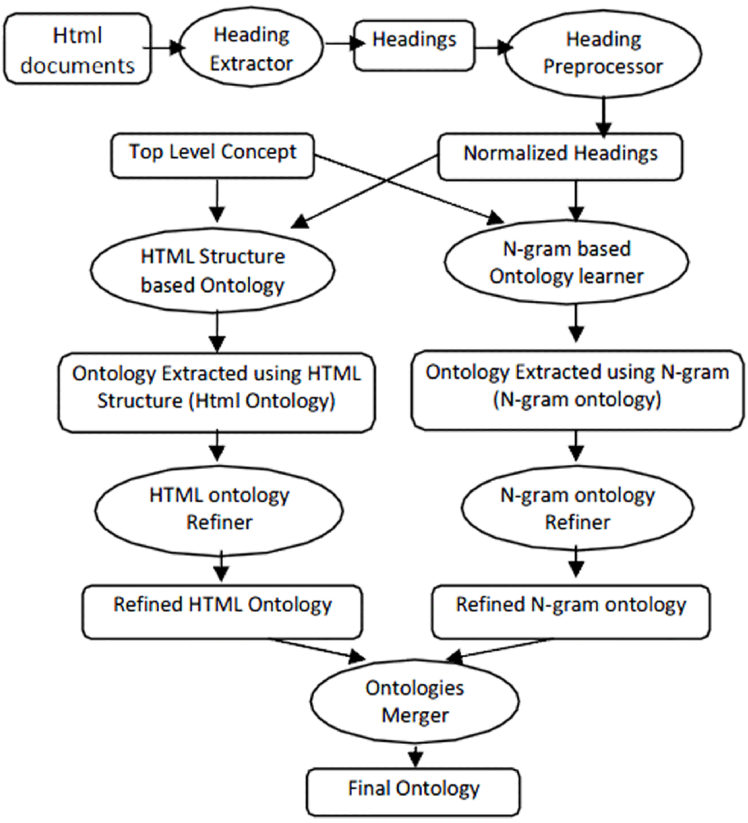

Abbildung 3.3: Ontology Learning von Webdokumenten: Aufbau Framework [34].

- HTML ontology refiner: Dieser Bereich überarbeitet die durch den HTML Ontology Learner gewonnene Ontologie, indem neue Konzepte erkannt werden, die bereits erstellten Konzepten untergeordnet sind.
- Ontologies Merger: Im Zuge dieses Moduls werden die erstellten Ontologien zu einer zusammengefasst.

Mittels dieser Methode ist es wiederum möglich, eine Taxonomie anhand von domänenspezifischen Webdokumenten aufzubauen. Die Evaluation des Systems hat jedoch erwiesen, dass die Ergebnisse hinsichtlich von precision und recall noch nicht zufriedenstellend sind. Das Auswerten von HTML-Überschriften zum Erstellen einer Taxonomie scheint jedoch durchaus als praktikabel, jedoch muss dahingehend an den Extraktionsregeln noch weiter gearbeitet werden. Weiter wäre es interessant, zusätzlich zur Auswertung der Überschriften auch den dazugehörigen Text zu analysieren, um weitere Informationen in die Ontologie übernehmen zu können. Dies war jedoch bei diesem Ansatz nicht intendiert. Die dritte Methode wurde von Davulcu et al.

[8] entwickelt und kann wegen dem begrenzten Rahmen dieser Arbeit nicht im Detail vorgestellt werden. Im Falle von OntoMiner wird die Domäne von Benutzer des Systems durch Angabe von zehn bis fünfzehn relevanter Webseiten selbst bestimmt. Anschließend erkennt das System Gemeinsamkeiten in den Webdokumenten und wandelt diese in hierarchische Strukturen in Form von XML um. Weiter parst ein Algorithmus die Homepages und identifiziert die zentralen Konzepte der Domäne. Daraus wird eine Taxonomie erstellt und Relationen zwischen den verschiedenen Kategorien bestimmt. Außerdem wird die erstellte Taxonomie durch selektives Crawlen ausgehende von dem zentralen Konzept, um zugehörige Links erweitert. Zusätzlich werden nicht relevante Daten wie Werbebanner oder Navigationsleisten nicht berücksichtigt [8]. Ein Vorteil der von Davulcu et al. [8] angewandten Methodik ist, dass die HTML-Tags keine Rolle für die Erstellung der Taxonomie einnehmen, wodurch Webseiten unterschiedlichster Struktur erfasst werden können.

Die drei vorgestellten Methoden haben aufgezeigt, dass sich Webdokumente für Ontology Learning eignen. Es wurde auch deutlich, dass es sich bei den erstellten Ontologien eher um reine Taxonomien handelt. Dadurch sind maschinelle Schlussfolgerungen nur begrenzt möglich, jedoch können bestimmte Domänen in einer hierarchischen Struktur abgebildet werden und beispielsweise in Semantic Web Applikationen weiter verwendet werden. Nachfolgend sollen Methoden vorgestellt werden, wie semi-strukturierte Daten im XML-Format für lernende Ontologien verwendet werden können.

3.2.2 XML

> In the recent years, XML (eXtensible Markup Language) has reached a wide acceptance as the relevant standardization for storing and exchanging data on the Web. [27]

Die große Verbreitung von XML als Datenaustauschformat hat zur Folge, dass eine große Menge an Informationen und Wissen in diesem Format vorhanden ist. XML bietet dabei eine Reihe von Vorteilen. Es verwendet zur Repräsentation der Daten natürliche Sprache und ist somit für jeden lesbar und verständlich. Weiter können alle verbreiteten Programmiersprachen, wie C++ oder Java mit diesem Format interagieren. Außerdem ist das Format offen gestaltet, woraus folgt, dass jeder seine eigenen Tags erstellen und einbinden kann. Dies kann jedoch auch dazu führen, dass es zu Missverständnissen hinsichtlich der Tagnamen kommt, da verschiedene Benutzer unterschiedliche Begrifflichkeiten für denselben Zusammenhang verwenden. Dadurch kann sich die Verarbeitung durch den Computer unter Umständen sehr schwierig gestalten. Zudem steht die Strukturierung von Daten im Fokus von XML und nicht die semantische Auszeichnung. Deswegen sind maschinelle Schlussfolgerungen anhand von XML-Dokumenten kaum möglich.

Deswegen ist es essentiell Möglichkeiten zu finden die Daten im XML-Format in wissensbasierte Beschreibungssprachen wie OWL umzuwandeln [27]. Im Feld Ontology Learning gibt es hierzu zwei verschiedene Ansätze. Einerseits wird versucht die Daten von XML mittels XML-Schema in die Formate RDF oder OWL umzuwandeln. Andererseits bestehen Methoden eine Ontologie anhand von Dokumenttypdefinitionen (DTD) zu erstellen. Beide Formate DTD und XML-Schema dienen dazu, das vorliegende XML-Dokument hinsichtlich der Struktur, der Inhalte und der Attribute näher zu spezifizieren. Zum besseren Verständnis dienen die nachfolgenden Definitionen von [35] zu den beiden Begriffen:

> A Document Type Definition (DTD) defines the legal building blocks of an XML document. It defines the document structure with a list of legal elements and attributes. A DTD can be declared inline inside an XML document, or as an external reference.
>
> The purpose of an XML Schema is to define the legal building blocks of an XML document, just like a DTD. An XML Schema:
>
> - defines elements that can appear in a document
> - defines attributes that can appear in a document
> - defines which elements are child elements
> - defines the order of child elements
> - defines the number of child elements
> - defines whether an element is empty or can include text
> - defines data types for elements and attributes
> - defines default and fixed values for elements and attributes

Weitere Informationen zu diesen Begriffen können innerhalb der Tutorien unter [36] und [35] gefunden werden. Nach Klärung dieser beiden Begriffe können nachfolgend bestehende Methoden vorgestellt werden, um aus XML-Dokumenten Ontologien zu erstellen. Im Paper von Ferdinand et al. [9] wird eine Technik vorgestellt, die aus zwei Komponenten besteht. Dabei werden aus XML-Dokumenten ein RDF-Graph erstellt und aus XML-Schemata eine OWL-Ontologie. Die erstellten Dokumente sind unabhängig voneinander, woraus resultiert, dass unter Umständen das OWL-Model nicht zu den Instanzen im RDF-Graph passt. Weiter wird im Zuge dieses Ansatzes nicht thematisiert, wie man vorgehen kann, wenn kein XML-Schema vorhanden ist [27]. Dies muss demnach immer vorhanden sein, damit eine Ontologie mittels dieser Methode erstellt werden kann. Eine weitere Methode wird von Bohring und Auer [3] vorgestellt, in der wiederum anhand eines XML-Schemas eine OWL-Ontologie aufgebaut wird. In diesem Ansatz werden die Sprachkonstrukte von XML-Schema mittels OWL-Elementen abgebildet, wie in Abb. 3.1 dargestellt wird.

Tabelle 3.1: Abbilden von XML-Schema in OWL [3].

XSD	*OWL*
xsd:elements, containing other elements or having at least one attribute	owl:Class, coupled with owl:ObjectProperties
xsd:elements, with neither sub-elements nor attributes	owl:DatatypeProperties
named xsd:complexType	owl:Class
named xsd:SimpleType	owl:DatatypeProperties
xsd:minOccurs, xsd:maxOccurs	owl:minCardinality, owl:maxCardinality
xsd:sentence, xsd:all	owl:intersectionOf
xsd:choice	combination of owl:intersectionO, owl:unionOf and owl:complementOf

Im Gegensatz zur Methode von [9] wird bei [3] automatisch ein XML-Schema anhand eines Stylesheets erstellt. Dadurch ist es möglich, auch XML-Dokumente, die noch kein Schema enthalten, zu verarbeiten. Weiter wird versucht, anhand der XML-Struktur Relationen zu extrahieren, die in der Ontologie abgebildet werden. Im Moment werden durch diese Methode jedoch nicht immer optimale Ergebnisse erzielt, da vor allem wenn kein XML-Schema vorhanden ist, die Umwandlung stark auf heuristischen Methoden beruht [3]. Das führt dazu, dass eine manuelle Nachbearbeitung der erstellten Ontologie vonnöten ist. Ein weiteres System zur Erstellung einer Ontologie anhand semi-strukturierter Daten wird in [1] vorgestellt. Im Zuge dieses Ansatzes werden zwei Technologien verwendet. Einerseits wird die Struktur und der Aufbau des XML-Dokuments geparst und ausgewertet. Andererseits werden mittels Natural Language Processing[2] die Texte im Dokument analysiert, um neue Klassen und Relationen in der Ontologie zu definieren. Der Vorteil dieser Methode ist, dass die Qualität der Ontologie nicht so stark von der Qualität des XML-Dokuments abhängt, da durch die Textanalyse Inkonsistenzen und Redundanzen vermieden werden können. Die von Rodrigues et al. [21], sowie das von Cruz und Nicolle [7] vorgestellten Frameworks stellen das Erweitern einer existierenden Ontologie in den Fokus ihrer Ausführungen. Beide Ansätze folgen spezifischen Umwandlungsregeln, die für ein jeweiliges XML-Schema gültig sind. Die Erstellung dieser Regeln führt zu einem hohen manuellen Aufwand, wodurch diese Methoden für größere Datenmengen eher inpraktikabel sind. Abschließend soll auf die von Thuy et al. [27] vorgestellte Methode eingegangen werden, die es ermöglicht, anhand eines Dokumentbeschreibungstyps eine OWL-Ontologie zu erstellen. In Abb. 3.4 wird der generelle Aufbau des Frameworks dargestellt. Im ersten Schritt wird ein DTD Schema, falls nicht vorhanden, mittels des Programms HiTSoftware erstellt.

[2]Dafür wurde das Gate Framework verwendet. Nähere Informationen dazu können unter [37] bezogen werden.

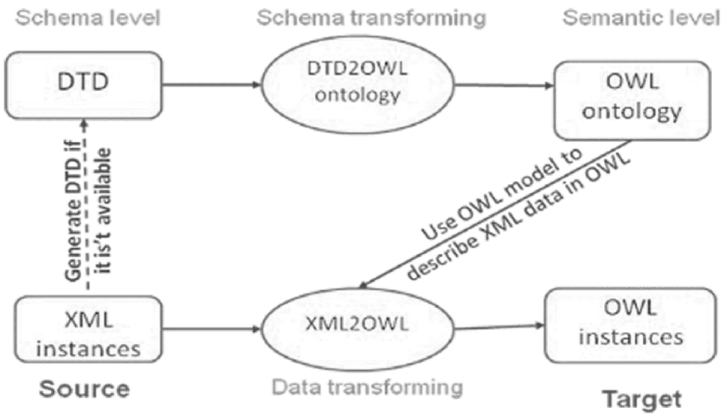

Abbildung 3.4: Aufbau Framework DTD to XML [27].

Anschließend wird anhand der Definitionen im DTD eine OWL-Ontologie aufgebaut. Dabei werden die Struktur, die Elementnamen und die Datentypen aus dem DTD übernommen. Weiter werden Relationen zwischen den Elementen im XML identifiziert und in der Ontologie abgebildet. Außerdem werden in diesem Schritt Namenskonflikte zwischen Elementen erkannt und behoben. Abschließend werden anhand des XML-Dokuments die Individuen für die jeweiligen Klassen erstellt [27]. Diese Methode ermöglicht es, anhand von Daten im XML-Format vollständige Ontologien im OWL-Format zu erstellten. Es werden anhand des DTD Schemas Klassen, Rollen, Individuen, sowie rudimentär Restriktionen erstellt. Dadurch ist eine weitere maschinelle Verarbeitung vor allem in Semantic Web Applikationen möglich, ohne dass ein großer manueller Aufwand entsteht.

Aus den vorangegangen Ausführungen wurde deutlich, dass die automatische Erstellung von Ontologien anhand von semi-strukturierten Daten im XML-Format möglich ist. Die meisten Methoden sind jedoch noch nicht ganz ausgereift und mitunter mit einem zusätzlichen manuellen Aufwand verbunden. Der von Thuy et al. [27] vorgestellte Ansatz scheint praktikabel zu sein, da die Methodik automatisch eine Ontologie mit einem hohen semantischen Gehalt erstellt und darüber hinaus ohne Benutzeraufwand auf verschiedenste XML-Dokumente angewandt werden kann. Den Abschluss dieses Kapitels sollen Erläuterungen zu den Methoden bilden, um aus unstrukturierten Daten eine Ontologie aufzubauen. Dabei wird das Framework TEXT-TO-ONTO in den Mittelpunkt der Ausführungen gestellt.

3.3 Wissensquelle Unstrukturierte Daten

3.3.1 Texte

Neben den bereits aufgeführten Methoden eine Ontologie anhand strukturierter oder semi-strukturierter Daten zu erstellen, besteht die Möglichkeit, unstrukturierte Wissensquellen, wie natürlich sprachliche Texte, zu verwenden. Durch große Verbreitung des World Wide Webs ist auch eine immense Anzahl an Dokumenten zu den verschiedensten Themengebieten verfügbar geworden. Daraus resultiert ein sehr hohes Maß an Wissen, dass für die Erstellung von Ontologien verfügbar ist. Dem großen Potential von Texten als Wissensquelle steht jedoch der komplexe Vorgang des maschinellen Analysierens von natürlich sprachlichen Dokumenten entgegen. Trotz dieser hohen Komplexität wurden einige Frameworks und Systeme zur Erstellung einer Ontologie anhand von natürlich sprachlichen Texten entwickelt, die im Folgenden erläutert werden sollen. Im Allgemeinen lässt sich bei allen Ansätzen feststellen, dass sie eine unterstützende Rolle bei der Erstellung einer Ontologie einnehmen. Es handelt sich auf Grund dessen um semi-automatische Verfahren im Bereich Ontology Learning. Im Paper von Hu und Liu [11] wird ein Framework vorgestellt, dass eine Ontologie für allgemeine aber auch spezifische Domänen erstellen kann. Als Wissensquelle dient ein Korpus an natürlich sprachlichen Texten. Aus diesen Dokumenten werden die zentralen Begriffe bzw. Fachausdrücke extrahiert und mittels Natural Language Processing weiter verarbeitet. Im Hintergrund des Systems wird mit WordNet[3] gearbeitet, wodurch ermöglicht wird, semantische Konzepte und Relationen zwischen den extrahierten Termen festzulegen. Auf Basis dieser Begriffe und aus dem WordNet gewonnener Wortverknüpfungen, wie Antonymie, Hyperonymie oder Hyponymen, wird eine OWL-Ontologie erstellt [11]. Die nächsten Ansätze konzentrieren sich darauf eine Ontologie anhand domänenspezifischer Textkorpora zu generieren. Das von Biébow et al. [2] vorgestellte System extrahiert, auf Basis der statistischen Auftretenshäufigkeit von Termen, mögliche Klassen für eine Ontologie. Anschließend kann ein Experte diese Begriffe weiter zu einer Ontologie verarbeiten und bietet somit im Gegensatz zu den anderen vorgestellten Systemen nur limitierte Möglichkeiten hinsichtlich der Automatisierung des Erstellungsprozesses einer Ontologie. Faure und Nédellec stellen in ihrem Paper [39] eine Methodik vor, eine Taxonomie anhand natürlich sprachlichen technischen Texten zu erstellen.

> The system is able to acquire taxonomic relations and subcategorization frames of verbs based on syntactic input. The ASIUM

[3] WordNet® is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations [38]. Weitere Informationen zum WordNet können unter [38] gefunden werden.

> system hierarchically clusters nouns based on the verbs that they co-occur with and vice versa [15].

Dieses System betreibt eine syntaktische Analyse von Texten, bei der die bedeutungstragenden Satzteile extrahiert werden. Stoppwörter und Adjektive werden hierbei entfernt. Anhand des Auftretens von Verben in Kombination mit Substantiven werden Konzepte bzw. Kategorien für die Ontologie definiert. Je öfter dabei das Nomen mit dem Verb auftritt, desto korrekter ist das Ergebnis. Auf Basis dieser Konzepte wird anschließend die Taxonomie für die Ontologie erstellt. Die Beschreibung der genauen Funktionsweise der einzelnen Schritte des Natural Language Processing Prozesses und des maschinellen Lernens würde bei Weitem den Rahmen dieser Arbeit überschreiten. An Stelle dessen soll auf Philipp Cimiano [6] verwiesen werden, der in diesem Buch detailliert die Grundlagen und Methoden für die Erstellung einer Ontologie anhand natürlich sprachlicher Texte erläutert. Diese Dissertation basiert unter anderem auch auf den Forschungsergebnissen von Alexander Mädche und Steffen Stab hinsichtlich ihres Frameworks Text-to-Onto, das im Folgenden in den Fokus der Ausführungen gestellt werden sollen. Das von Mädche und Staab [15] vorgestellte System folgt wiederum einem semiautomatischen Prozess zur Erstellung einer Ontologie anhand domänenspezifischer Dokumente. Dabei konzentrieren sich die Autoren nicht ausschließlich darauf eine Taxonomie zu erstellen, sondern auch nicht-hierarchische Relationen zwischen Konzepten aus den Texten zu gewinnen. Mittels einer „Analyse von Wörtern und ihrer textlichen Umgebung (shallow text processing)“ [40] werden Wortpaare identifiziert und für den Aufbau einer Taxonomie aufbereitet. Mit Hilfe eines lernenden Algorithmus und der erstellten Hierarchie werden abstraktere Relationen aus den extrahierten Termen definiert [15]. In Abb. 3.5 sind die zentralen Komponenten des Frameworks aufgeführt, die im Folgenden näher erläutert werden sollen. Die „Text & Processing Management Component“ wird vom Ersteller der Ontologie genutzt, um Texte eines bestimmten Themengebiets auszuwählen, die vorher gesammelt wurden. Außerdem werden Methoden vom ‘Text Processing Server“ und Algorithmen der „Learning & Discovering“ Komponente ausgewählt. Der „Text Processing Server“ arbeitet mit Natural Language Processing Methoden, die vom System SMES (Saarbrücken Message Extraction System)[4] bereitgestellt werden. Die Ergebnisse werden in einem XML-Dokument gespeichert, das den mit Annotationen versehenen Text enthält und zur weiteren Verarbeitung an die „Learning & Discovering“ Komponente übergeben wird. Das Modul „Lexical DB & Domain Lexicon“ enthält ein Lexikon mit lexikalischem Wissen, das für die syntaktische Analyse der Dokumente nötig ist. Daraus folgt, dass in diesem Lexikon syntaktische mit semantischen Informationen verknüpft sind, die für die Erstellung der Ontologie gebraucht wird. Die „Learning &

[4] Genaue Spezifikationen und Erläuterungen zu den Kernfunktionen von SMES können unter [17] nachvollzogen werden.

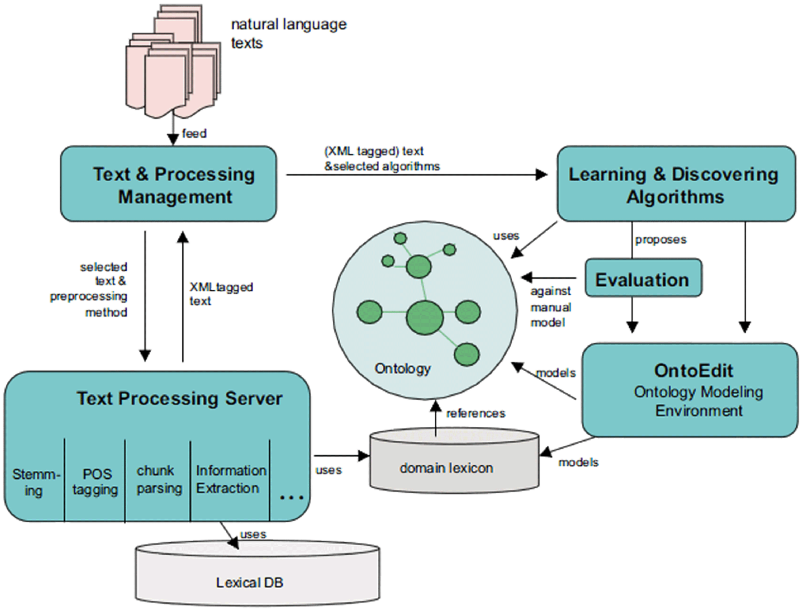

Abbildung 3.5: Aufbau des Frameworks Text-To-Onto [15].

Discovering“ Komponente analysiert den annotierten Text mit verschiedenen Methoden, wie Extraktion von Termen („Term Extraction“), um wichtige Begriffe für die Taxonomie zu erschließen. Nicht-hierarchische Relationen werden mittels eines lernenden Algorithmus anhand allgemeiner Assoziationsregeln definiert. Die erstellte Ontologie kann im Modul „Ontology Engineering Environment“ durch einen Experten weiter bearbeitet werden. Dabei bestehen in OntoEdit[5] neben den Möglichkeiten zur Bearbeitung der Ontologie Optionen zur Wartung, Dokumentation und zum Austausch zwischen Ontologien. In Abb. 3.6 ist die Benutzeroberfläche von OntoEdit dargestellt, wobei unter anderem die Taxonomie, eine Relation, das erstellte Lexikon, ein annotiertes XML-Dokument und die Visualisierung der Ontologie erkennbar wird [15]. Das Framework Text-To-Onto bietet von allen vorgestellten Methoden die meisten Funktionen und wurde deshalb exemplarisch für eine Möglichkeit zum Erstellen einer Ontologie anhand natürlich sprachlicher Texte näher ausgeführt. Das System bietet momentan eine große Anzahl an Methoden und Algorithmen für die Analyse von Dokumenten. Wegen der guten Erweiterbarkeit ist es jedoch denkbar, weitere Ansätze in dieses System zu integrieren. Es existieren noch zahlreiche weitere Methoden um aus unstrukturierten Daten eine Ontologie zu erstellen. Wegen des begrenzten Rahmens dieser Arbeit soll an dieser Stelle auf die Forschungsbeiträge von

[5]Eine komplette Beschreibung der Funktionen dieses Editors für Ontologien kann unter [24] und [26] gefunden werden.

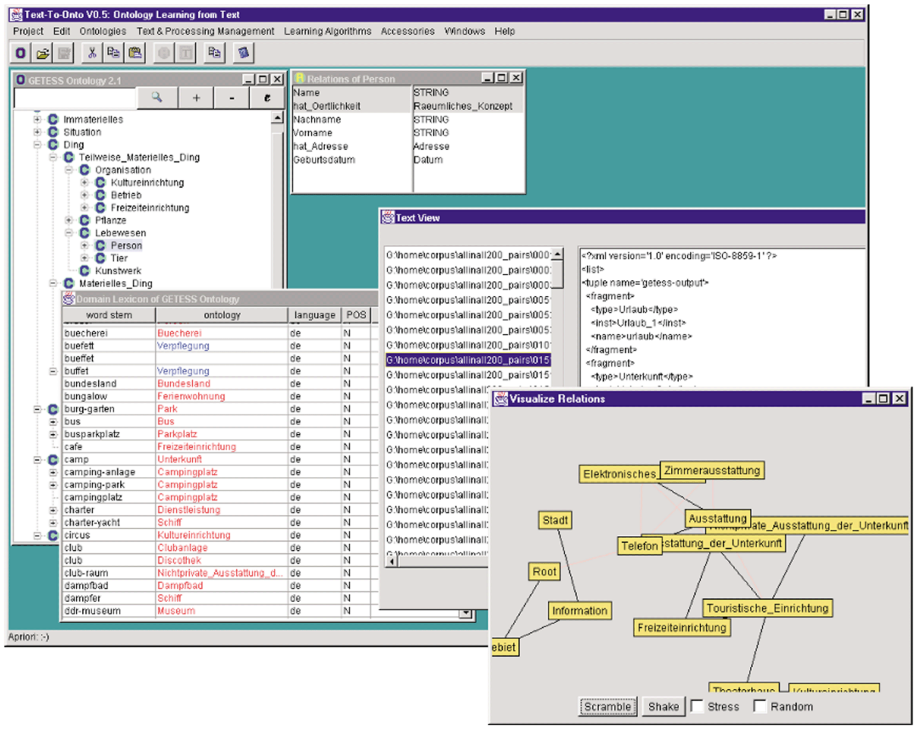

Abbildung 3.6: OntoEdit [15].

Yamaguchi [28] und Buitelaar et al. [4] verwiesen werden.

Aus den vorangegangenen Ausführungen hinsichtlich der Verarbeitung von unstrukturierten, semi-strukturierten und strukturierten Daten zu einer Ontologie wurde deutlich, dass eine große Anzahl an Methoden dafür zur Verfügung steht. Im folgenden Kapitel soll die Verwendung von Foren als Wissensquelle für Ontology Learning erläutert und diskutiert werden. Dabei wird auf die Wissensquelle Board-System eingegangen. Weiter wird der Aufbau von Foren und Board-Systemen beschrieben, wobei zusätzlich eine Abgrenzung der beiden Begriffen voneinander vorgenommen wird. Anschließend folgt eine Erläuterung der Systemarchitektur der entwickelten Software BoardToOntology und im Zuge dessen eine Beschreibung der angewandten Methodik.

Kapitel 4

BoardToOntology - Methodik

4.1 Wissensquelle Board-System

> Das Forum ist ein Treffpunkt (lat. forum = Marktplatz) für die Besucher Ihrer Website. Je nach der dort behandelten Thematik wird Ihr Forum zu einer „gemütlichen Bank“, auf die man sich setzen kann, wenn man vom vielen Herumrennen müde ist, um Leute zu treffen, mit denen man einen netten Schnack [sic] halten kann, oder es wird zu einem Akademikertreff, in dem interessante Fachdiskussionen laufen [sic] – die Übergänge sind fließend, ebenso, wie es im „realen Leben“ wenig reine Kaffeeklatscher und wenig reine Fachleute gibt. [41]

Das einleitende Zitat von Astrid Steinmann formuliert den Zweck eines Forums zwar in einem legeren Sprachstil, jedoch werden die zentralen Merkmale eines Forums damit treffend festgelegt. Es wird beschrieben, dass mit Forensystemen eine Plattform zur Verfügung gestellt wird, in der sich Personen über bestimmte Themengebiete austauschen und ihr Wissen verbreiten können. Dabei ergibt sich eine große Vielfalt an abgebildeten bzw. diskutierten Inhalten, was sich mit den verschiedenen Interessensgebieten der Benutzer begründen lässt. Weiter haben Internetforen die Eigenschaft, es jedem Interessierten zu ermöglichen, einen neuen themenspezifischen Beitrag zu formulieren oder einen bereits bestehenden zu kommentieren. Dadurch können Wissensbasen entstehen, die eine bestimmte Domäne beschreiben bzw. ein hohes Maß an Informationen über diese bereitstellen.

An dieser Stelle setzen diese Arbeit und die in diesem Rahmen entwickelte Software an. Der Fokus lag im Zuge dessen auf der Verarbeitung der Forendaten zu einer Ontologie mittels Methoden aus dem Feld Ontology Learning. Daraus ergab sich die Fragestellung, ob sich die Inhalte aus Foren als Wissensquelle für den Aufbau einer Ontologie eignen. Außerdem wurde in der entwickelten Software BoardToOntology eine exemplarische Implementierung eines solchen Systems vorgenommen. Vorangehend wurden Metho-

den vorgestellt, die es ermöglichen aus verschiedenen Datenquellen eine Ontologie zu gewinnen. Dabei war vor allem die Art der Strukturierung der vorhandenen Wissensquelle von Bedeutung. Diese Einteilung lässt sich auch bei der Entwicklung der Software wiederfinden, wobei einerseits die allgemeine Beschreibung des Forums im XML-Format und somit in semi-strukturierter Form verarbeitet wurde. Andererseits wurden die unstrukturierten textuellen Daten aus den Foreneinträgen und dem Beitragstitel analysiert. Im Zuge dessen wurde versucht, eine Verschlagwortung der einzelnen Foreneinträge vorzunehmen und zusätzlich eine Taxonomie zu erstellen. Die Sprache des Forums wurde auf Deutsch, das Themengebiet auf die Softwareentwicklung für mobile Endgeräte festgelegt. Auf Grund der technischen Domäne ist eine hohe Anzahl von englischen Wörtern zu finden, was in der Verarbeitung der textuellen Daten berücksichtigt werden musste. Wie bereits vorangehend beschrieben, werden die Forenbeiträge ausschließlich von Benutzern erstellt, wodurch sich unter anderem ein verminderter Sprachstil ergibt. Grundlegend ist für den Einsatz der implementierten Methodik Voraussetzung, dass das Forum über eine Sitemap verfügt. Anhand dieser werden alle verfügbaren Daten über die jeweiligen Forenbeiträge gesammelt.

Die Auswahl des Forschungsgegenstandes fiel auf das Forum Android-Hilfe.de, welches auf dem Forensystem vBulletin aufbaut. Auf Grund der großen Anzahl an Themengebieten über das mobile Betriebssystem Android wurde die Domäne auf die Rubrik der Applikationsentwicklung weiter eingeschränkt. Im Gegensatz zu den bereits vorgestellten Systemen zur maschinellen Erstellung einer Ontologie anhand unstrukturierter Daten muss in diesem Fall kein Textkorpus über ein bestimmtes Themengebiet zusammengestellt werden. Die Sammlung der Daten ist mit Auswahl des jeweiligen Forums abgeschlossen und auf Grund der hohen Anzahl von Internetforen kann ein hohes Maß an Themengebieten erschlossen werden. Die entwickelte Methodik kann auf dem derzeitigen Stand jedoch nur auf das ausgewählte Forum angewandt werden und eine Berücksichtigung von Daten aus anderen Foren ist nicht möglich. Dies war jedoch auch nicht Gegenstand der Forschungsfrage und war dementsprechend vernachlässigbar. Das Forschungssubjekt verfügt über eine Suchfunktion, womit nach bestimmten Themen und Beiträgen gesucht werden kann. Dabei ist es unter anderem auch möglich nach bestimmten Stichwörtern zu suchen, welche vom Benutzer vergeben werden müssen [42].

Im Folgenden soll auf den grundlegenden Aufbau eines Forums eingegangen werden. Dabei sollen die vorhandenen Daten beschrieben werden, sowie die hierarchische Struktur der Forenbeiträge. Anschließend soll der vorherrschende Sprachstil im Forum Android-Hilfe.de analysiert werden. Darauf aufbauend folgt eine ausführliche Beschreibung über die einzelnen Module der im Rahmen dieser Arbeit entwickelten Software.

4.2 Forenaufbau

> Bei der Modellierung von web-basierten Diskussionsplattformen haben sich mittlerweile zwei Modellierungstypen durchgesetzt: Foren und Boards. [...] In der Praxis werden sie dagegen meist unreflektiert verwendet, vor allem auch deshalb, weil es bislang nur wenige Reflexionen zu ihrer begrifflichen Unterscheidung gibt. [43]

Stefan Münz beschreibt in [43] den prinzipiellen Aufbau von Diskussionsplattformen im Internet. Dabei unterscheidet er zwischen *Forum* und *Board* und zeigt die terminologische Unschärfe in der Verwendung dieser Begriffe auf. In welchen Punkten die Unterscheidung ansetzt, soll im Folgenden thematisiert werden. Die weiteren Untersuchungen hinsichtlich des Forschungsgegenstands Android-Hilfe.de nehmen die folgenden Ausführungen als Grundlage. Der größte Unterschied zwischen einem Forum und einem Board-System liegt in der Ordnung und Strukturierung der jeweiligen Beiträge der Benutzer.

> Die gedankliche Basis-Einheit eines Forums ist der so genannte Thread. Dieses englische Wort, das etymologisch mit dem deutschen Wort Draht verwandt ist, lässt sich heute ins Deutsche je nach Kontext mit Faden, Kette, Garn oder Gewinde übersetzen. Auf eine Diskussion übertragen, bezeichnet es den Diskussionsfaden oder die Verkettung aller Beiträge zu der betreffenden Diskussion. [43]

Im Forum werden einer bestimmten These die einzelnen Beiträge in einer Baumstruktur angeordnet. Diese Behauptung bewegt sich üblicherweise im Diskurs des Threads. Es kann jedoch auch zu „Thread-Drifts“ kommen, wobei die Diskussion über ein bestimmtes Detail eines Beitrags weitergeführt wird und innerhalb eines ursprünglichen Themengebiets liegt. Die direkten Antworten auf das sogenannte Ausgangs-Posting werden auf der nächsten Unterebene dargestellt. Beiträge, die sich wiederum auf die Antworten beziehen, werden diesen ungeordnet und so weiter. Dabei sind die Diskussionsstandpunkte auf der jeweiligen Ebene nach Aktualität sortiert. Die beschriebene Hierarchie kann in Abb. 4.1 nachvollzogen werden. In dieser Grafik wäre der Ausgangspunkt für die Diskussion der Knoten eins. Auf der nächsten Ebene ist Knoten vier der neuste, gefolgt von drei und zwei. Diese Darstellungsform kann jedoch dazu führen, dass die Beiträge hinsichtlich der Aktualität falsch interpretiert werden könnten. Der Knoten sieben ist unter Umständen aktueller als der Knoten neun, obwohl dieser weiter oben visualisiert wird [43]. Im Gegensatz zu dieser Baumstruktur bei Foren steht die lineare Anordnung der Beiträge in einem Board-System. Es wird zudem nicht von einem Leitfaden für die Diskussion ausgegangen, sondern die Antworten auf ein bestimmtes Thema chronologisch angereiht, wie in Abb. 4.1

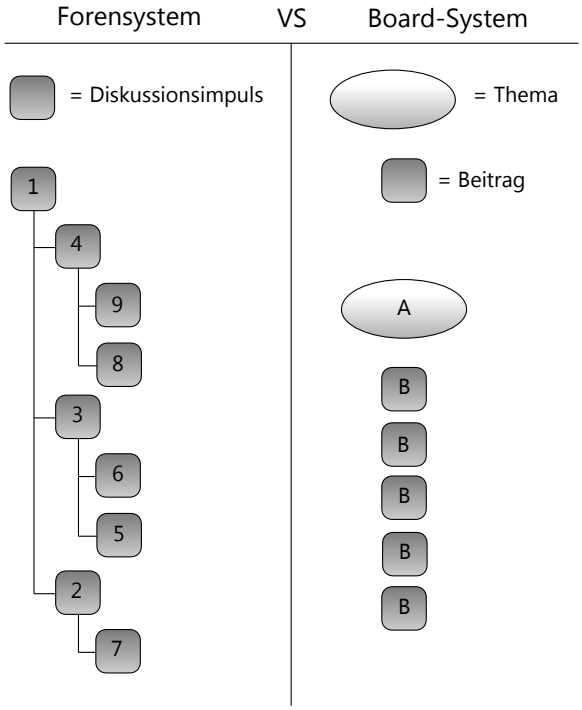

Abbildung 4.1: Gegenüberstellung der Struktur von Forum und Board-System [43].

deutlich wird. Dadurch kann sich beispielsweise die fünfte Antwort auch auf die zweite beziehen und eventuell nichts mehr mit dem eigentlichen Thema zu tun haben. Die grundlegende Metapher hinter Boards ist somit nicht der Leitfaden, an dem neue Standpunkte mittels Knoten angehängt werden, sondern eine Liste. Ein weiterer Unterschied zwischen Foren und Boards liegt in den Diskussionsgegenständen. Während es in einem Forum vorkommen kann, dass sich Diskussion vom eigentlichen Themengebiet entfernen, beziehen sich die Themen und die Antworten darauf auf die jeweilige Domäne des Board-Systems [43]. Der Forschungsgegenstand Android-Hilfe.de ist gemäß den vorangehenden Ausführungen den Boards zuzuordnen und basiert auf der Software vBulletin.[1] Nachfolgend soll der Aufbau des in dieser Arbeit behandelten Forschungssubjekts erläutert werden. Android-Hilfe.de folgt bei der Darstellung der jeweiligen Themengebiete den vorangehend festgelegten Merkmalen eines Board-Systems. Aufgrund der immensen Anzahl an Themengebieten über die mobile Plattform Android sind Kategorien eingeführt worden. Neben dem ausgewählten Bereich App-Entwicklung, bestehen beispielsweise Rubriken zum Betriebssystem Google Android und den verschiedenen Anbietern von Android Smartphones. Nach Auswahl einer gesuchten

[1] Weitere Informationen können der offiziellen Webseite unter [44] entnommen werden.

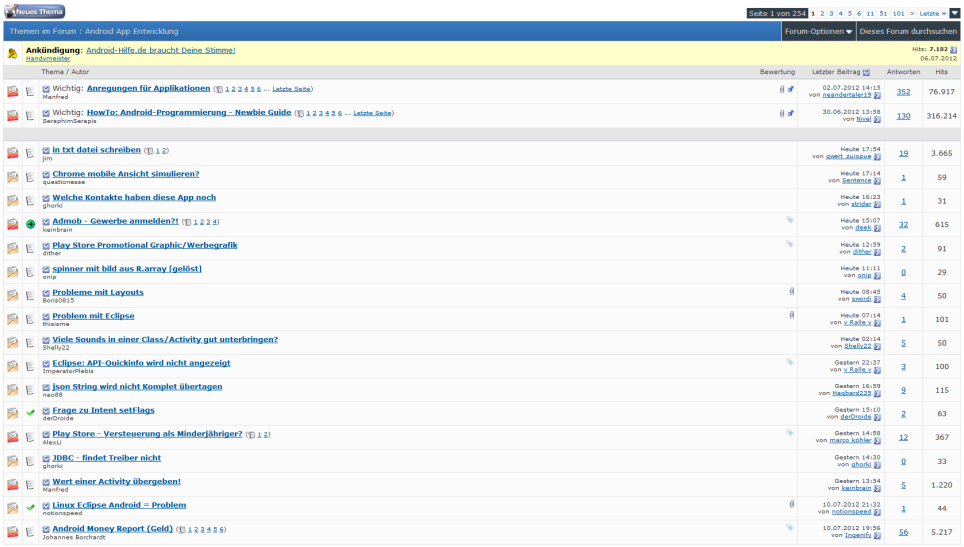

Abbildung 4.2: Auflistung der Themen.

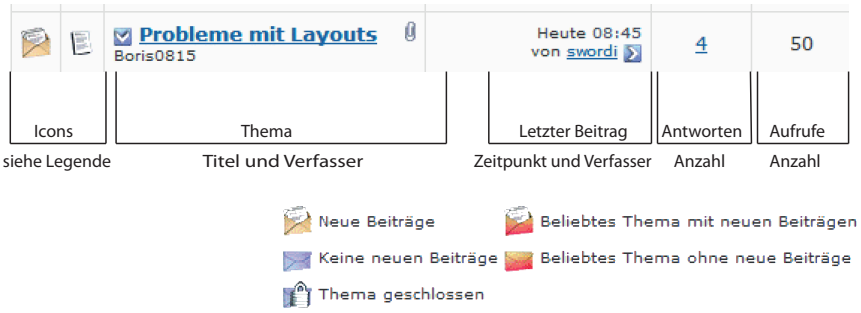

Abbildung 4.3: Auflistung eines Beitrags.

Kategorie werden die verfügbaren Themen in einer Liste dargestellt, wie in Abb. 4.2 ersichtlich wird. In der Übersicht werden verschiedene Informationen zum jeweiligen Thema gegeben. An erster Stelle wird mittels Symbolen visualisiert, ob neue Beiträge zu einem Eintrag vorhanden sind oder ob es sich um ein beliebtes Thema mit vielen Besuchern handelt. Eine Aufstellung mit allen möglichen Status der Icons ist in Abb. 4.3 gegeben. Dem Statusindikator folgen der Name des Themas und der Benutzername des Verfassers. Dem schließen sich Informationen darüber an, wann und von wem der letzte Beitrag geschrieben wurde, sowie die Gesamtmenge der gegebenen Antworten und die Zahl der Aufrufe. Die beschriebenen Fakten werden in Abb. 4.3 anhand einer exemplarischen Kurzübersicht illustriert. Nach Auswahl eines

Abbildung 4.4: Exemplarische Darstellung einer Antwort.

Themas werden einerseits die Antworten andererseits weitere Informationen zum Verfasser des Beitrags dargestellt. Über den Benutzer werden folgende Angaben gemacht:

- Benutzerkategorie
- Registrierungsdatum
- Anzahl der Beiträge
- Anzahl der Anerkennungen für Antworten
- Angaben zur Reputation des Benutzers

Die Benutzerkategorie wird anhand der Anzahl der Antworten des jeweiligen Benutzers berechnet. Die Berechnungsgrundlage für das Board-System Android-Hilfe.de kann in Tab. 4.1 nachvollzogen werden. Da es sich beim Forschungsgegenstand um ein technisches Board-System handelt, kann es dazu kommen, dass Quellkode in den Antworten enthalten ist. Dieser wird durch eine besondere Formatierung hervorgehoben. Weiter wird die Option angeboten mittels eines Zitats einer Antwort direkt auf deren Inhalt zu referenzieren. In Abb. 4.4 wird die Darstellungsform einer Antwort ersichtlich. Die beschriebenen Eigenschaften des Board-Systems Android-Hilfe.de lassen sich, bis auf die Benutzerkategorien, auf andere Boards übertragen. Dadurch lassen sich die anhand des Forschungsgegenstands gewonnenen Erkenntnisse als eine allgemeingültige Definition der Darstellung von Boards ansehen [43].

Im Rahmen der vorangegangenen Ausführungen wurde die Unterscheidung zwischen den Termini Forum und Board-System deutlich. Weiter folgte die Einordnung der zu untersuchenden Diskussionsplattform in die Kategorie der Boards. Darauf aufbauend ist eine detaillierte Beschreibung des Aufbaus des ausgewählten Board-Systems Android-Hilfe.de vorgenommen worden. Im Zuge dessen sind die verfügbaren Daten zu den einzelnen Themen und Benutzern aufgezeigt worden, was für die spätere Verarbeitung in der entwickelten Software von Bedeutung sein wird. Nachfolgend soll eine kurze Analyse der Antworten hinsichtlich des Sprachstils und der enthaltenen Semantik durchgeführt werden.

Tabelle 4.1: Berechnungsgrundlage Benutzerkategorien.

Benutzerkategorie	*Antwortenanzahl*
Gast	0
Neuer Benutzer	0 - 30
Juniormitglied	30 - 50
Android-Hilfe Mitglied	50 - 200
Erfahrener Benutzer	200 - 300
Fortgeschrittenes Mitglied	300 - 500
Android Experte	500 - 1000
Android Guru	1000 - 3000

4.3 Systemarchitektur

4.3.1 Überblick

Die entwickelte Software BoardToOntology lässt sich in die Teilbereiche Datenerfassung, Datenhaltung und Datenverarbeitung aufgliedern. In Abb. 4.5 wird ein Überblick über die einzelnen Komponenten und deren Zusammenspiel untereinander gegeben. In der Datenerfassung werden alle verfügbaren Daten des Forschungsgegenstands Android-Hilfe.de erfasst. Dabei werden die textuellen Daten, sowie alle Zusatzinformationen, wie Benutzerdaten, in aufbereiteter Form im implementierten Datenmodell gespeichert. Dieses dient als zentrale Datenschnittstelle zwischen der Datenhaltung und den restlichen Komponenten des Programms. Es wurde nach dem Composite Pattern entwickelt, wodurch die vorhergehend beschriebene Struktur des Board-Systems exakt übernommen werden konnte. Die gewonnenen Daten werden ins XML-Format überführt und über einen XML-Parser wieder verfügbar gemacht.

Im nächsten Schritt fordert die Ontology Mapper Komponente im Zuge der Datenverarbeitung alle verfügbaren Daten an. Innerhalb dieses Programmteils werden die Benutzermethoden und die unbearbeiteten Textdaten in der Ontologie abgespeichert. Dafür werden zunächst alle benötigten Klassen, Rollen und Restriktionen definiert und angelegt. Anschließend folgt die Erstellung der Individuen innerhalb der zugehörigen Klassen. Außerdem werden die Zuordnung der jeweiligen Werte mittels konkreter Rollen und die Verknüpfung der Klassen untereinander mit Hilfe der abstrakten Rollen vorgenommen.

Abschließend werden die textuellen Daten im Natural Language Processing Modul aufbereitet und verarbeitet. Dafür werden innerhalb der Board-Beiträge die Satzgrenzen definiert und anschließend eine Syntaxanalyse vorgenommen. Auf Basis der gewonnenen Satzteile werden die Wortarten be-

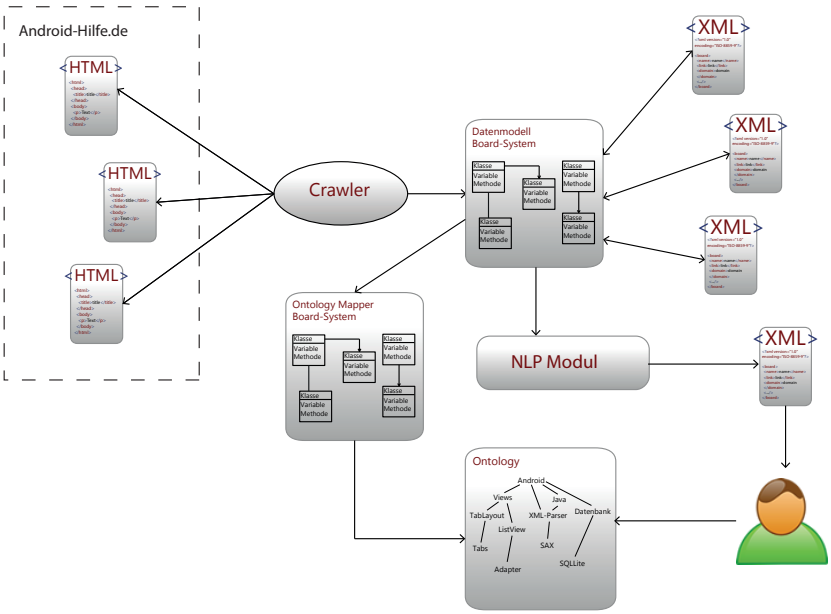

Abbildung 4.5: Überblick: BoardToOntology.

stimmt und dabei die bedeutungstragenden Einheiten extrahiert. Anschließend wird die Auftretenshäufigkeit der entnommenen Terme bestimmt. Darauf aufbauend wird eine Termgewichtung vorgenommen und alle in Frage kommenden Wörter in ein annotiertes XML-Dokument gespeichert. Diese Terme sind mögliche Kandidaten für den Aufbau einer Taxonomie und für Schlagwörter zu den einzelnen Beiträgen. Durch die vorangegangenen Ausführungen konnte ein Überblick über die entwickelte Software gewonnen werden. Im Folgenden sollen die Aufgaben und die Funktionsweise der einzelnen Komponenten erläutert werden.

4.3.2 Datenerfassung

Für die Erfassung der Daten des Forschungsgegenstands Adroid-Hilfe.de wurde ein multithreadfähiger Internet-Crawler implementiert. Dieser extrahiert die HTML-Struktur von Webseiten ausgehend von einer Auflistung von Weblinks. Im Rahmen der entwickelten Methodik wurde dementsprechend eine Möglichkeit gesucht, eine geordnete Liste aller Themen der Rubrik App-Entwicklung zu erhalten. Für diese Anforderung bietet das untersuchte Board-System eine Sitemap. In dieser werden jeweils 250 Einträge pro Seite inklusive der permanenten Verlinkung nacheinander aufgeführt. Die einzelnen Links zu den Seiten können aus der Navigation, die in Abb. 4.6 ersichtlich

Tabelle 4.2: Extraktionsmuster für benötigte Daten.

HTML-Tag	*Art*	*Kriterium*	*Inhalt*
table	ID	post + Beitrags-ID	Datensatzbeitrag
div	Klasse	bigusername	Benutzername
a	Klasse	bigusername	Link zum Benutzer
div	Text	Beiträge:	Anzahl der Beiträge
div	Text	Registriert seit	Registrierungsdatum
div	Text	Erhielt	Benutzerreputation
div	ID	post + Nachrichts-ID	Text eines Beitrags

ist, extrahiert werden. Auf Basis dieser Liste mit Seiten, in denen die Verlinkung zu allen verfügbaren Themen im Board-System enthalten ist, kann der Crawl-Vorgang gestartet werden. Der Crawler extrahiert nach einander die Links zu den jeweiligen Themen und speichert die Rohdaten und die für die Updatefunktion benötigten Metainformationen ab.

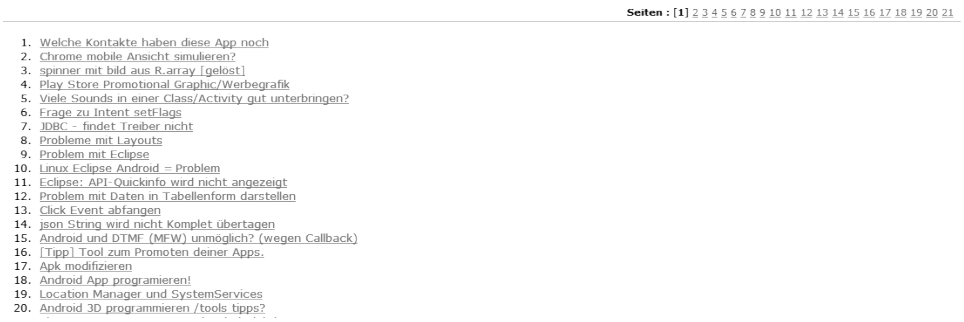

Abbildung 4.6: Android-Hilfe.de Sitemap.

Die gewonnen Rohdaten werden mit Hilfe eines HTML-Parsers aufbereitet. Obwohl die Struktur und die Inhalte der Boards einem gleichem Schema folgen, unterscheiden sie sich in der Benennung der HTML-Tags und der Struktur der Dokumente. Auf Grund dessen konnte nur eine an den Forschungsgegenstand angepasste Analyse der HTML-Inhalte vorgenommen werden. Die verfügbaren Daten wurden anhand von HTML-Tags, CSS-Klassen-, und ID-Namen, sowie mittels des enthaltenen Texts extrahiert. Die angewandten Kriterien, um die benötigten Daten zu erhalten, können durch Tab. 4.2 nachvollzogen werden. Zusätzlich zu der Datenextraktion erfolgt eine Untersuchung der Themenbeiträge auf enthaltene Zitate und Quellkode. Die Zitate müssen entfernt werden, damit bei der späteren Bestimmung der Termfrequenz keine Verfälschungen durch redundante Textpassagen entstehen. Im Zuge der Textanalyse wird eine Syntaxanalyse der einzelnen Texte

vorgenommen. Diese kann nur für natürlich sprachliche Daten vorgenommen werden, woraufhin der Quellcode aus den Beiträgen entnommen und als eigenes Datum definiert wird.

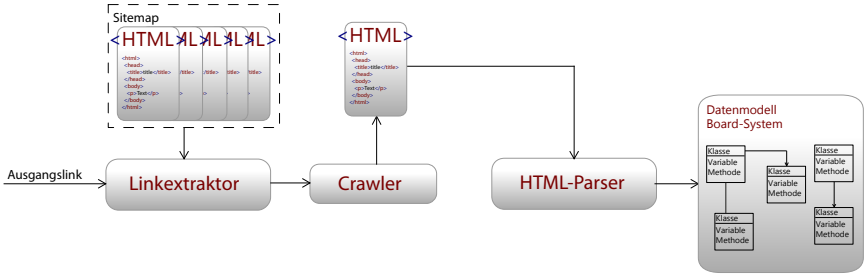

Abbildung 4.7: Datenerfassung von Board-System Android-Hilfe.de.

Die extrahierten Daten werden nach Abschluss der Analyse in das Datenmodell gespeichert, von dem ausgehend die weitere Verarbeitung stattfindet. Der in Abb. 4.7 ersichtlichen Grafik kann der Gesamtaufbau des Moduls zur Datenerfassung entnommen werden. Im nächsten Abschnitt soll auf die Struktur des XML-Dokuments eingegangen und der Aufbau des Datenmodels beschrieben werden.

4.3.3 Datenhaltung

Für die temporäre Datenhaltung zur Laufzeit wurde ein Datenmodell nach dem Composite Pattern implementiert. Um die Daten persistent zu speichern, wurde eine XML-Struktur definiert in der die Daten gespeichert werden. In Abb. 4.8 kann der Aufbau der Komponente zur Datenhaltung nachvollzogen werden. Die weiteren Ausführungen beziehen sich zunächst auf die Beschreibung des Composite Patterns. Daraufhin folgen Erläuterungen zum Aufbau des XML-Dokuments und dessen Verarbeitung.

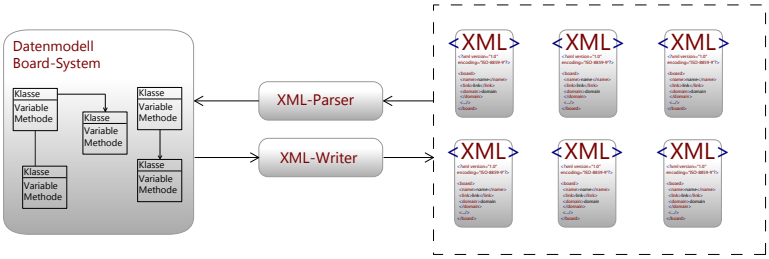

Abbildung 4.8: Aufbau Datenhaltung.

Das Composite Pattern eignet sich vor allem dazu, eine Hierarchie von Objekten aufzubauen. Dabei können alle enthaltenen Elemente über das Wurzelelement bezogen werden. Der Aufbau und die Funktionsweise dieses Design Patterns wird in nachfolgendem Zitat entnommen von [45] treffend zusammengefasst:

> Es wird eine gemeinsame Schnittstelle für die Elementbehälter (Composite, Kompositum; Aggregat, Knoten) und für die atomaren Elemente (Leaf, Blatt) definiert: Component. Diese Schnittstelle Component definiert die Methoden, die gleichermaßen auf Composites und auf Leafs angewandt werden sollen. Composites delegieren oft Aufrufe (operate()) an ihre Components, die atomare Leafs oder wiederum zusammengesetzte Composites sein können.

Dieses Design Pattern für das Datenmodell wurde unter anderem auf Grund der Gemeinsamkeiten hinsichtlich der Strukturierung mit dem Forschungsgegenstand gewählt.

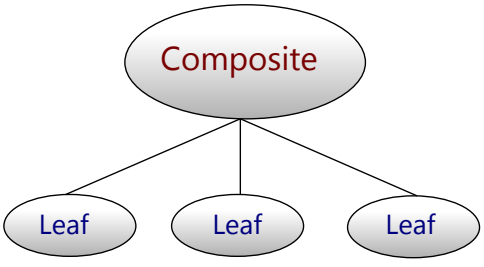

Abbildung 4.9: Grafische Darstellung Composite Pattern.

Wie aus den vorangegangenen Ausführungen bezüglich des Aufbaus von Diskussionsplattformen deutlich wurde, ist die gewählte Rubrik

App-Entwicklung des Forschungssubjekts in Themen und Antworten bzw. Beiträgen zu diesen gegliedert. Dieser Aufbau lässt sich mittels des Composite Patterns exakt abbilden. Das Kompositum beschreibt dabei das Thema und den Text des ersten Eintrags. Diesem Thema sind in hierarchischer Anordnung alle Antworten dazu untergeordnet. Diese werden somit dem Knoten als Blätter hinzugefügt. In Abb. 4.9 wird exemplarisch eine Anordnung von diesem Composite und dessen Leafs visualisiert. Es wird deutlich, dass alle Antworten (Leafs) auf einer Ebene sind und in Listenform dem Thema (Composite) zugeordnet sind, was der Struktur eines Board-Systems entspricht.

```
<?xml version="'1.0"' encoding="'ISO-8859-9"'?>
<board>
  <name>String</name>
  <link>String</link>
  <domain>String</domain>
  <boardTopics>
    <boardTopic>
      <topicTitle>String</topicTitle>
      <numberOfEntries>Integer</numberOfEntries>
      <lastEntryDate>Date</lastEntryDate>
      <entryLink>String</entryLink>
      <new>Boolean</new>
        <boardPost>
          <firstEntry>Boolean</firstEntry>
          <postDate>DateTime</postDate>
          <userName>String</userName>
          <userRegistryDate>Date</userRegistryDate>
          <userEntries>Integer</userEntries>
          <userReputation>String</userReputation>
          <userLink>String</userLink>
          <postMessage>String</postMessage>
          <sourceCode String </sourceCode>
          <hasSourceCode>SBoolean</hasSourceCode>
        </boardPost>
      </boardTopic>
  </boardTopics>
</board>
```

Anhand den zur Laufzeit gehaltenen Daten werden über den XML-Writer die XML-Dokumente als persistenter Speicher erstellt. Sind die Boarddaten in einem anderen Teil der Software vonnöten, wird mittels des XML-Parsers das jeweilige XML-Dokument ausgelesen und im Datenmodell abgelegt. Im XML erfolgt eine Speicherung alle verfügbaren Daten innerhalb der in vorhergehend ersichtlicher Struktur. Nach Beschreibung der Schnittstelle zum Auslesen aller verfügbaren Daten sollen die weiteren Ausführungen die angewandte Methodik hinsichtlich der Datenanalyse und -verarbeitung thematisieren. Dabei wird das Ontology Mapping der semi strukturierten Daten im XML-Format erläutert. Darauffolgend soll die Natural Language Processing Komponente und die Vorgehensweise bezüglich der Extraktion von möglichen Termen für die Verschlagwortung und den Aufbau einer Taxonomie

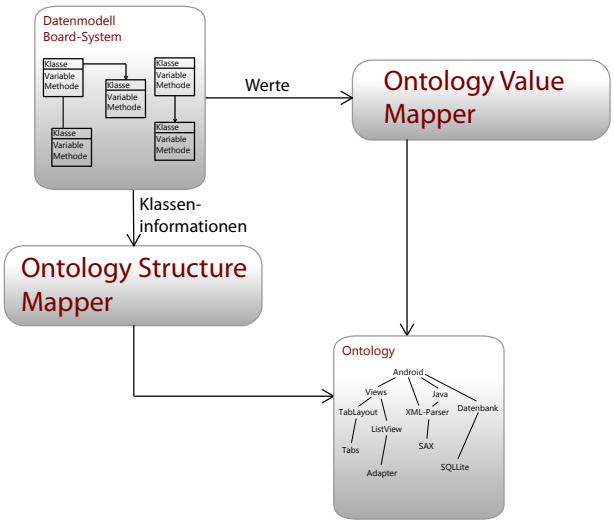

Abbildung 4.10: Ontology Mapper: Aufbau einer Ontologie anhand der Boarddaten.

dargelegt werden.

4.3.4 Datenverarbeitung

Die Komponente zur Datenverarbeitung umfasst zwei Teilgebiete: das Übertragen der Boarddaten aus dem XML in die Ontologie und die Extraktion von in Frage kommender Termen für den Aufbau einer Taxonomie bzw. für die Verschlagwortung. Im Folgenden soll im ersten Schritt die Vorgehensweise zur Anreicherung der Ontologie mit den gesammelten Werten beschrieben werden. Anschließend folgen Erläuterungen zu der angewandten Methodik zur Bestimmung relevanter Terme.

Ontology Mapper

In Abb. 4.10 wird ein Überblick über die grundlegende Funktionsweise der Komponente Ontology Mapper illustriert. Im ersten Schritt werden die benötigten Klassen, Rollen und Einschränkungen in der Ontologie angelegt. Dabei kann wiederum die angelegte Klassenstruktur im Datenmodell genutzt werden. Für die Übertragung der Daten in die Ontologie werden drei Klassen, wie in Abb. 4.11 ersichtlich ist, erstellt. Im Datenmodell bestehen äquivalente Klassen, die jeweils die Werte für die gesammelten Boardaten enthalten. Unter Verwendung des Konzepts Java Reflections ist es in der Folge möglich, anhand der spezifizierten Datentypen die abstrakten und konkreten Rollen zu erstellen.

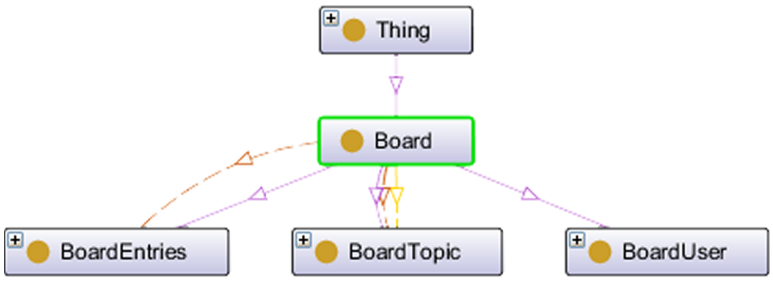

Abbildung 4.11: Aufbau Ontologie für die Daten des Board-Systems.

Konkrete Rollen ermöglichen es, Individuen einer Klasse in der Ontologie mit Werten zu verknüpfen. Dafür werden in der Programmiersprache Java primitive Datentypen, die Klasse *String* und *Calendar* verwendet. Auf Grund dessen werden mittels dieser Datentypen die datatype properties erstellt. Dabei wird als Domäne Klasse der Variable angegeben und als Range der Datentyp. Abstrakte Rollen verbinden in einer Ontologie zwei Klassen untereinander. Diese Beziehung untereinander kann mit einer Objektinstanz, die von einer Klasse in Java gehalten werden, verglichen werden. Unter Berücksichtigung dessen ist es möglich object properties anhand dieser Instanzen zu erstellen. Der Domain wird hierbei die Klasse, die eine Instanz der anderen Klasse hält, zugewiesen, der Range die Klasse der Instanz. In Tab. 4.3 wird für die Datentypen, die im Datenmodell vorhanden sind, die entsprechende konkrete Rolle mit domain und range angegeben, in Tab. 4.4 dementsprechend die abstrakten Rollen.

Tabelle 4.3: Umwandlung von Java-Feldern zu konkreten Rollen in der Ontologie.

Datentyp	*Feldname*	*Rollenname*	*Domain*	*Range*
String	mName	hasName	Board	String
int	mDomain	hasDomain	Board	String
String	mName	hasName	Board	String
Calendar	mLastEntryDate	hasLastEntryDate	BoardTopic	dateTime
Boolean	mHasSourceCode	hasSourceCode	BoardTopic	boolean

Tabelle 4.4: Umwandlung von Java-Feldern zu abstrakten Rollen in der Ontologie.

Datentyp	*Feldname*	*Rollenname*	*Domain*	*Range*
ArrayList<ABoardEntry>	mBoardEntries	hasBoardEntries	Board	BoardEntries
ArrayList<BoardTopic>	mBoardTopics	hasBoardTopics	Board	BoardTopic
BoardUser	mBoardUser	hasBoardUser	BoardEntries	BoardUser

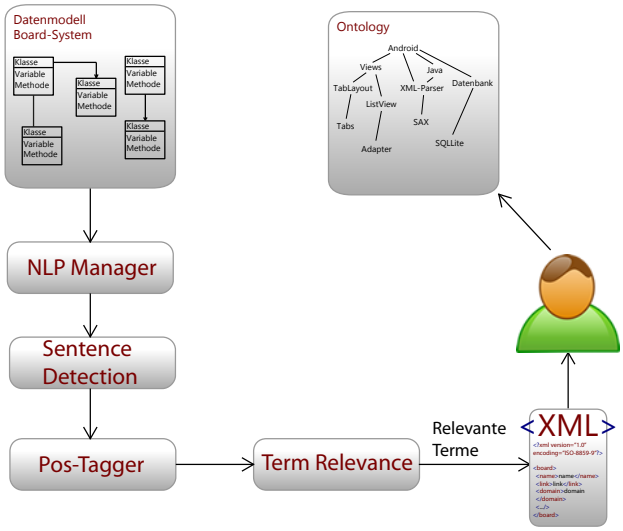

Abbildung 4.12: Aufbau Ontologie für die Daten des Board-Systems.

Nachdem die Struktur in der Ontologie aufgebaut wurde, werden die verfügbaren Daten über den Forschungsgegenstand Android-Hilfe.de als Individuen in den jeweiligen Klassen angelegt bzw. durch konkrete und abstrakte Rollen untereinander verbunden.

Bestimmung relevanter Terme

Diese Komponente des entwickelten Systems BoardToOntology ermittelt mit Methoden aus den Bereichen Natural Language Processing und Information Retrieval relevante Terme. Diese werden in einem annotierten XML-Dokument abgespeichert. Infolgedessen kann ein Experte auf Basis dieser textuellen Einheiten eine Taxonomie erstellen. Des Weiteren erfolgt eine Zuordnung der relevantesten Termen als Schlagwörter für die jeweiligen Beiträge des Board-Systems in der Ontologie. Ein Überblick über die zentralen Module dieses Programmteils kann durch die Abb. 4.12 gewonnen werden. Im ersten Schritt werden die Textdaten in den Beiträgen hinsichtlich der Satzgrenzen analysiert. Dadurch findet eine Gliederung des Textes in seine Sätze statt, die im Folgenden in einer Syntaxanalyse verarbeitet werden. Bei dieser Untersuchung der Satzstruktur werden die Satzglieder bestimmt. Diese werden weiter hinsichtlich der Verwendung im Satz spezifiziert. Im Zuge dieser Syntaxanalyse werden zusätzlich die Wortarten bestimmt und mit Tags des Stuttgart-Tübingen-Tagsets versehen. Ein Ergebnis einer exemplarischen Syntaxanalyse, dargestellt in einer Baumstruktur, kann in nachvollgendem Beispiel nachvollzogen werden.

```
1 <ROOT>
```

```
  <S>
    <NP>
      <NN>
        <Funktioniert/>
      </NN>
      <MPN>
        <NE>
          <Android/>
        </NE>
        <NE>
          <Referenz/>
        </NE>
      </MPN>
    </NP>
    <VP>
      <VVPP>
        <beschrieben/>
      </VVPP>
    </VP>
    <\$.>
      <./>
    </\$.>
  </S>
</ROOT>
```

Für die weiteren Untersuchungen hinsichtlich der relevantesten Terme für das jeweilige Thema, sind die Substantive von Bedeutung. Diese können in drei verschiedene Kategorien eingeteilt werden:

- Substantive
- Eigennamen
- Fremdwörter

In Boards mit einer technischen Domäne stehen Eigennamen und Fremdwörter häufig für Fachbegriffe aus dem Englischen oder des jeweiligen Diskurses. Dieser Fakt wird im Zuge der Gewichtung der Terme berücksichtigt. Die Term-Gewichtung in Kombination mit der Auftretens-Häufigkeit der Terme führt letztendlich zu der Auswahl der bedeutungstragendsten lexikalischen Einheiten. Im Rahmen der Bestimmung der relevantesten Terme wird auf die tf-idf Formel, ersichtlich in 4.1, aus dem Bereich Information Retrieval zurückgegriffen.

$$\text{Termgewichtung} = \frac{n_t}{N} \cdot \log \frac{T}{d} \tag{4.1}$$

n_t: Anzahl des Terms im Text
N: Gesamtanzahl der Wörter des Textes
T: Anzahl der Texte im Korpus
d: Anzahl der Texte in denen Term vorkommt

Die Formel setzt sich aus der Multiplikation der Termfrequenz (tf) und der inversen Dokumentfrequenz (idf) zusammen. Aus der inversen Dokumentfrequenz lässt sich schließen, dass „je seltener ein Begriff in den Dokumenten

vorkommt, desto höher ist sein idf Wert“ [46]. Als Korpus wird in diesem Fall das Ergebnis einer Seite der Sitemap angesehen und somit 250 Themen. Wie vorangehend bereits erwähnt, haben bestimmte Kategorien von Substantiven eine höhere Relevanz als andere. Zudem ist durch die Struktur eines Board-Systems vorgegeben, dass Terme, die im Thementitel vorkommen, mit hoher Wahrscheinlichkeit eine sehr hohe Relevanz für die Beschreibung der enthaltenen Beiträge haben. Deswegen werden diese Lemmata mit einem Faktor multipliziert.

Nach Abschluss der Analyse hinsichtlich der Relevanz werden die Terme mit deren syntaktischen Kategorie und der Termgewichtung sortiert in ein XML geschrieben. Die Struktur dieses XML-Dokuments wird in nachfolgendem XML-Kode dargestellt. Weiter werden die fünf wichtigsten Lemmata als Schlagwörter für die jeweiligen Themen in die Ontologie aufgenommen.

```
<?xml version="'1.0"' encoding="'ISO-8859-9"'?>
<candidatTerms>
    <boardTopic>
      <name>String</name>
      <title>String</title>
      <namedEntities>
         <term>String</term>
      </namedEntities>
      <normalNoun>
         <term>String</term>
      </normalNoun>
      <foreignLanguage>
         <term>String</term>
      </foreignLanguage>
    </boardTopic>
</board>
```

Aus den vorangegangen Ausführungen wurde deutlich, wie das System im Ganzen aufgebaut ist. Weiter wurden die einzelnen Komponenten, insbesondere hinsichtlich der Funktionsweise und des Bezugs auf den Forschungsgegenstand, erläutert. Nachfolgend soll auf die konkrete Implementierung der einzelnen Module von BoardToOntology und den im Zuge dessen verwendeten Technologien eingegangen werden.

Kapitel 5

Implementierung

Im Rahmen der vorangegangenen Ausführungen wurde die Arbeit in das Forschungsgebiet Ontology Learning eingeordnet. Nach Definition und Erläuterung der wichtigsten Konzepte aus diesem Bereich folgte ein Überblick über die bereits entwickelten Systeme und Ansätze zur semi- bzw. automatischen Erstellung einer Ontologie. Darauf aufbauend folgten eine Beschreibung der angewandten Methodik und eine nähere Bestimmung des Forschungsgegenstands. In diesem Kapitel soll nun auf die konkrete Implementierung des entwickelten Systems eingegangen werden. Dabei stehen der Programmablauf, sowie die verwendeten Frameworks und Bibliotheken im Fokus der Ausführungen. Für die Entwicklung kamen die Technologien Java, HTML, XML und OWL zum Einsatz.

5.1 Datenerfassung und Datenhaltung

5.1.1 Internet-Crawler – Erfassen der Boarddaten

Die Verarbeitung der Meta, und Forendaten erfordert die persistente Speicherung der Daten in einem maschinenlesbaren Format. Der Zugriff auf die Rohdaten des Forschungsgegenstands über die Datenbank des Boardsystems war nicht möglich. Deswegen musste ein Crawler implementiert werden, um alle verfügbaren Daten, strukturiert in einem HTML-Dokument, erhalten zu können. Im Java-Umfeld existieren einige Frameworks, die die benötigten Funktionalitäten enthalten. Auf Grund der großen Anzahl von Daten wurde jedoch eine Bibliothek gesucht, mit der es möglich ist, parallel Unterseiten einer Webseite zu analysieren und zu empfangen. Mit dem Internetcrawler crawler4j, entwickelt von Yasser Ganjisaffar, nutzt Jave Threads und ist deswegen für diese Anforderung geeignet. Außerdem wird, wie im nachfolgenden Zitat nachvollzogen werden kann, eine simple Implementierung suggeriert, was zudem Auswahlkriterium war.

> Crawler4j is an open source Java crawler which provides a simple

> interface for crawling the Web. You can setup a multi-threaded web crawler in 5 minutes! [47]

Die Datenerfassung nimmt eine wichtige Rolle für die Funktionalität der im Zuge dieser Arbeit entwickelten Software ein, weswegen die Funktionsweise des verwendeten Crawlers nachfolgend in den Fokus der Ausführungen gestellt werden soll.

Die Nutzung des Internetcrawlers crawler4j benötigt im ersten Schritt eine Konfiguration. In dieser wird angegeben, wie viele Threads maximal genutzt werden sollen, sowie die Crawltiefe und die Verzögerung zwischen den einzelnen Crawlvorgängen.

> Crawl Depth represents the minimum number of times a website visitor must click on a Link from the „root“ Webpage in order to get to a particular Webpage. [48]

Wie aus den vorangegangenen Ausführungen deutlich wurde, wird für die Datenerfassung die Sitemap des Board-Systems verwendet. Dadurch erhält man ein Webdokument, in dem die Themen mit den dazugehörigen Links, aufgelistet sind. Aufgrund dessen ist die Crawltiefe auf eins gesetzt worden, da wie in oben genanntem Zitat deutlich wird, die Beiträge mit einem Klick erreicht werden können. Die Verzögerung bei den Crawlvorgängen wird benötigt, um eine Serverüberlastung bei der Datenquelle zu vermeiden. Nach Abschluss der Konfiguration wird im Zuge des implementierten Systems die Unterseiten der Sitemap erfasst und die erste dem Crawler übergeben. Dieser startet nachfolgend die angegebene Anzahl an Threads und beginnt die Links zu extrahieren. Die konkrete Implementierung kann an folgendem Kodeauszug nachvollzogen werden.

```
1  CrawlConfig config = new CrawlConfig();
2  config.setCrawlStorageFolder(rootFolder);
3  config.setMaxDepthOfCrawling(1);
4  config.setPolitenessDelay(200);
5  PageFetcher pageFetcher = new PageFetcher(config);
6  RobotstxtConfig robotstxtConfig = new RobotstxtConfig();
7  RobotstxtServer robotstxtServer = new RobotstxtServer(robotstxtConfig,
       pageFetcher);
8  CrawlController controller = new CrawlController(config, pageFetcher,
       robotstxtServer);
9  controller.addSeed(seed);
10 controller.start(LocalDataCollectorCrawler.class, NUMBER_OF_CRAWLERS);
```

Nach Extraktion der Links stellt der implementierte Crawler die zwei Methoden *shouldVisit(WebURL url)* und *visit(Page page)* zur Verfügung. Die erste Methode liefert zurück, ob ein bestimmter Link berücksichtigt werden soll. Ist dies der Fall, wird die Methode *visit(Page page)* aufgerufen.

```
1 Pattern filters = Pattern.compile(".*(\\.(css|js|bmp|gif|jpe?g" + "|png|
      tiff?|mid|mp2|mp3|mp4" + "|wav|avi|mov|mpeg|ram|m4v|pdf" + "|rm|smil
      |wmv|swf|wma|zip|rar|gz))\$");
```

```
2
3 @Override
4 public boolean shouldVisit(WebURL url) {
5   String href = url.getURL().toLowerCase();
6   return (!filters.matcher(href).matches() && href.startsWith("http://
      www.android-hilfe.de/android-app-entwicklung/") && href.endsWith(".
      html"));
7 }
8
9 @Override
10 public void visit(Page page) {
11   System.out.println("Visited: " + page.getWebURL().getURL());
12   if(page.getParseData() instanceof HtmlParseData) {
13     if(page.getWebURL().getURL().startsWith("http://www.android-hilfe.de
      /android-app-entwicklung/")) {
14       HtmlParseData htmlParseData = (HtmlParseData) page.getParseData();
15       synchronized (this) {
16         setForumRawData(htmlParseData, page.getWebURL().getURL());
17       }
18     }
19   }
20 }
```

Da es sich bei den Beiträgen des Forschungsgegenstands nur um HTML-Dokumente mit einer bestimmten Domäne handelt, werden alle anderen Dateiendungen ausgefiltert. In der Methode *visit(Page page)* erfolgt die Extraktion der HTML-Daten und Informationen bezüglich des Datums des letzten Eintrags und der Anzahl der Einträge eines Themas, die für die Update-Funktion benötigt werden. Diese Daten werden in einer Liste gespeichert und nach Abschluss des Crawlvorgangs über die bereitgestellte Methode *getMyLocalData()* zurückgegeben. Die gecrawlten Daten werden nachfolgend an den HTML-Parser übergeben und die aufbereiteten Daten in ein XML geschrieben. Der beschriebene Vorgang wird für alle Seiten der Sitemap wiederholt, bis alle verfügbaren Daten des Board-Systems erfasst wurden. Eine Gesamtübersicht über die Funktionsweise des implementierten Crawlers kann in Abb. 5.1 nachvollzogen werden.

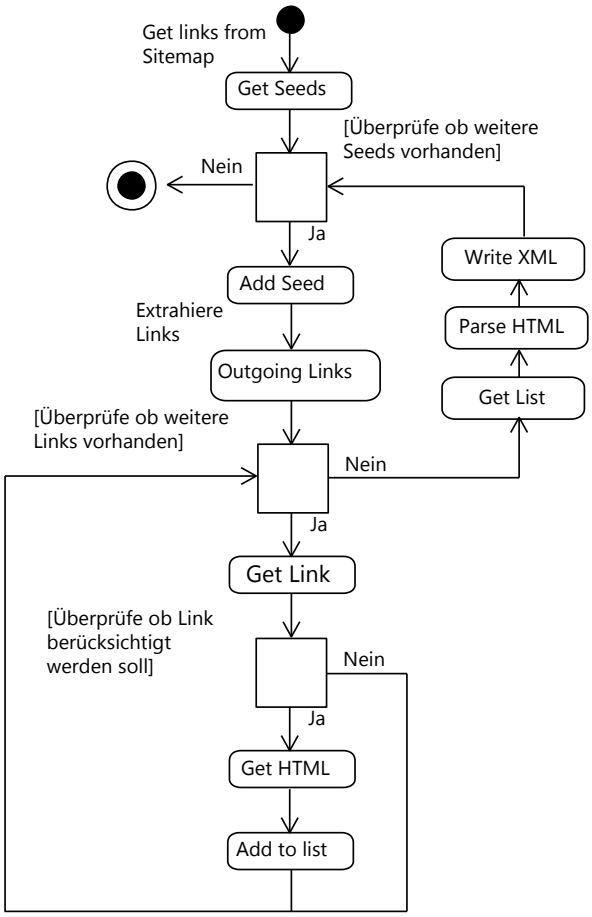

Abbildung 5.1: Aktivitätsdiagramm: Internet-Crawler.

Mit dem implementierten Crawler wird auch eine Möglichkeit mitgeliefert, HTML zu parsen und einen Reintext aus diesem zu gewinnen. Da im Zuge der Entwicklung jedoch deutlich wurde, dass neben dem Text der Beiträge noch weitere Daten benötigt werden, wurde unter Verwendung der Bibliothek jsoup ein HTML-Parser integriert. Eine Beschreibung der konrekten Umsetzung des HTML-Parsers soll im nächsten Abschnitt vorgenommen werden.

5.1.2 Analyse der Rohdaten

> jsoup is a Java library for working with real-world HTML. It provides a very convenient API for extracting and manipulating data, using the best of DOM, CSS, and jquery-like methods. [49]

Für die Extraktion der wichtigsten Daten aus den gewonnen Rohdaten im HTML-Format wurde die Bibliothek jsoup verwendet. Wie im einleitenden Zitat angemerkt, ist es möglich, mit jsoup anhand von CSS-Selektoren auf HTML-Elemente zuzugreifen. Außerdem ist es möglich Regular Expressions zu definieren, um Elemente aus dem DOM-Baum zu erhalten. Nachfolgend soll erläutert werden, wie die in Abschnitt 4.1 beschriebenen Daten mittels Methoden aus der Bibliothek jsoup extrahiert werden.

Um mit jsoup arbeiten zu können, müssen im ersten Schritt die Daten im HTML-Format in ein Dokument-Objekt eingelesen werden. Danach kann mittels einer jquery-ähnlichen Syntax auf den DOM-Baum zugegriffen und die Daten ausgelesen werden. Der Titel des Themas ist bei dem vorliegenden Board-System jeweils im Titel-Tag hinterlegt und kann über die Methode *getElementsByTag()* bezogen werden. Es muss ledlich der Boardname ersetzt werden, da dies im Titel selber nicht relevant ist und zu Verfälschungen bei der syntaktischen Analyse führen würde.

```
1 String topicTitle = doc.getElementsByTag("title").text();
2 topicTitle = topicTitle.replaceAll(" - Android-Hilfe.de", "");
```

Weiter sind die einzelnen Beiträge zu einem Thema immer durch eine HTML-Tabelle strukturiert, die durch einen CSS-ID identifizierbar ist. Diese setzt sich aus dem Präfix *post* und einer generierten Zahlenfolge zusammen. Da sich die Zahl bei den jeweiligen Einträgen immer ändert, wurde dafür eine Regular Expression verwendet.

```
1 Elements postMessages = doc.select("table[id~=post[0-9]+]");
```

Dadurch erhält man ein Array mit allen in dieser Tabelle enthaltenen Elementen, welches daraufhin durchiteriert wird. Für die weitere sprachliche Analyse ist es notwendig, dass Zitate und Quellkode aus den Beiträgen entfernt werden. Dafür wird der Text aus dem jeweiligen HTML-Tag durch einen leeren String ersetzt. Für Inhalte, die nicht durch CSS-Klassifikationen identifizierbar sind, musste auf enthaltene Inhalte in den Tags zurückgegriffen werden. Sollte eine Zuordnung auf Basis dieser Inhalte nicht funktionieren, wurden die Daten mittels Regular Expressions ausgelesen. Folgendes Kodebeispiel beschreibt die Methoden von jsoup, die für diese Extraktionsform bereitgestellt werden.

```
1 boardUser.setReputation(docPostMessage.select(":containsOwn(Erhielt)").
      text());
2 String postDate = docPostMessage.select("td:matchesOwn
      (([123]0|[012][1-9]|31)(\\.|-|/|,)(0[1-9]|1[012])(\\.|-|/|,)
      (19[0-9]{2}|2[0-9]{3})(, )([0-9]+)(:)([0-9])+)").text();
```

Die gewonnenen Daten werden im implementierten Datenmodell gespeichert, bevor sie zu einem XML-Dokument weiter verarbeitet werden. Die Verarbeitung der XML-Daten (Schreiben und Lesen), sowie das implementierte Datenmodell soll im Folgenden kurz erläutert werden.

5.1.3 Datenmodell und XML-Verarbeitung

Das Datenmodell in Java wurde mit dem Composite Design Pattern umgesetzt. Wie aus den vorangegangen Ausführungen hervorging, kann mit Hilfe dieser Datenstruktur der Aufbau des Forschungsgegenstands Android-Hilfe.de abgebildet werden. Die Implementierung des Musters wird in Abb. 5.2 mittels eines Klassendiagramms visualisiert.

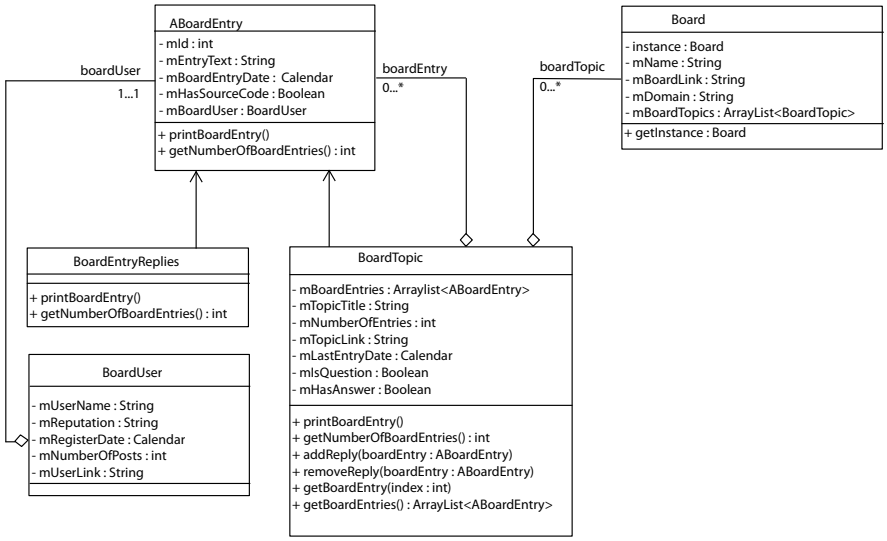

Abbildung 5.2: Implementierung Composite Design Pattern.

Es wird deutlich, dass die gewonnenen Daten komplett in diesem Modell gespeichert werden. Dabei können alle Antworten über das jeweilige Thema bezogen werden. Auf Grund der Strukturierung der Sitemap werden bis zu 250 Themen zur Laufzeit in diesem Datenmodell gehalten. Um diese Daten von jedem Programmteil zugänglich zu machen, ohne das jeweilige Objekt übergeben zu müssen, wurde ein Singleton implementiert. In diesem werden die allgemeinen Daten zum Board-System, wie der Name, die Domäne und die Web-Adresse, gespeichert. Dazu wird eine Liste mit allen zur Verfügung stehenden Themen bereitgestellt. Über diesen Singleton beziehen die Ontology Mapping Komponente und das Natural Language Modul die Daten des Boards. Um die Daten nicht permanent neu vom Board-System beziehen zu müssen, wurde eine XML-Schnittstelle definiert. Die Struktur wurde im vorherigen Kapitel ausführlich beschrieben. Die Implementierung erfolgte mit dem in Java integrierten DOM-XML-Parser, weswegen auf eine ausführliche Beschreibung an dieser Stelle verzichtet und auf den beigelegten Kode verwiesen wird. Die Speicherung der Daten im XML-Format bringt jedoch mit sich, dass die Möglichkeit bereitgestellt werden musste, den Datenbe-

stand mit aktuelleren Daten zu erneuern. Auf die Implementierung dieser Funktionalität soll im Folgenden kurz eingegangen werden.

Erfassung neuer Inhalte

Auf der Sitemap des zu untersuchenden Board-Systems werden die neuesten Einträge immer auf der ersten Seite eingeordnet. Auf Grund dessen wird für die Aktualisierung der Daten der Link zu dieser dem Internet-Crawler übergeben. Dieser parst die 250 Einträge und gibt, wie vorher beschrieben, die Rohdaten des Board-Systems zurück. Anschließend werden die im XML-gespeicherten Daten in das Datenmodell eingelesen und die Titel miteinander verglichen. Sollte ein neues Thema vorhanden sein, werden dessen Rohdaten an den HTML-Parser übergeben und ins XML-Dokument gespeichert. Im Moment werden alle neuen Daten an das XML-Dokument der ersten Seite der Sitemap angehängt, was bei einem längeren Betrieb der Software zu Problemen hinsichtlich der Performance führen könnte. Außerdem werden im Moment nur die Titel verglichen und nicht die Eintragsanzahl der Themen.

Nachdem die Implementierung der Erfassung und Haltung der Board-Daten aus den vorangehenden Ausführungen deutlich wurde, soll im Folgenden auf die Verarbeitung zu einer Ontologie eingegangen werden. Weiter soll die angewandte Methodik zur Bestimmung von möglichen Termen zum Aufbau einer Taxonomie vorgestellt werden.

5.2 Ontology Mapping

5.2.1 Technologieauswahl für die Kommunikation mit der OWL-Ontologie.

Auf Basis der erfassten Daten des Forschungsgegenstands Android-Hilfe.de erfolgt eine Weiterverarbeitung zu einer Ontologie. Im Zuge der Implementierung wurden mehrere Bibliotheken evaluiert, die sich zum Erstellen einer Ontologie im Java-Umfeld eignen. Im Zuge dieser Technologierecherche hinsichtlich Frameworks, die das Umsetzen einer Ontologie mit der Beschreibungssprache OWL ermöglichen, fanden sich drei Systeme, die diesem Kriterium entsprechen:

- Protègè-OWL Api
- OWL-API
- Apache Jena Ontology API

Alle drei Bibliotheken zeichnen sich durch eine gute Dokumentation und einer großen Anzahl an Kodebeispielen für Einsteiger aus. Die Nutzung der Protègè-OWL Api wurde jedoch wegen der hohen Komplexität und dem überdimensionierten Funktionsumfang rasch ausgeschlossen. Im Vergleich der beiden anderen Systeme wurde die Apache Jena Ontology API ausge-

wählt. Die Auswahl begründet sich einerseits aus der schnellen und einfachen Implementierung, andererseits aus dem intuitiven Umgang mit der Bibliothek. Im Gegensatz zu der OWL-API wirkte das Arbeiten mit Jena unkomplizierter. Folgendes Kodebeispiel zeigt das Erstellen einer OWL-Klasse und die Zuordnung zu einer Superklasse in den jeweiligen Systemen.

```
1 //OWL-API
2 try {
3 OWLClass superCls = mOntologyFactory.getOWLClass(IRI.create(ontologyIRI
      + superClass));
4 OWLClass subCls = mOntologyFactory.getOWLClass(IRI.create(ontologyIRI +
      classToInsert));
5
6 OWLAxiom axiom = mOntologyFactory.getOWLSubClassOfAxion(superCls, subCls
      );
7 AddAxiom addAxiom = new AddAxiom(ontology, axiom)
8 manager.applyChanges(addAxiom)
9 manager.saveOntology(ontology);
10 catch (OWLException e) {
11 e.printStackTrace();
12 }
13 //Jena-API
14 OntClass superCls = mOntologyModel.getOntClass(mNamespace + superClass);
15 OntClass subCls = mOntologyModel.createClass(mNamespace + classToInsert)
      ;
16 superCls.addSubClass(subCls);
17 try {
18 OntologyManager.getInstance().saveOntology(mOntologyPath);
19 } catch (FileNotFoundException e) {
20   e.printStackTrace();
21 }
```

Anhand dieses kurzen Beispieles wird deutlich, dass das Manipulieren der Ontologie mit Jena direkter erfolgt. Während mit der OWL-API ein Axiom bestimmt wird, mit dem letztendlich die Struktur zur Ontologie hinzugefügt wird, erfolgt bei der Jena Ontology API eine direkte Zuordnung über die Klassen, was den Kode lesbarer und einfacher verständlich macht und somit das Arbeiten mit dieser Bibliothek stark vereinfacht. Deswegen wurde für die Definition der Schnittstelle zur Ontologie auf die Apache Jena Ontology API zurückgegriffen.

5.2.2 Schnittstelle zur Ontologie

Die grundlegenden Konzepte der Beschreibungssprache OWL sind, wie in den vorangegangenen Ausführungen bereits erläutert, Klassen, Rollen, sowie Individuen. Klassen beschreiben dabei eine Gruppe von Individuen, die gleiche Eigenschaften besitzen und lassen sich in hierarchischer Form abbilden. Individuen sind die konkreten Objekte der jeweiligen Klassen und können mittels konkreter Rollen miteinander in Beziehung gesetzt werden. Abstrakte Rollen dienen dazu, Klassen miteinander zu verknüpfen. Darüber hinaus ist es mög-

lich, die Wertebereiche der Klassen und der Individuen durch verschiedene Restriktionen einzuschränken. Die beschriebenen Möglichkeiten zur Erstellung einer Ontologie mittels der Beschreibungssprache OWL mussten für die Verarbeitung der erfassten Daten zur Verfügung gestellt werden. Dafür wurde einerseits, wie vorhergehend beschrieben, die Apache Jena OWL-API verwendet. Andererseits erfolgte die Entwicklung einer Datenstruktur, um eine Kapselung der Interaktion mit der Ontologie von dem eigentlichen Mapping-Prozess zu erreichen und eine zentrale Stelle zur Kommunikation mit der Ontologie zu definieren. Alle Operationen zur Bearbeitung der Ontologie erfolgten unter Verwendung der ApacheJena Ontology Api mittels der Klasse *OntModel*, die geeignete Methoden dafür bereitstellt. Die Manipulationen an der Ontologie werden demnach nicht direkt an dieser vorgenommen, sondern werden an diesem Modell vorgenommen. Um das *OntModel* global verfügbar zu machen, wurde ein Singleton implementiert, in dem die zu bearbeitende Ontologie in das Modell eingelesen wird. Die Erzeugung dieses Modells kann anhand folgenden Kodebeispiels nachvollzogen werden:

```
1 public OntModel readOntology(String pathToOntology, String nameSpace)
      throws FileNotFoundException {
2 InputStream inputStream = new FileInputStream(pathToOntology);
3   mNameSpace = nameSpace;
4 mOntologyModel =
5 ModelFactory.createOntologyModel(OntModelSpec.OWL_MEM);
6   mOntologyModel.read(inputStream, mNameSpace);
7   return mOntologyModel;
8 }
```

Weiter wird in diesem der Namensraum gehalten und eine Methode zum Speichern der Änderung in die Ontologie bereitgestellt. Im Zuge der Übertragung der Daten des Board-Systems in die Ontologie werden vor allem die Methoden zum Erstellen und Abrufen von Klassen, Individuen und Rollen benötigt. Die Struktur für diese Interaktionen mit der Ontologie wurde nach dem Core J2EE Pattern Data Access Object [50] entwickelt.

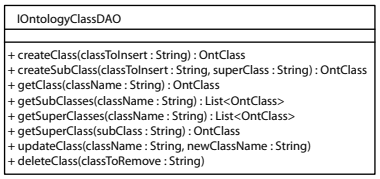

Abbildung 5.3: Klassendiagramm: Implementierung DAO-Interfaces.

Dafür wurden Interfaces implementiert, die alle benötigten Operationen zur Manipulation der Ontologie definieren. Das in Abb. 5.3 ersichtliche Interface steht examplarisch für den grundlegenden Aufbau dieser Schnittstellen.

Die konkreten Implementierungen der Interfaces sollen im Folgenden kurz anhand von Kodebeispielen beschrieben werden. In jedem Data Access Object wird zunächst im Konstruktor das *OntModel* und der Namensraum der Ontologie vom *OntologyManager* bezogen. Mit Hilfe des Modells können die Änderungen an der Ontologie vollzogen werden. Im Folgenden Kodebeispiel werden exemplarisch die Anweisungen für die Erzeugung und Abfrage der benötigten OWL-Sprachkonstrukte abgebildet.

```
1  //OWL-Klassen
2  //Erzeugen einer Klasse und Zuordnung als Subklasse
3  OntClass superCls = mOntologyModel.getOntClass(mNamespace + superClass);
4  OntClass subCls = mOntologyModel.createClass(mNamespace + classToInsert)
       ;
5  superCls.addSubClass(subCls);
6  //Abruf einer Klasse
7  mOntologyModel.getOntClass(mNamespace + className);
8  //Abruf aller Subklassen als Liste
9  mOntologyModel.getOntClass(mNamespace + className).listSubClasses()
10 .toList();
11 //Abstrakte Rollen
12 //Erzeugen
13 ObjectProperty property = mOntologyModel.createObjectProperty(mNamespace
       + propertyName);
14 property.addDomain(domain);
15 property.addRange(range);
16 //Verknüpfung zweier Klassen mit einer abstrakten Rolle
17 Statement statement = mOntologyModel.createStatement(subject, verb,
       object);
18 mOntologyModel.add(statement);
19 //Konkrete Rollen
20 //Erzeugen
21 DatatypeProperty property = mOntologyModel.createDatatypeProperty(
       mNamespace + propertyName);
22 property.setDomain(domain);
23 property.setRange(range);
24 //Verknüpfung eines Individuums mit einem Wert mittels einer konrekten Rolle
25 Statement statement = mOntologyModel.createStatement(subject, verb,
       literal);
26 mOntologyModel.add(statement);
27 //Individuen
28 //Erzeugen
29 Individual ind = cls.createIndividual(mNamespace + individual);
30 //Abruf aller einer Klasse zugeordneten Individuen
31 (List<Individual>) cls.listInstances().toList();
```

Es wird deutlich, dass mit einer geringen Kodemenge die wichtigsten Operationen für die Bearbeitung einer Ontologie implementiert werden können. Zudem kann nachvollzogen werden, dass wie vorher beschrieben, alle Veränderungen an der Ontologie über das *OntModel* vorgenommen werden. Neben dem Erzeugen von Klassen, Rollen und Individuen bietet die OWL-Beschreibungssprache die Möglichkeit, weitere Restriktionen auf die erstellten Konzepte zu setzen. Dabei ist es möglich, Einschränkung hinsichtlich des

Wertebereichs der Properties vorzunehmen und die Anzahl der verknüpften Elemente mittels einer Kardinalität zu begrenzen.

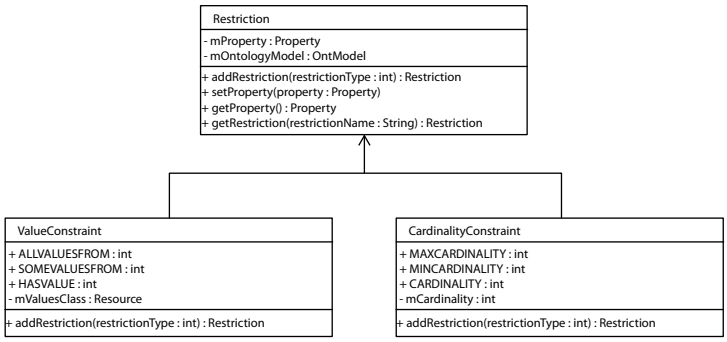

Abbildung 5.4: Klassendiagramm: Implementierung Restriktionen.

Das in Abb. 5.4 ersichtliche Klassendiagramm beschreibt die Implementierung hinsichtlich der möglichen OWL-Restriktionen. Der Methode *addRestriction()* wird dabei der gewünschte Restriktionstyp übergeben, die jeweilige Restriktion gesetzt und zurückgegeben.

Nach Abschluss des Bearbeitungsprozess muss das Modell in die Ontologie gespeichert werden. Dieser Vorgang kann über eine Methode im *OntologyManager* angestoßen werden:

```
1 OutputStream out = new FileOutputStream(pathToOntology);
2     if(mOntologyModel != null) {
3       mOntologyModel.write(out);
4     } else {
5       InputStream inputStream = new FileInputStream(pathToOntology);
6       mOntologyModel = ModelFactory.createOntologyModel(OntModelSpec.
      OWL_MEM);
7       mOntologyModel.read(inputStream, mNameSpace);
8       mOntologyModel.write(out);
9     }
```

Treten keine Fehler auf, werden die Daten in die angegebene Ontologie übertragen und können mittels eines Editors wie Protègè betrachtet werden.

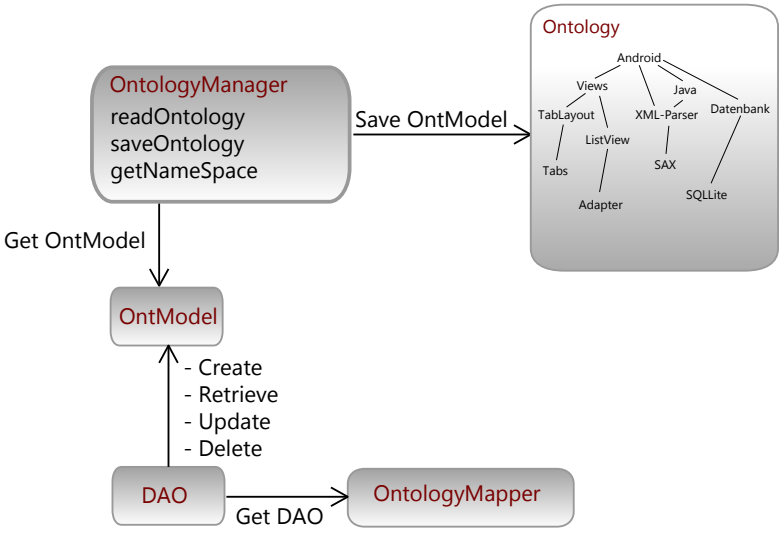

Abbildung 5.5: Überblick über den Kommunikationsprozess mit der Ontologie.

Ein Überblick des vorangehend beschriebenen Prozesses zur Bearbeitung der Ontologie wird Abb. 5.5 visualisiert.

Nachdem erläutert wurde, wie die Kommunikation mit der Ontologie implementiert worden ist, soll im Folgenden auf die Übertragung der Daten des Forschungsgegenstands Android-Hilfe.de in die Ontologie beschrieben werden.

5.2.3 Ontology Mapping: Boarddaten

Der Ontology-Mapping Prozess für das Übertragen der gesammelten Daten des Forschungsgegenstands Android-Hilfe.de besteht aus zwei Schritten. Zuerst wird eine Struktur für das Board-System in der Ontologie aufgebaut. Darauf aufbauend erfolgt im zweiten Schritt eine Zuordnung und Verknüpfung der Themen, Antworten und Benutzer mittels der erstellten Konzepte.

Aufbau der Struktur

Für das Erzeugen der Struktur des Board-Systems werden einerseits alle erstellten *Data Access Objekte* benötigt, andererseits Methoden um die OWL-Konstrukte in der Ontologie zu erstellen. Dafür wurde eine abstrakte Super-

klasse implementiert, die alle DAOs instanziiert und innerhalb der erbenden Klassen zur Verfügung stellt, sowie die benötigten Methoden definiert. Die entwickelte Programmstruktur kann in Abb. 5.6 nachvollzogen werden. Im Folgenden soll die Implementierung der Methoden anhand des Aufbaus der Struktur für die Themen des Board-Systems mit dazugehörigen Kodebeispielen erläutert werden.

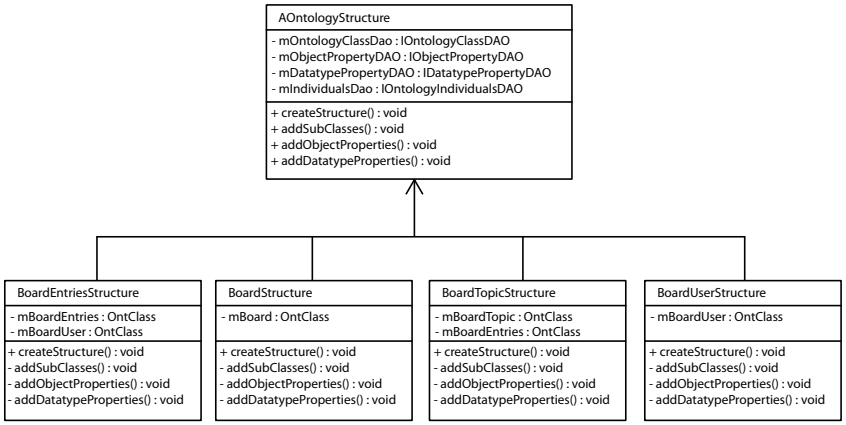

Abbildung 5.6: Klassendiagramm: Erzeugung der Struktur für die Ontologie.

Die Methode *createStructure()* dient dazu, den Aufbau der Struktur in der jeweiligen Klasse anzustoßen, und ruft dementsprechend die anderen Methoden auf, wie in folgenden Kodebeispiel ersichtlich wird.

```
1 @Override
2 public void createStructure() {
3   addObjectProperties();
4   addDatatypeProperties();
5   addSubClasses();
6 }
```

In der Methode *addSubClasses()* werden die benötigten Subklassen für das Board-System generiert. Für die Klassennamen wurden Java-Enums erstellt, damit diese an einer zentralen Stelle definiert und geändert werden können.

```
1 @Override
2 public void addSubClasses() {
3   mOntologyClassDao.createSubClass(EOntClassNames.BOARDTOPIC.
      getSpecificOntClassName(), mBoardTopic.getLocalName());
4   mOntologyClassDao.createSubClass(EOntClassNames.BOARDENTRIES.
      getSpecificOntClassName(), mBoardEntries.getLocalName());
5 }
```

Weiter werden in *addObjectProperties()* die abstrakten Rollen in der Ontologie erzeugt. Dafür wurden die von Java bereitgestellten Reflections verwen-

det. Dementsprechend werden die Feldnamen der jeweiligen Klasse bezogen und in einem Array gespeichert. Dieses wird durchlaufen und die Felder hinsichtlich ihres Datentyps untersucht. Sollte es sich nicht um einen primitiven Datentyp handeln, wird eine abstrakte Rolle in der Ontologie erzeugt. Weiter erfolgt mittels dieser Methode die Generierung von Restriktionen. Im folgenden Kodebeispiel wird das Thema mit den Einträgen verknüpft und festgelegt, dass jedem Thema einige Beiträge zugeordnet werden können.

```
1 @Override
2 public void addObjectProperties() {
3   Field[] fields = BoardTopic.class.getDeclaredFields();
4   for (Field field : fields) {
5     String fieldName = field.getName().replaceFirst("m", "");
6     if(field.getType().getName().equalsIgnoreCase("java.util.ArrayList")
      ) {
7       ObjectProperty hasProperty = mObjectPropertyDAO.
      createObjectProperty(mBoardTopic, mBoardEntries, "has" + fieldName);
8       mObjectPropertyDAO.addStatement(mBoardTopic, hasProperty,
      mBoardEntries);
9       mBoardTopic.addSuperClass(OntologyUtils.addValueRestriction(
      hasProperty, ValueConstraint.SOMEVALUESFROM, mBoardEntries));
10    }
11  }
12 }
```

Das Erzeugen der konkreten Rollen folgt dem gleichen Prinzip wie dem der abstrakten Rollen. Das folgende Kodebeispiel zeigt die Implementierung der Methode und einen Auszug aus der dazugehörigen Methode *writeDataTypeProperties(Field field, OntClass ontClass)*, in der die Datentypen abgefragt und darauf basierend die konkreten Rollen erstellt werden. In diesem Fall wird das Datatype-Property mittels einer Restriktion hinsichtlich der Kardinalität versehen. Diese Restriktion gilt für alle konkreten Rollen, da im Falle des Board-Systems jedem Individuum nur ein Wert zugeordnet werden kann.

```
1 @Override
2 public void addDatatypeProperties() {
3   Field[] fields = BoardTopic.class.getDeclaredFields();
4   for (Field field : fields) {
5     OntologyUtils.writeDataTypeProperties(field, mBoardTopic);
6   }
7 }
8
9 public static void writeDataTypeProperties(Field field, OntClass
      ontClass) {
10  DatatypeProperty property = null;
11  DatatypePropertyDAO datatypePropertyDAO = new DatatypePropertyDAO();
12  String fieldName = field.getName().replaceFirst("m", "");
13  if(field.getType().getName().equalsIgnoreCase("java.lang.String")) {
14    property = datatypePropertyDAO.createDatatypeProperty(ontClass, XSD.
      xstring, "has" + fieldName);
15    ontClass.addSuperClass(addCardinalityRestriction(property,
```

```
      CardinalityConstraint.CARDINALITY, 1));
16   }
17 [\ldots]
18 }
```

Die vorgestellten Methoden werden in den in Abb. 5.6 ersichtlichen Klassen analog implementiert. Die Erzeugung der Struktur für die Ontologie wird zentral in der Klasse *BoardStructureMapper* mit der Methode *createOntologyStructure* angestoßen. Nachdem die Struktur in der Ontologie erzeugt wurde, können im zweiten Schritt die gesammelten Daten des Board-Systems übertragen werden. Dieser Prozess soll im Folgenden in den Fokus der Ausführungen gestellt werden.

Übertragen der Daten in die Ontologie

Das Übertragen der Daten in die Ontologie wird über die Klasse *BoardMapper* mit der enthaltenen Methode *mapBoardToOntology()* gestartet. Ähnlich wie bei der Erstellung der Struktur werden auch Java-Reflections eingesetzt. Das folgende Kodebeispiel zeigt das Hinzufügen eines Themas zu der Ontologie. Dafür werden im ersten Schritte die Felder der Klasse bezogen und danach ein Individuum in der Ontologie erzeugt. Im Anschluss werden alle konkreten Rollen bezogen und mit dem jeweiligen Wert des Feldes verknüpft.

```
1 public void writeTopicToOnt(BoardTopic forumThread, String
      individualName) {
2   Field[] forumThreadFields = BoardTopic.class.getDeclaredFields();
3   Individual threadIndividual = mIndividualsDAO.createIndividual(
      mBoardTopic, individualName);
4   for(DatatypeProperty property : mDatatypePropertyDAO.
      getDatatypeProperties()) {
5     for(Field field : forumThreadFields) {
6       String threadValue = OntologyUtils.getValueForField(field,
      forumThread, field.getName());
7       if((property.getLocalName()).contains(field.getName().replaceFirst
      ("m", ""))) {
8         threadIndividual.addProperty(property, threadValue);
9       }
10    }
11  }
12 }
```

Um die Feldwerte zu beziehen wurden in der jeweiligen Klasse Get-Methoden erzeugt. Der Aufruf dieser Methode erfolgt dynamisch über Java-Reflections anhand des jeweiligen Feldnamens, wie an folgendem Kodebeispiel nachvollzogen werden kann.

```
1 String methodPostfix = fieldName.replaceFirst("m", "");
2 Method meth = null;
3 if (field.getType().getName().equals("java.lang.Boolean")) {
4   String test = methodPostfix.substring(0, 1).toLowerCase();
5   methodPostfix = methodPostfix.replaceFirst("\\w", test);
```

```
 6   meth = obj.getClass().getMethod(methodPostfix, new Class[] {});
 7 } else {
 8   meth = obj.getClass().getMethod("get" + methodPostfix, new Class[] {})
       ;
 9 }
10 Object retObj = meth.invoke(obj, new Object[] {});
```

Das zurückgegebene Objekt wird auf den Datentyp des Feldes gecastet und mit dem über die konkrete Rolle verknüpften Individuum in die Ontologie übertragen.

Aus den vorangegangenen Ausführungen wurde deutlich, wie die Übertragung der gesammelten Daten des Forschungsgegenstands Android-Hilfe.de und dem damit verbundenen Aufbau einer geeigneten Struktur in der Ontologie implementiert wurde. Im folgenden Abschnitt soll auf die textuelle Analyse der Themen und Einträge eingegangen werden. ImZugedessen wird erläutert, wie die Auswahl von geeigneten Termen für die Erstellung einer Taxonomie entwickelt wurde.

5.3 Natural Language Processing

Das im Rahmen dieser Arbeit entwickelte Projekt verfügt neben der Datenerfassung und der Verarbeitung dieser zu einer Ontologie auch über eine Komponente für die Auswertung der textuellen Daten. Für die Realisierung dieser Textanalyse mussten Methoden aus dem Natural Language Processing implementiert werden. Diese sollen nachfolgend in den Mittelpunkt der Ausführungen gestellt werden. Auf Basis dieser Implementierung wurde aus den Themen und Beiträgen des Forschungsgegenstands Android-Hilfe.de relevante Terme für den Aufbau einer Taxonomie in der Domäne Android App-Entwicklung extrahiert. Diese wurden mittels der aus dem Information Retrieval bekannten TF-IDF Formel gewichtet und nach Relevanz sortiert in ein XML-Dokument verarbeitet. Eine Beschreibung der Umsetzung der Termgewichtung folgt im zweiten Teil dieses Abschnitts.

5.3.1 Aufbereitung der textuellen Daten

Die Aufbereitung der textuellen Daten hat die Aufgabe, bestimmte Satzteile bzw. Wortarten aus den Beträgen des Board-Systems zu extrahieren. Basierend auf diesen Syntaxfragmenten werden die relevanten Terme bestimmt. Zur Durchführung der Syntaxanalyse müssen jedoch zunächst die Sätze innerhalb des Textes erkannt werden.

Satzerkennung

Die Zerlegung des Textes in die einzelnen Sätze erfolgt unter Verwendung der Bibliothek Apache OpenNLP [51], welche diese Funktionalität unter anderem für die benötigte Sprache Deutsch bereitstellt. Das Erkennen von Sätzen

innerhalb eines Beitrags erscheint zunächst als triviale Aufgabe, jedoch treten einige Problem auf Grund der Ambiguitäten der Satzzeichen auf, was in folgendem Zitat von Carstensen et al. [5] deutlich wird:

> [...] Ambiguitäten ergeben sich daher vor allem durch Satzzeichen. Ausrufe- und Fragezeichen werden zwar selten in Markennamen wie Joop! oder Guess? gebraucht, markieren aber für gewöhnlich genau so wie das Semikolon und der Doppelpunkt ein Satzende. Der Punkt wird allerdings nicht nur in dieser Funktion verwendet, sondern auch, um Abkürzungen oder Datumsangaben zu kennzeichnen oder als Trennzeichen in Zahlen.

Auf eine eigens implementierte Lösung wurde daher verzichtet und auf OpenNLP zurückgegriffen. Die Konfiguration der Satzerkennungskomponente des ausgewählten Frameworks erfolgt mit der Instanziierung der Klassen *SentenceDetectorME* und der Auswahl des für die Sprache notwendigen Modells, was in folgendem Kodebeispiel nachvollzogen werden kann.

```
1 SentenceDetectorME sentenceDetector = null;
2     mModelIn = null;
3     try {
4       mModelIn = new FileInputStream("data/nlp/sentencedetection/de-sent
      .bin");
5       final SentenceModel sentenceModel = new SentenceModel(mModelIn);
6       mModelIn.close();
7       sentenceDetector = new SentenceDetectorME(sentenceModel);
8
9     } catch (final IOException ioe) {
10      ioe.printStackTrace();
11    } finally {
12      if (mModelIn != null) {
13        try {
14          mModelIn.close();
15        } catch (final IOException e) {
16          System.out.println("Exception at: " + BoardSentenceDetector.
      class.getSimpleName() + " \nException: " + e.toString());
17          e.printStackTrace();
18        }
19      }
20    }
```

Nach Konfiguration des Frameworks kann über die Methode *sentDetect(String text)* ein Array mit den erkannten Sätzen bezogen werden, das für die weitere Syntaxanalyse verwendet wird.

Syntaxanalyse

Im Zuge der Syntaxanalyse werden die Satzteile und deren Funktion im Satz bestimmt. Dieser Prozess gestaltet sich für die deutsche Sprache als sehr kompliziert, da im Gegensatz zum Englischen die Satzstruktur wesentlich freier

gestaltbar ist. Während im Englischen durchgängig die Reihenfolge Subjekt-Prädikat-Objekt eingehalten wird, ist es im Deutschen durchaus denkbar, dass das Objekt an erster Stelle steht [20]. Die Stanford Universität hat für diesen komplexen Prozess einen Parser entwickelt, der im hier entwickelten Projekt zum Einsatz kam. Dieser arbeitet mit probabilistischen kontextfreien Grammatiken und erstellt mittels POS-Tagging Baumgraphen für die geparsten Sätze. Im Folgenden soll kurz ein Exkurs in die Computerlinguistik vorgenommen werden, um die im Zuge der Syntaxanalyse verwendeten Konzepte zu erläutern.

> Ziel der automatischen Wortartenannotierung (auch POS-Tagging, von engl. part of speech) ist es herauszufinden, ob z. B. die Wortform gleichen als Adjektiv (die gleichen Beispiele) oder als Verb (Kaninchen gleichen Hasen) gebraucht wird. Tatsächlich trifft man meist wesentlich feinere Unterscheidungen (z.B. Infinitiv vs. Indikativ vs. Imperativ bei Verben), die durch ein sogenanntes Tagset festgelegt werden. [5]

Es werden demnach die Satzteile und deren Funktion bzw. Stellung im Satz durch ein Tagset bestimmt. Der Stanford Parser verwendet das Stuttgart-Tübingen-Tagset (STTS) [52] und ordnet diese mittels einer probabilistischen kontextfreien Grammatik dem passenden Satzteil zu. Der Aufbau einer solchen Grammatik lässt sich anhand folgenden einfachen Beispiels nachvollziehen.

S	→	NP VP	1
NP	→	DET N	0,67
NP	→	PRON	0,33
VP	→	V	0,4
VP	→	V NP	0,6

Abbildung 5.7: Beispiel einer probabilistischen kontextfreien Grammatik [5].

Abb. 5.7 zeigt die definierten Regeln einer Grammatik. Daraus folgt, dass ein Satz durch eine Nominalphrase (NP) und eine Verbalphrase (VP) beschrieben wird und das Gewicht 1 erhält, da nur diese Expansion definiert wird. Weiter wird beschrieben mit welcher Wahrscheinlichkeit ein bestimmter Aufbau der Nominal- und Verbalphrase auftritt. Folgt man dem Beispiel wird festgelegt, dass eine Nominalphrase mit doppelt so großer Wahrscheinlichkeit aus einem bestimmten Artikel und einem Nomen besteht, wie aus einem Pronomen [5]. Auf Basis der erstellten Grammatik und des Tagsets wird ein Baumgraph erstellt. Ein Beispiel wird in Abb. 5.8 gezeigt.

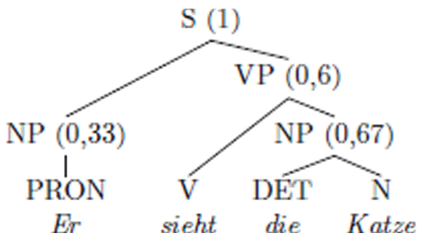

Abbildung 5.8: Beispiel eines Baumgraphen [5].

Für die Extraktion möglicher Terme, einerseits zur Verschlagwortung der einzelnen Themen und andererseits für den Aufbau einer Taxonomie, werden die Baumgraphen eingesetzt. Bei den extrahierten Termen handelt es sich um Substantive, da sich diese für die Beschreibung eines Dokuments am besten eigenen. Das Stuttgart-Tübingen-Tagset unterscheidet dabei zwischen zwei Arten von Nomen: Eigennamen (NE) und normalen Nomen (NN). Da es sich um ein Board-System aus dem Bereich Android App-Entwicklung handelt, werden häufig fremdsprachliche Begriffe eingesetzt. Auf Grund dessen wurde das Tag fremdsprachichge Mittel (FM) als Extraktionskriterium mitaufgenommen. Die extrahierten Terme werden in das in Abb. 5.9 ersichtliche Datenmodell gespeichert, bevor die Termgewichtung vorgenommen wird.

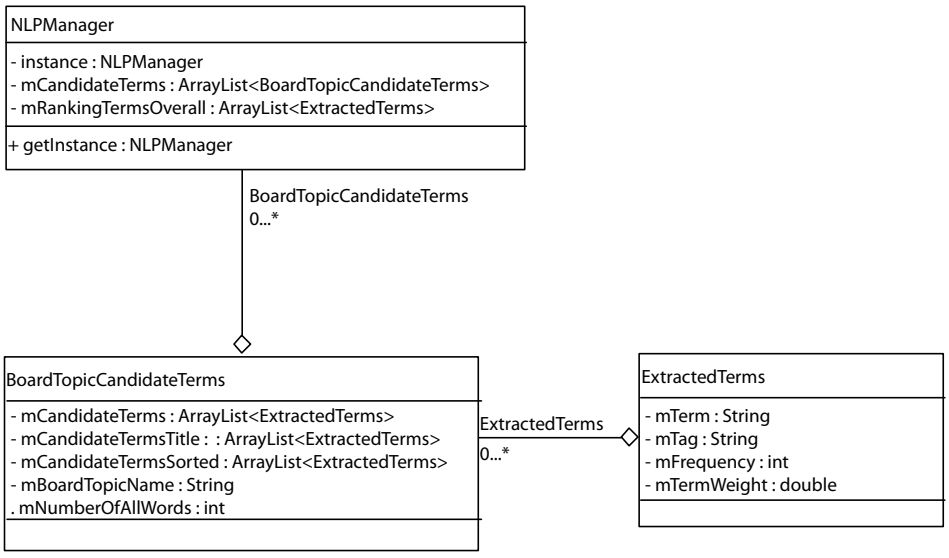

Abbildung 5.9: Datenmodell für die extrahierten Terme.

Mit Hilfe der Termgewichtung ist es möglich die extrahierten Terme nach Relevanz zu sortieren. Die Implementierung für die Berechnung der Termrelevanz soll nachfolgend thematisiert werden.

5.3.2 Bestimmung der Termrelevanz

Für die Berechnung der Relevanz eines Terms zur Beschreibung eines Themas des Forschungsgegenstands Android-Hilfe.de wurde die TF-IDF Formel verwendet. Diese Kalkulation erfolgt über die Auftretenshäufigkeit des Terms und die Anzahl aller Wörter im Text. Weiter müssen die Anzahl aller Dokumente im Korpus und die Auftretenshäufigkeit des Wortes in allen Dokumenten bekannt ermittelt werden. Zur Bestimmung der Termfrequenz innerhalb eines Themas werden die extrahierten Terme miteinander verglichen und im Falle einer Überstimmung von Wörtern aufsummiert. Damit jedoch der Vergleich von gleichen Termen mit unterschiedlicher Flexion nicht zu falschen Ergebnissen führt, muss vor der Berechnung eine Grundformenreduktion [1] durchgeführt werden. Dafür wurde das Snowball Framework, entwickelt von Porter, implementiert, dass eine Grundformenreduktion für das Deutsche ermöglicht. In folgendem Kodebeispiel ist die Konfiguration und die Nutzung des Stemmers abgebildet.

```
1 Class stemClass = Class.forName("nlp.stemmer.org.tartarus.snowball.ext.
      germanStemmer");
2       extractedTerms.add(term.getTerm());
3       SnowballStemmer stemmer = (SnowballStemmer) stemClass.newInstance
      ();
4       stemmer.setCurrent(term.getTerm());
5       stemmer.stem();
6       term.setTerm(stemmer.getCurrent());
```

Die Gesamtanzahl der Wörter wird im Zuge der Syntaxanalyse vorgenommen, in der jeder Satz in die Wörter unterteilt wird. Im Zuge dieser Unterteilung werden die Wörter gezählt und in der Klasse *BoardTopicCandidateTerms* gespeichert. Die Anzahl der Dokumente im Korpus ergibt sich aus der Anzahl der Themen in einem XML-Dokument, demnach 250. Für die Errechnung der Auftretenshäufigkeit des Terms innerhalb aller Dokumente wurde die im nachfolgenden Kodebeispiel ersichtliche Methode implementiert, die anhand einer Liste aller vorkommenden Terme die Anzahl bestimmt.

```
1 private int countTermOccurence(String term) {
2     int count = 0;
3     for(int i = 0; i < mExtractedTerms.size(); i++) {
4       if(mExtractedTerms.get(i).equalsIgnoreCase(term)) {
5         count++;
6       }
```

[1] Bei einer Grundformenreduktion werden die durch Flexion oder Deklination auftretenden Präfixe oder Suffixe eines Wortes entfernt, um die Stammform zu erhalten. Beispiele für Wörter und ihre Stammformen können unter [53] nachgesehen werden.

```
7     }
8     return count;
9   }
```

Nach Ermittlung aller benötigten Werte, wird das Termgewicht mittels der TF-IDF-Formel berechnet, wie im Folgenden Kodesegment dargestellt wird.

```
1 termWeight = ((double)term.getFrequency() / (double)numberOfWords) *
      Math.log(NUMBERTEXTSINKORPORA / countTermOccurence(term.getTerm()));
```

Neben der Gewichtung mit der TF-IDF-Formel werden einerseits die Wortart und andererseits die Stellung der Terme im Text berücksichtigt. Da es sich um ein Board-System mit der Domäne Software-Entwicklung handelt sind fremdsprachliche Mittel höher zu bewerten und auf Grund dessen wird deren Termgewicht mit einem Faktor von 1,1 multipliziert. Weiter haben Eigennamen eine hohe Relevanz für die Beschreibung eines Themas, weswegen ein Faktor von 1,2 bestimmt wurde. Die aussagekräftigsten Terme für den Inhalt eines Board-Beitrags stehen im Titel des Themas. Deswegen werden alle aus dem Titel gewonnenen Begriffe mit einem Faktor von 1,5 multipliziert. Auf Basis dieser Gewichtung kann eine Rangliste der Terme erstellt werden. Dabei erfolgt eine Sortierung nach absteigendem Gewicht, wodurch die relevantesten Ausdrücke am Anfang stehen. Die Ergebnisse werden in zwei XML-Dokumenten gespeichert. Das erste enthält eine Auflistung aller Themen mit den dazugehörigen relevantesten Termen. Im zweiten werden alle extrahierten Terme in einer Rangliste dargestellt.

Aus den vorangegangenen Ausführungen wurde deutlich, wie das System aufgebaut wurde und die Implementierung erfolgt ist. Außerdem sind die verwendeten Frameworks und Bibliotheken vorgestellt worden. Im nächsten Kapitel soll eine Evaluierung der erstellten Software mit einer Auswertung der Ergebnisse, sowie einem Vergleich mit bestehenden Systemen, vorgenommen werden.

Kapitel 6

Evaluierung

In den folgenden Erläuterungen hinsichtlich der Evaluierung des entwickelten Systems soll überprüft werden, ob die Zielsetzung dieser Arbeit erfüllt wurde. Außerdem soll ein kurzer Vergleich mit bestehenden Methoden aus dem Feld Ontology Learning gegeben werden. Wie aus den vorangegangenen Ausführungen deutlich wurde, war es Ziel im Rahmen dieser Arbeit eine Software zu entwickeln, die es ermöglicht, automatisch die Daten aus einem Board-System zu erfassen und zu einer Ontologie zu verarbeiten. Weiter sollten anhand der textuellen Daten mögliche Terme für den Aufbau einer Taxonomie bzw. einer Verschlagwortung eines Themas des Board-Systems extrahiert und nach Relevanz bewertet werden.

6.1 Datenerfassung und Ontology Mapping

6.1.1 Datenerfassung

Die Erfassung der Daten des Forschungsgegenstands Android-Hilfe.de erfolgt, wie im vorangehenden Kapitel beschrieben, mittels eines Internet-Crawlers. Das Board-System enthält in der Domäne Android App-Entwicklung zum Zeitpunkt der Untersuchung 4898 Themen. Auf Grund der immensen Datenmenge wurde einerseits ein Internet-Crawler implementiert, der eine parallele Verarbeitung unterstützt. Um die Laufzeit des Crawl-Prozesses möglichst gering zu halten, musste weiter darauf geachtet werden, dass die Linkstruktur in einer aufbereiteten Form vorliegt. Deswegen wurde die Sitemap des Board-Systems verwendet. Daraus ergibt sich eine Gesamtdauer der Datenerfassung inklusive Verarbeitung in ein XML-Dokument von 20 Minuten bei einer Bandbreite von 10 Mbit/Sekunde. Pro Seite mit jeweils 250 Einträgen werden demnach 1 Minute und 10 Sekunden benötigt. Auf Basis der spezifischen Struktur und der unterschiedlichen Daten wurde die Datenerfassung nur für den untersuchten Forschungsgegenstand implementiert und ist somit nicht allgemein auf alle Board-Systeme anwendbar. Ein Ansatz zum Crawlen von Forendaten unabhängig von der Datenquelle wurde unter an-

derem von der Microsoft Research Gruppe entwickelt. Diese verwenden zur Datenserfassung ebenfalls die Sitemap und arbeiten mit logischen Markov Netzwerken (MLNs), um unabhängig von der Strukturierung die folgenden Felder zu extrahieren [29]:

- post title
- post author
- post time
- post content

Für eine detailliertere Erläuterung der verwendeten Algorithmen und der Ergebnisse wird auf das Paper von Yang et al. [29] verwiesen. Mit Hilfe dieser Methodik könnte der implementierte Internet Crawler weiterentwickelt und somit Daten von mehreren Foren oder Boards erfasst werden ohne einen neuen HTML-Parser implementieren zu müssen.

Weiter ist es möglich, dass für die Datenerfassung die Datenbank des Board-Systems zur Verfügung steht. In diesem Fall können die entwickelten Funktionalitäten trotzdem verwendet werden. Es müssen die im Java-Datenmodell benötigten Werte aus der Datenbank bezogen oder die Datenbank in die definierte XML-Struktur überführt werden. Anschließend kann auf Basis dieser Daten das Ontology Mapping und die textuelle Analyse vorgenommen werden.

6.1.2 Ontology Mapping

Beschreibung und Ergebnisse

Die Übertragung der gesammelten Daten in die Ontologie erfolgte auf Basis der definierten XML-Struktur und des in Java implementierten Datenmodells. Daraus ergab sich, dass anhand des eingesetzten Composite Patterns und der Klassenabhängigkeiten die OWL-Konstrukte Klassen, Rollen und Individuen in der Ontologie abgebildet werden konnten. Dabei wurden die Java-Klassen zu OWL-Klassen übernommen und die Rollen nach dem in Tab. 6.1 ersichtlichen Muster erstellt.

Tabelle 6.1: Umwandlung von Java-Feldern zu Rollen in der Ontologie.

Datentyp	*Feldname*	*Rollenname*	*Domain*	*Range*
Datentyp	Feldname	hasFeldname	Klasse	Datentyp
Objekt	Feldname	hasKlassename	Klasse	instanziierte Klasse

Für die Themen und Beiträge werden mittels des Klassennamens, der Seite der Sitemap und einer ID Individuen erstellt. Mittels der konkreten Rollen erfolgte eine Zuordnung der Daten aus dem Board-System. Die Verknüpfung der OWL-Klassen mittels der abstrakten Rollen erfolgte auf Basis

des zugrunde liegenden Objekts. Eine Liste von Objekten beschreibt dabei, dass eine Klasse mehrere Werte aus einer anderen Klasse referenzieren kann. Die Instanziierung eines einzelnen Objekts führt zur Einschränkung, dass genau eine Verknüpfung mit einer anderen Klasse übernommen werden kann. Im Folgenden wird die aus dem Ontology Mapping entstandene Ontologie beschrieben. In Abb. 6.1 wird die Struktur für ein Thema des Forschungsgegenstands Android-Hilfe.de deutlich.

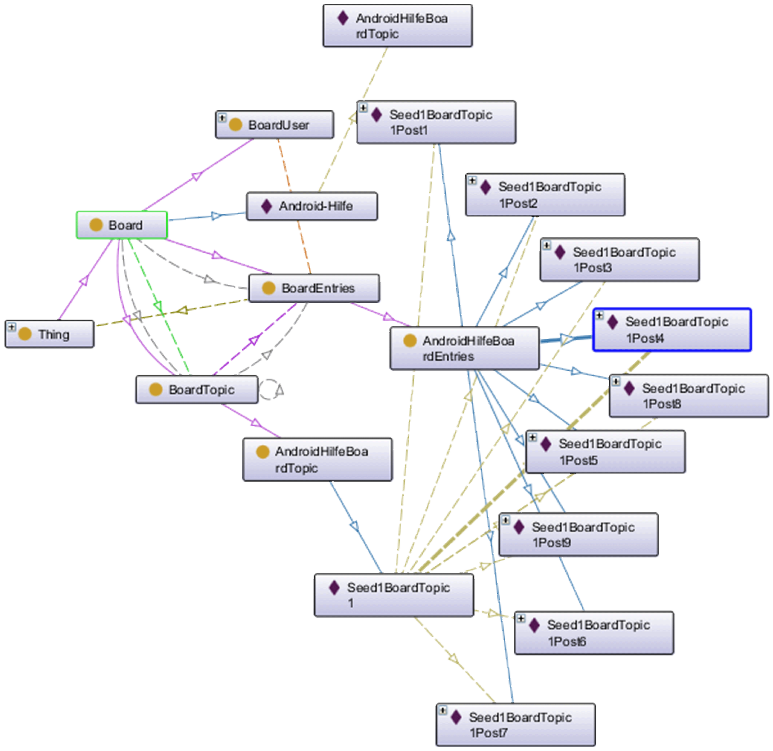

Abbildung 6.1: Struktur der erstellten Ontologie.

Für die einzelnen Themen wird die OWL-Klasse *AndroidHilfeBoardTopic* erstellt, der alle Beiträge mit einer abstrakten Rolle zugeordnet werden. Die Beiträge sind über die generierten konkreten Rollen mit den jeweiligen Werten verknüpft. In den folgenden Abbildungen sind die Klassen *Board*, *AndroidHilfeBoardTopic* und *AndroidHilfeBoardEntries* mit den zugehörigen Rollen und exemplarischen Werten dargestellt.

Abbildung 6.2: Ansicht Protégé – Klasse Board mit durch Rollen zugeordneter Werte.

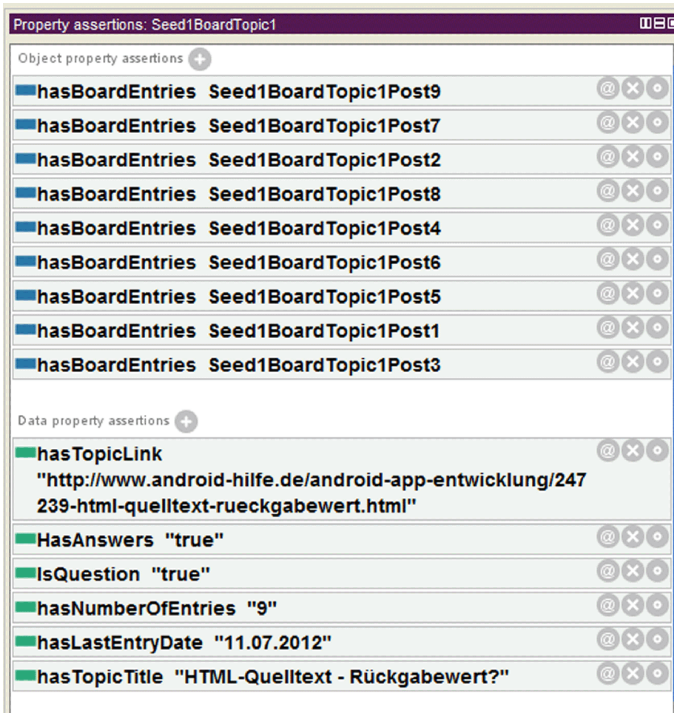

Abbildung 6.3: Ansicht Protégé – Klasse AndroidHilfeBoardTopic mit durch Rollen zugeordneter Werte.

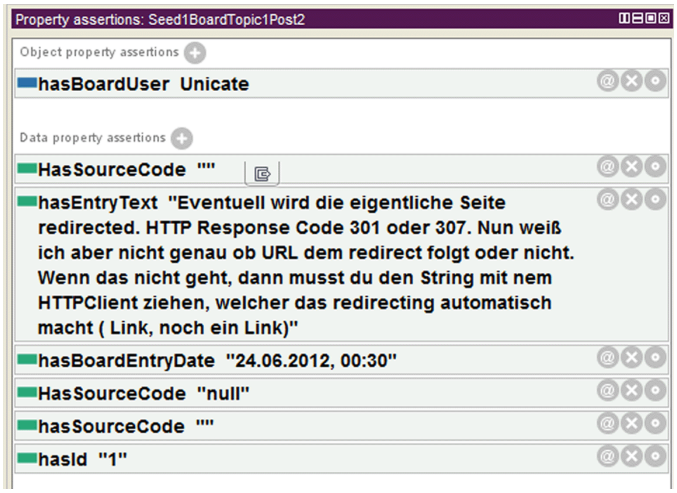

Abbildung 6.4: Ansicht Protégé – Klasse AndroidHilfeBoardEntries mit durch Rollen zugeordneter Werte.

Zudem wurden die gesammelten Daten über die Benutzer in die Ontologie eingepflegt. Dafür wurde eine Klasse *AndroidHilfeBoardUser* definiert, in der die Nutzernamen als Individuum angelegt worden sind.

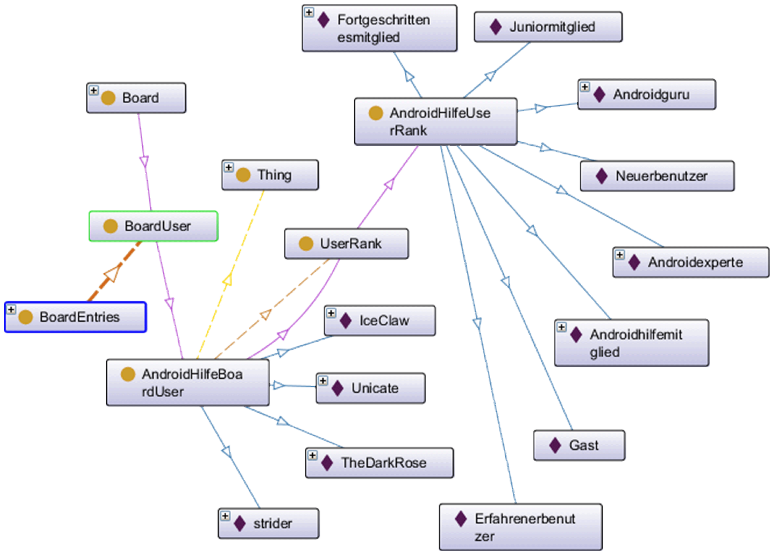

Abbildung 6.5: Struktur der Benutzer in der Ontologie.

Weiter ist eine Subklasse *UserRank* angelegt worden, in der die möglichen Benutzerklassen des Board-Systems enthalten sind. Durch die Abb. 6.5 wird der Aufbau der Benutzerstruktur in der Ontologie illustriert und in Abb. 6.6 die Verknüpfung dieser mit den zugehörigen Werten.

Abbildung 6.6: Ansicht Protégé – Klasse AndroidHilfeBoardUser mit durch Rollen zugeordneter Werte.

Nach Darstellung der entstandenen Ontologie soll nachfolgend ein kurzer Vergleich der implementierten Methode gegenüber der Ansätze zum Aufbau einer Ontologie anhand eines XML-Dokumentes gegeben werden.

Vergleich

Im Abschnitt 3.2.2 wurden Methoden beschrieben, die es ermöglichen anhand von XML-Dokumenten eine Ontologie zu erstellen. Dafür werden XML-Schemata oder DTDs benötigt. Auf Basis dieser Beschreibung der Struktur eines XML-Dokuments können Regelsysteme gebildet werden. In diesen Regeln wird definiert, welches Element aus dem Schema oder der DTD dem jeweiligen OWL-Konstrukt entspricht. In der im Rahmen dieser Arbeit entwickelten Methodik wird anhand einer speziellen XML-Struktur eine Ontologie aufgebaut. Ein weiterer Unterschied liegt darin, dass in der entwickelten Software keine reine Taxonomie erstellt, sondern eine Datenstruktur aufgebaut wird. Die Erstellung einer Taxonomie auf Basis eines XML-Dokuments wäre mit dem hier implementierten System jedoch durchaus denkbar. Der Mapping-Prozess ist, wie aus den vorangegangenen Ausführungen deutlich wurde, via Java-Reflections mit dem erstellten Datenmodell verbunden. Dieses Datenmodell bildet die XML-Struktur in Java ab. Um die Verarbeitung von XML-Dokumenten mit allgemeiner Struktur zu ermöglichen, müsste ein generisches Datenmodell erstellt werden, in dem die im XML gebildete Hierarchie abgebildet wird. Demzufolge wäre zum Beispiel ein Elternelement ohne direkten Wert eine Klasse und die dazugehörigen Kindelemente mit Werten die Felder. Nach Generierung des Datenmodells mittels der XML-Struktur könnte der hier entwickelte Ontology Mapper verwendet werden, um die nötigen OWL-Konstrukte zu bilden und in einer Ontologie zu speichern.

6.2 Termextrahierung und Termrelevanz

Für das im Rahmen dieser Arbeit entwickelte Projekt wurde neben dem Ontology Mapping Prozess eine rudimentäre textuelle Auswertung der Daten des Board-Systems Android-Hilfe.de implementiert. Dabei werden mit Hilfe der im vorigen Kapitel beschriebenen Methoden des Natural Language Processing Terme extrahiert, die für die Verarbeitung zu einer Taxonomie oder für die Verschlagwortung der einzelnen Beiträge in Frage kommen könnten. Auf Grund der hohen Komplexität der maschinellen Auswertung von natürlich sprachlichen Dokumenten entsteht eine hohe Laufzeit bei dieser Komponente. Die Analyse des kompletten Datensatzes des Forschungsgegenstands nimmt ca. eineinhalb Stunden ein. Neben der immensen Rechenzeit traten wegen dem verminderten Sprachniveaus auf Diskussionsplattformen mehrere Probleme auf. Die von Benutzern erstellten Beiträge sind gekennzeichnet durch häufiges Vertippen, falschem Satzbau und fehlerhafter Rechtschreibung. Dies erschwert die Syntaxanalyse zusätzlich und führt zu Fehlverhalten des Parsers bei der Sprachverarbeitung. Eine mögliche Lösung wäre, vor der textuellen Analyse eine automatische Rechtschreibprüfung durchzuführen. Ein weiteres Problem entstand durch fehlende Satzzeichen oder lange Schachtelsätze. Dadurch ergaben sich Satzlängen von über 200 Wörtern, was zu einer *OutOfMemoryException* beim Stanford Parser führte. Die einzige Lösung, einem Programmabbruch durch diese Problematik entgegenzuwirken, war die maximale Satzlänge auf 100 Zeichen zu begrenzen und einen geringen Datenverlust in Kauf zu nehmen. Die extrahierten Terme wurden in zwei XML-Dokumenten gespeichert, eines mit der Zuordnung der Wörter zu dem jeweiligen Beitrag und das andere mit allen Ausdrücken in einer Rangliste. In beiden Fällen wurden die Terme nach absteigender Relevanz angeordnet. Für die Bewertung der Ergebnisse wurden die erstellten XML-Dokumente stichprobenhaft untersucht. Nachfolgend wird ein Beispiel für die extrahierten Wörter eines Themas und deren Relevanzbewertung gegeben:

```
 1 <boardTopic>
 2 <topicName>version 3.0</topicName>
 3 <nameEntities>
 4 <word termFrequence="1" termWeight="0.014441">Android</word>
 5 </nameEntities>
 6 <normalNoun>
 7 <word termFrequence="5" termWeight="0.282089">Version</word>
 8 <word termFrequence="2" termWeight="0.171964">Tablet</word>
 9 <word termFrequence="1" termWeight="0.11503">Honeycomb</word>
10 <word termFrequence="1" termWeight="0.11503">Android-Betriebssystems</
       word>
11 <word termFrequence="1" termWeight="0.11503">Android-Smartphones</word>
12 <word termFrequence="1" termWeight="0.11503">löschen</word>
13 <word termFrequence="2" termWeight="0.086643">Google</word>
14 <word termFrequence="1" termWeight="0.085982">Smartphone</word>
15 <word termFrequence="1" termWeight="0.059025">Lächeln</word>
16 <word termFrequence="1" termWeight="0.028881">Zitat</word>
```

```
17 </normalNoun>
18 <foreignLanguage>
19 <word termFrequence="5" termWeight="0.282089">version</word>
20 <word termFrequence="2" termWeight="0.20118">tablets</word>
21 </foreignLanguage>
22 </boardTopic>
```

In diesem Beitrag wird die Frage gestellt, ob Android 3.0 nur für Tablets oder auch für Smartphones nutzbar ist. Hier hat die Relevanzbewertung gut funktioniert und die Terme könnten auch für eine Verschlagwortung verwendet werden. Auch für den Aufbau einer Taxonomie würden sich diese eignen. Die Untersuchungen haben jedoch ergeben, dass vor allem bei längeren Diskussionen innerhalb eines Themas eine Nutzung der Terme nur erschwert möglich ist, da einerseits relevante Ausdrücke erst in der Mitte der Rangliste zu finden sind und andererseits zu viele Terme vorhanden sind. Die Bewertung der Wörter hinsichtlich der Relevanz könnte durch Modifizierung der TF-IDF Formel verbessert werden. Außerdem sollte für die Domäne Android App-Entwicklung ein Trainingskorpus erstellt werden, um bessere Ergebnisse beim Natural Language Processing zu erhalten. Unter den extrahierten Wörtern sind jedoch eine große Anzahl zutreffender Konzepte für die Applikationsentwicklung unter Android enthalten, die nach weiteren Aufbereitungsschritten einen Experten bei der Entwicklung einer Taxonomie in diesem Bereich unterstützen könnten. Dadurch lässt sich feststellen, dass sich das untersuchte Board-System als Wissensquelle für Ontology Learning durchaus eignet, wobei noch weitere Untersuchung hinsichtlich der Aufbereitung der textuellen Daten vorgenommen werden müssen.

Kapitel 7

Conclusio

7.1 Zusammenfassung

Im Zuge der Weiterentwicklung des World Wide Webs zum Semantic Web ist die semantische Auszeichnung von bestehenden Wissens- und Datenquellen unvermeidbar. Die Verknüpfung von Konzepten und ihrer Bedeutung in verschiedenen Kontexten erfolgt zumeist in einer Ontologie. Die manuelle Erstellung einer Ontologie ist jedoch ein zeitaufwändiger, komplexer und fehleranfälliger Prozess. Im Rahmen dieser Arbeit wurden Methoden beschrieben, die verschiedene Wissensquellen automatisch zu einer Ontologie verarbeiten. Dieser Vorgang wird unter anderem mit Methoden aus der Künstlichen Intelligenz vorgenommen und ist auf Grund dessen mit einer hohen Komplexität verbunden. Die Heterogenität der Datenquellen ist dabei von besonderer Relevanz, da sich dadurch einerseits die Methoden unterscheiden. Andererseits ist es wichtig, verschiedene Formate, die Wissen enthalten, zu unterstützen. Dabei ist es zudem notwendig, potenzielle Wissenssammlungen ausfindig zu machen und Methoden dafür zu entwickeln. An diesem Punkt setzt die im Zuge dieser Arbeit entwickelte Methodik an. Es wurde untersucht, ob sich ein Board-System einer bestimmten Domäne als Wissensquelle für Ontology Learning eignet. Dabei entstand eine Software, die die Daten des Board-Systems Android-Hilfe.de sammelt und zu einer Ontologie weiterverarbeitet. Die erstellte Ontologie dient jedoch nicht der Beschreibung der Domäne Android App-Entwicklung, sondern als Basis für Verbesserungen hinsichtlich der Usability des Board-Systems. Für die Entwicklung einer Taxonomie wurden Terme aus den textuellen Daten der Themen extrahiert, die sich zur Beschreibung dieser eignen. Die implementierte Methodik bietet dafür grundlegend die Funktionalität. An dieser Stelle steckt jedoch noch großes Potential für Verbesserung, da sich die Verarbeitung der natürlich sprachlichen Daten als sehr komplex herausgestellt hat. Aus den Ergebnissen lässt sich dennoch schließen, dass sich Board-Systeme als Wissensquelle zum Aufbau einer Taxonomie mit Erweiterung zu einer Ontologie durchaus

eignen.

7.2 Ausblick

Die entwickelte Methodik für die Erfassung von Board-Daten und die Weiterverarbeitung zu einer Ontologie, sowie der Extraktion von relevanten Termen bietet eine gute Grundlage für zukünftige Arbeiten. Auf Basis der erstellten Ontologie lassen sich die Daten des Board-Systems weiter semantisch aufbereiten. Dabei wäre es denkbar eine semantische Suche zu ermöglichen, indem beispielsweise ähnliche Beiträge miteinander verknüpft werden und dem Suchenden vorgeschlagen werden. Außerdem könnten anhand der Benutzerdaten Experten für verschiedene Gebiete bestimmt werden, die für die Beantwortung neuer Themen automatisch vorgeschlagen werden können oder eine Benachrichtigung bekommen, wenn eine Frage zu dem jeweiligen Fachgebiet gestellt wurde. Die angewandte Methodik zum Übertragen der Daten aus der erstellten XML-Struktur bietet grundsätzlich die Möglichkeit XML-Dokumente mit allgemeiner Struktur zu einer Ontologie zu verarbeiten, was zusätzlich implementiert werden könnte. Großes Verbesserungspotential liegt in der Termextraktion. Die Bestimmung der Relevanz ausschließlich anhand der TF-IDF Formel vorzunehmen ist nicht ausreichend. Es müsste folglich ein Trainingskorpus für die Domäne Applikationsentwicklung für das Betriebssystem Android erstellt werden, um die Extrahierung der Terme schon bei der Syntaxanalyse einzuschränken.

7.3 Fazit

Die semantische Auszeichnung von Daten im Zuge des Semantic Webs bietet ein großes Potential für die Verbesserung bestehender Prozesse und wird zukünftig eine bessere User Experience bei Web-Applikationen oder auch Desktoplösungen ermöglichen. Die Forschung hinsichtlich des automatischen Erstellens von den dazu benötigten Ontologien ist zwar weit fortgeschritten, jedoch ist an eine kommerzielle Nutzung noch nicht zu denken. Durch diese Arbeit konnte ein tiefer Einblick in das Feld Ontology Learning gewonnen werden. Die Entwicklung der Software hat es ermöglicht, eine große Anzahl an Frameworks kennenzulernen, sowie Erfahrungen mit den Schwierigkeiten und der hohen Komplexität bei der Verarbeitung von natürlich sprachlichen Daten zu sammeln. Außerdem wurde deutlich, dass eine Entwicklung im Bereich Ontology Learning ein hohes Maß an Basiswissen voraussetzt, um mit dem eigentlichen Entwicklungsprozess beginnen zu können. Die Verarbeitung der Board-Daten zu einer Ontologie konnte erfolgreich vorgenommen werden. Die Extraktion der relevanten Terme konnte nur rudimentär implementiert werden, jedoch konnten daraus wichtige Schlüsse für diese Arbeit gezogen werden.

Anhang A

Regelsystem für die Wissensquelle Datenbanken

In Abschnitt 3.1.1 wird eine Methodik beschrieben, um aus der Wissensquelle Datenbank eine Ontologie zu erstellen. Die nachfolgenden Ausführungen dienen der Erläuterungen der für dieses System bentötigten Regeln, beginnend mit Definition der verwendeten mathematischen Zusammenhänge, entnommen aus [13]:

- Jede Relation R_i ist einer Tabelle zugeordnet
- Die Spalten einer Tabelle werden als Attribute A bezeichnet
- Die Funktion $attr(R_i)$ beinhaltet alle Attribute in einer spezifischen Relation R_i
- Die Funktion $dom(A_i)$ beinhaltet den Wertebereich von Attributen A_i, wobei gilt $A_i \in A$
- Die Funktion $pkey(R_i)$ beschreibt die Primärschlüssel einer Relation R_i, wobei $pkey(R_i) \subseteq attr(R_i)$ gelten muss.
- Die Funktion $fkey(R_i)$ beschreibt die Fremdschlüssel einer Relation R_i, wobei $pkey(R_i) \subseteq attr(R_i)$ gelten muss.

  For relations R_i and R_j in database, supposed that $A_i \subseteq attr(R_i)$ and $A_j \subseteq attr(R_j)$, $t_i(A_i)$ expresses the values of tuple t_i in attributes A_i, and $t_j(A_j)$ expresses the values of tuple t_j in attributes A_j. For each$t_i(A_i)$ in R_i, if in R_j there exists $t_i(A_i) = t_j(A_j)$, A_i and A_j are called inclusion dependency, denoted as $R_i(A_i) \subseteq R_j(A_j)$. [13]

Die oben angeführte Definition beschreibt die Einschlussabhängigkeit zwischen zwei Attributen, die gilt wenn für jede Wertemenge [1] in einer Relation in einer anderen Relation die gleiche Wertemenge existiert. Diese Ein-

[1] Der Begriff Wertemenge bezeichnet hier den mathematischen Begriff Tupel und wurde für eine bessere Verständlichkeit gewählt.

schlussabhängigkeit wird damit bezeichnet, dass die eine Relation Teilmenge der anderen Relation ist.

> For Relations R_i and R_j in database, supposed that $A_i \subseteq attr(R_i)$ and $A_j \subseteq attr(R_j)$, if there exist $R_i(A_i) \subseteq R_j(A_j)$ and $R_j(A_j) \subseteq R_i(A_i)$, A_i and A_j are called equivalence, denoted as $R_i(A_i) = R_j(A_j)$. [13]

Diese Definition bezieht sich auf die Gleichwertigkeit zweier Attribute. Die Äquivalenz gilt wenn zwei Relationen jeweils Teilmenge der anderen sind, vorausgesetzt die beiden Attribute sind jeweils Teilmenge der jeweiligen Relation. Diese Entsprechung zweier Attribute wird damit bezeichnet, dass eine Relation, in Abhängigkeit des Attributes, gleich der anderen Relation ist ($R_i(A_i) = R_j(A_j)$). Nachdem das mathematische Vokabular und die benötigten Definitionen spezifiziert wurde, kann nachfolgend das Regelset für die automatische Erstellung einer Ontologie anhand einer relationalen Datenbank beschrieben werden. In den Ausführungen hinsichtlich der Beschreibungssprache OWL wurde deutlich, dass die wichtigsten Elemente zur Erstellung einer OWL-Ontologie Klassen, Individuen, Rollen und Restriktion sind. Li et. al haben davon ausgehend Regeln entwickelt, um diese Elemente automatisch anhand der Relationen und Attribute der Datenbank zu erstellen. Die ersten beiden Regeln beziehen sich auf das Erstellen von Klassen.

Regeln zur Erstellung von Klassen

> Rule 1. For relations $R_1, R_2, \ldots, R_i$ in database, supposed that $P_1 = pkey(R_1), P_2 = pkey(R_2), \ldots, P_i = pkey(R_i)$ if $R_1(P_1) = R_2(P_2) = \ldots = R_i(P_i)$, then information spread across $R_1, R_2, \ldots R_i$ should be integrated into a ontological class c_i. [13]

Diese Regel besagt, dass, wenn der Primärschlüssel einer Relation gleich dem Primärschlüssel einer oder mehrerer anderer Relationen ist, dann können die Daten in den Attributen in einer OWL-Klasse abgebildet werden.

> Rule 2. An ontological class C_i can be created based on the Relation R_i, if there does not exist a relation that can be integrated with R_i, that is to say, Rule 1 cannot be satisfied, and one of the following conditions can be satisfied:
>
> **(i)** $\mid pkey(R_i) \mid = 1$
> **(ii)** $\mid pkey(R_i) \mid > 1$, and there exists A_i, where $A_i \in pkey(R_i)$ and $A_i \notin fkey(R_i)$. [13]

Die erste Prämisse, damit Regel zwei greift, ist, dass Regel eins nicht erfüllt wird. Ist diese Voraussetzung erfüllt, können zwei Fälle auftreten,

damit eine Klasse erstellt wird. Entweder die Anzahl der Primärschlüssel ist eins oder die Anzahl der Primärschlüssel ist größer als eins und es existiert ein Attribut, das Primärschlüssel ist und nicht gleichzeitig Fremdschlüssel. Zur besseren Verständlichkeit der Regeln zur Erstellung von OWL-Klassen dient das Beispiel in Tab. A.1.

Tabelle A.1: Datenbankschema [13].

Relation	*Primary Key*	*Foreign Key*
Student (StuID int, StuName string, Hometown int)	StuID	Hometown referring to CityID in City
PhdStudent(StuID int, Year data)	StuID	StuID referring to StuID in Student
City(CityID int, CityName string)	CityID	N
Department(DepID int, DepName string)	DepID	N
Study(StuID int, DepID int)	StuID, DepID	StuID referring to StuID in Student, DepID referring to DepID in Department
Staff(LectID int, Room int)	LectID	N
Staff-Default(LectID int, Address string)	LectID	LectID referring to LectID in Staff

Im ersten Schritt kann bei diesem Datenbankschema Regel eins angewandt werden, da der Primärschlüssel der Relation Staff gleich dem in Staff-Default ist. Somit können die in diesen Tabellen enthaltenen Daten in der OWL-Klasse Staff abgelegt werden. Die restlichen Relationen (PHDStudent, City, Department , Study) werden als Klassen in die Ontologie übernommen. In diesem Fall greift Regel zwei, da Regel eins nicht erfüllt ist und die Relationen jeweils nur einen Primärschlüssel enthalten [13]. Die nächsten Regeln dienen dazu, abstrakte Rollen in der Ontologie zu erstellen.

Regeln zur Erstellung von abstraken Rollen

> Rule 3. For Relations R_i and R_j, if $R_i(A_i) \subseteq R_j(A_j)$ and $A_i \not\subset pkey(R_i)$ are satisfied, then an object property P can be created based on A_i. Suppose that the clases corresponding to R_i and R_j are C_i and C_j respectively, the domain and range of P are C_i and C_j. [13]

Durch Regel drei wird beschrieben, dass basierend auf einem Attribut einer Relation eine abstrakte Rolle erstellt werden kann, wenn die Relation echte Teilmenge einer anderen Relation ist und das Attribut nicht der Primärschlüssel der Relation ist. Die zu den Relationen gehörigen Klassen werden dann als Domäne bzw. Wertebereich verwendet.

> Rule 4. For Relations R_i and R_j, two ontological objects property "has-part" and "is-part-of" can be created, if the two conditions are satisfied: (i) $\mid pkey(R_i) \mid > 1$ (ii) $fkey(R_i) \subset pkey(R_i)$, where $fkey(R_i)$ referring to R_j. [13]

Regel vier besagt, dass zwei inverse abstrakte Rollen für zwei Relationen erstellt werden können, wenn die erste Relation mehr als einen Primärschlüssel besitzt und der Fremdschlüssel dieser Relation eine echte Teilmenge des Primärschlüssels ist, wobei der Fremdschlüssel auf die zweite Relation zeigt.

> Rule 5. For Relations R_i, R_j and R_k, if $A_i = pkey(R_i), A_j = pkey(R_j), A_i \cup A_j = fkey(R_k)$ and $A_i \cap A_j =$, then two object properties P_j' and P_j'' can be created based on the semantics of R_k. Suppose that the classes corresponding to R_i and R_j are C_i and C_j and the domain and range of P_j'' are C_j and C_i. P_j' and P_j'' are two inverse properties. [13]

Im Fall von Regel fünf werden drei Relationen betrachtet. Sollte eine Relationen aus zwei Fremdschlüssel zu zwei anderen Relationen bestehen, dann kann basierend auf diesen Fremdschlüsseln zwei abstrakte Rollen erstellt werden. Im Beispiel von Tab. A.1 wäre das für die Relation Study der Fall, woraufhin zwei object properties belong-to und has-student erstellt werden können.

> Rule 6. For Relation $R_1, R_2, \ldots, R_i$ and R_j, if $A_1 = pkey(R_1), A_2 = pkey(R_2), \ldots, A_i = pkey(R_i), A_1 \cup A_2 \cup \ldots \cup A_i = fkey(R_j)$ and $A_1 \cap A_2 \cap \ldots \cap A_i =$, then object properties $P_i^1, P_i^2, \ldots, P_i^i * (i-1)$ can be created. The semantics of $P_i^1, P_i^2, \ldots, P_i^i * (i-1)$ are based on the decomposition of n-ary relationship provided by R_j. [13]

Regel sechs stellt eine Erweiterung von Regel fünf dar und kann somit auf n-äre Relationen angewandt werden. Ansonsten ist der Inhalt analog zu Regel fünf.

> Rule 7. For an ontological class C_i and the datatype properties set of C_i denoted as $DP(C_i)$, if C_i is corresponding to relations $R_1, R_2, \ldots, R_i$ in database, then for every attribute in $R_1, R_2, \ldots, R_i$, if it cannot be used to create object property by using Rule 3, then it can be used to create datatype property of C_i. The domain and range of each property P_i are C_i and $dom(A_i)$ respectively, where $P_i \in DP(C_i)$ and $A_i \in attr(R_i)$. [13]

Regeln zur Erstellung von konkreten Rollen

Regel sieben legt die Erstellung von konkreten Rollen in der Ontologie fest. Es wird definiert, dass alle Relationen die nicht als abstrakte Rollen verwendet werden können, als konkrete Rollen in der Ontologie angelegt werden. Regel acht dient dem Aufbau einer hierarchischen Struktur und besagt, dass, sollte Regel eins nicht greifen und eine Relation Teilmenge einer zweiten Relation sein, dann ist diese Subklasse der korrespondierenden Relation.

> Rule 8. For relation R_i and R_j, supposed that $P_i = pkey(R_i), P_j = pkey(R_j)$, if Rule 1 does not applied and $Ri(Pi) \subset R_j(P_j)$ is satisfied, then the class/property corresponding to R_i is a subclass/subproperty of the class/property corresponding to R_j. [13]

Die nächsten drei Regeln beziehen sich auf die Kardinalität der Rollen.

> Rule 9. For relation R_i and $A_i \in attr(R_i)$, if $A_i = pkey(R_i)$ or $A_i = fkey(R_i)$, then the minCardinality and maxCardinality of the property P_i corresponding to A_i is 1.
> Rule 10. For relation R_i and $A_i \in attr(R_i)$, if A_i is declared as NOT NULL, the minCardinality of the property P_i corresponding to A_i is 1.
> Rule 11. For relation R_i, and $A_i \in attr(R_i)$, if A_i is declared as UNIQUE, the maxCardinality of the property P_i corresponding to A_i is 1. [13]

Regeln zur Erstellung von Individuen

Die letzte Regel bezieht sich auf das Anreichern der angelegten Klassen mit Individuen. Es wird festgelegt, dass alle Wertemengen einer Relation als Individuen in der korrespondierenden Klasse angelegt werden können.

> Rule 12. For a ontological class C_i, if C_i is corresponding to relations R_1, R_2, ..., R_i in database, then every tuple $t_i, t_i \in R_1 \bowtie R_2 \bowtie \ldots \bowtie R_i$, can be a instance of C_i. [13]

Anhang B

Inhalt der CD-ROM/DVD

Format: DVD-ROM, Single Layer, DVD-R-Format

B.1 Masterarbeit (PDF)

Pfad: /

Masterarbeit.pdf . . . . *BoardToOntology* – Ontology Learning auf Basis der Wissensquelle Online-Diskusionsplattform

Arbeiten mit BoardToOntology.pdf Anleitung zur Benutzung der entwickelten Software

B.2 Online-Quellen (PDF)

Pfad: /Online-Quellen

09_Clustering_Method_for_Verb_Frames_and_Ontology.pdf A Corpus-based Conceptual Clustering Method for Verb Frames and Ontology Acquisition

11_Ontology_Learning_from_Textual_Web_Documents Ontology Learning from Textual Web Documents

12_lehrbuch_information_retrieval.pdf Information Retrieval 1 Grundlagen, Modelle und Anwendungen

16_extraktion_von_ontologien_aus_natürlichsprachlichen_texten.pdf Extraktion von Ontologien aus natürlichsprachlichen Texten

35_owl_overview.pdf . Überblick der OWL Web Ontology Language

36_owl_guide.pdf . . . Erklärungen zu der OWL Web Ontology Language

37_introduction_xml_schema.pdf Erläuterungen zum XML-Schema

38_introduction_dtd.pdf Erläuterungen zum XML-DTD
40_wordnet.pdf . . . . Definition und Grundstruktur des Wordnets
42_steinmann_forum.pdf Artikel über Foren
43_android_hilfe_forum_nutzen.pdf Benutzungsanleitung des Board-Systems Android-Hilfe.de
44_forenundboards.pdf Vergleich zwischen Foren und Board-Systemen
46_composite_pattern.pdf Erläuterungen zum Composite Pattern
47_crawler4j.pdf . . . . Erläuterungen zum verwendeten Crawler
48_crawl_depth.pdf . . Definition der Crawlertiefe
49_jsoup_cookbook.pdf Verwendung von jsoup
50_java_sun_dao_pattern Erläuterungen zum Oracle Sun DAO-Pattern
51_opennlp_manual.pdf Anleitung openNLP
52_stts.pdf . . . . . Stuttgart-Tübinger-Tagset
53_snowball_stemming_table.pdf Beispiele für gestemmte Begriffe

B.3 Masterprojekt (Java)

Pfad: /Masterprojekt

/src . . . . . Source-Kode des Projekts
/lib . . . . . Bibliotheken für das Projekt
/ontology . . . . . Ontologie in die geschrieben wird
/data . . . . . Daten die für die Verarbeitung benötigt werden bzw. erstellt werden

B.4 Beispieldaten (XML, OWL)

Pfad: /Exampledata

CandidateTerms/ . . . . Exemplarischer Datensatz der Termextraktion
CrawledBoardData/ . . Exemplarischer Datensatz für die gesammelten Daten des Forschungsgegenstands Android-Hilfe.de
Ontologie/ . . . . . Exemplarische Ontologie
OverallRankedTerms/ . Exemplarischer Datensatz der Termextraktion

Quellenverzeichnis

Literatur

[1] Nathalie Aussenac-Gilles und Mouna Kamel. „Ontology Learning by Analyzing XML Document Structure and Content“. In: *Proceedings of the International Conference on Knowledge Engineering and Ontology Development*. (Madeira, Portugal). Hrsg. von Jan L. G. Dietz. Setubal, Portugal: INSTICC Press, Okt. 2009, S. 159–165.

[2] Brigitte Biébow, Sylvie Szulman und Av. Clément. „TERMINAE: A Linguistics-Based Tool for the Building of a Domain Ontology“. In: *Knowledge Acquisition, Modeling and Management*. Hrsg. von Dieter Fensel und Rudi Studer. Berlin & Heidelberg: Springer, 1999. Kap. 4, S. 49–66.

[3] Hannes Bohring und Sören Auer. „Mapping XML to OWL Ontologies“. In: *Marktplatz Internet: von E-Learning bis E-Payment, 13. Leipziger Informatik-Tage, LIT 2005, 21.-23. September 2005, Leipzig*. (Leipzig, Germany). Hrsg. von Klaus P. Jantke, Klaus-Peter Fähnrich und Wolfgang S. Wittig. Bonn: Gesellschaft für Informatik e.V., Sep. 2005, S. 147–156.

[4] Paul Buitelaar, Philipp Cimiano und Berenike Loos. *Proceedings of the 2nd Workshop on Ontology Learning and Population: Bridging the Gap between Text and Knowledge*. Sydney, Australia: Association for Computational Linguistics, 2006.

[5] K.U. Carstensen u. a. *Computerlinguistik und Sprachtechnologie: Eine Einführung*. 3. Aufl. Heidelberg: Spektrum Akademischer Verlag, 2010.

[6] Philipp Cimiano. *Ontology Learning and Population from Text: Algorithms, Evaluation and Applications*. New York: Springer-Verlag, 2006.

[7] Christophe Cruz und Christophe Nicolle. „Ontology Enrichment and Automatic Population From XML Data.“ In: *Proceedings of the 4th International VLDB Workshop on Ontology-based Techniques for DataBases in Information Systems and Knowledge Systems*. (Auckland, New Zealand). Hrsg. von Martine Collard. Berlin & Heidelberg: Springer, Aug. 2008, S. 17–20.

[8] Hasan Davulcu, Srinivas Vadrevu und Saravanakumar Nagarajan. „OntoMiner: Bootstrapping and Populating Ontologies from Domain Specific Websites“. In: *Proceedings of the First International Workshop on Semantic Web and Databases*. (Berlin, Germany). Hrsg. von Isabel F. Cruz u. a. Berlin: Humboldt-Universität, Sep. 2003, S. 24–33.

[9] Matthias Ferdinand, Christian Zirpins und D. Trastour. „Lifting XML Schema to OWL“. In: *Web Engineering - 4th International Conference, ICWE 2004, Munich, Germany, July 26-30, 2004, Proceedings*. (Munich, Germany). Hrsg. von Nora Koch, Piero Fraternali und Martin Wirsing. Berlin & Heidelberg: Springer, Juli 2004, S. 354–358.

[10] P. Hitzler u. a. *Semantic Web: Grundlagen*. Berlin & Heidelberg: Springer, 2007.

[11] He Hu und Da-You Liu. „Learning OWL ontologies from free texts“. In: *Proceedings of 2004 International Conference on Machine Learning and Cybernetics*. (Shanghai, China). New York, USA: Institute of Electrical & Electronics Enginee (IEEE), Aug. 2004, S. 1233–1237.

[12] K. Kessel und S. Reimann. *Basiswissen deutsche Gegenwartssprache*. 3. Aufl. Stuttgart: UTB, 2010.

[13] Man Li, Xiao-Yong Du und Shan Wang. „Learning ontology from relational database“. In: *Proceedings of International Conference on Machine Learning and Cybernetics*. (Guangzhou, China). Hrsg. von Daniel S. Yeung u. a. Berlin & Heidelberg: Springer, Aug. 2005, S. 3410–3415.

[14] Hiep Phuc Luong, Susan Gauch und Qiang Wang. „Ontology Learning Through Focused Crawling and Information Extraction“. In: *Proceedings of the 2009 International Conference on Knowledge and Systems Engineering*. (Hanoi). Hrsg. von Ngoc Thanh Nguyen u. a. Washington, DC, USA: IEEE Computer Society, Nov. 2009, S. 106–112.

[15] Alexander Mädche und Steffen Staab. „Semi-automatic Engineering of Ontologies from Text“. In: *Proceedings of the 12th International Conference on Software Engineering and Knowledge Engineering*. (Chicago, USA). Hrsg. von D. Cooke und R. et al Mittermeir. Skokie, USA: Knowledge System Institute, Juli 2000, S. 231–239.

[16] Alexander Maedche und Steffen Staab. „Ontology Learning for the Semantic Web“. In: *IEEE Intelligent Systems* 16.2 (2001), S. 72–79.

[17] Günter Neumann und Jakub Piskorski. „A shallow text processing core engine“. In: *Computational Intelligence* 18.3 (2002), S. 451–476.

[18] Peter Øhrstrøm, Jan Andersen und Henrik Schärfe. „What has happened to ontology“. In: *Proceedings of the 13th international conference on Conceptual Structures: common Semantics for Sharing Knowledge*. (Kassel, Germany). Hrsg. von Frithjof Dau, Marie-Laure Mugnier und

Gerd Stumme. Berlin & Heidelberg: Springer-Verlag, Juli 2005, S. 425–438.

[19] Borys Omelayenko. „Learning of Ontologies for the Web: the Analysis of Existent Approaches“. In: *Proceedings of the First International Workshop on Web Dynamics, in conjunction with the 8th International Conference on Database Theory*. (Londen, UK). Hrsg. von Mark Levene und Alexandra Poulovassilis. London: Birkbeck University – Department of Computer Science und Information Systems, Jan. 2001, S. 16–25.

[20] Anna Rafferty und Christopher D. Manning. „Parsing Three German Treebanks: Lexicalized and Unlexicalized Baselines“. In: *Proceedings of the Workshop on Parsing German*. (Columbus, Ohio). Hrsg. von Sandra Kübler und Gerald Penn. Stroudsburg, PA, USA: Association for Computational Linguistics, Juni 2008, S. 40–46.

[21] Toni Rodrigues, Pedro Rosa und Jorge Cardoso. „Mapping XML to Existing OWL Ontologies“. In: *International Conference WWW/Internet 2006*. (Murcia, Spain). Hrsg. von Pedro Isaías, Miguel Baptista Nunes und Inmaculada J. Martínez. Lissabon: IADIS Press, Okt. 2006, S. 72–77.

[22] David Sànchez und Antonio Moreno. „Creating Ontologies from Web documents“. In: *Recent Advances In Artificial Intelligence Research And Development (Frontiers in Artificial Intelligence and Applications)*. Hrsg. von Jordi Vitriá, Petia Radeva und Isabel Aguiló. Amsterdam: IOS Press, 2004. Kap. 2, S. 11–26.

[23] S. Staab und R. Studer. *Handbook on Ontologies*. 2. Aufl. Berlin & Heidelberg: Springer, 2009.

[24] Steffen Staab und Alexander Mädche. „Axioms are Objects, too – Ontology Engineering beyond the Modeling of Concepts and Relations“. In: *Proceedings of the Workshop on Applications of Ontologies and Problem-solving Methods, 14th European Conference on Artificial Intelligence ECAI*. (Berlin, Germany). Hrsg. von Werner Horn. Amsterdam: IOS Press, Aug. 2000, S. 321–325.

[25] Jan Strickmann. „Analysemethoden zur Bewertung von Entwicklungsprojekten: ein integriertes semantisches Modell von Projekt- und Produktdaten zur Bewertung der Entwicklungsleistung im Projektcontrolling“. Diss. Oldenburg: Carl von Ossietzky Universität, Fakultät für Informatik, Wirtschafts- und Rechtswissenschaften, Okt. 2008.

[26] York Sure u. a. „OntoEdit: Collaborative Ontology Development for the Semantic Web“. In: *International Semantic Web Conference*. (Chia, Italia). Hrsg. von Ian Horrocks und James A. Hendler. Berlin & Heidelberg: Springer, Juni 2002, S. 221–235.

[27] Pham Thi Thu Thuy, Young-Koo Lee und SungYoung Lee. „DTD2OWL: automatic transforming XML documents into OWL ontology“. In: *Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human*. (Seoul, Korea). Hrsg. von Sungwon Sohn, Ling Chen und Soonwook Hwang et al. New York, USA: ACM, Nov. 2009, S. 125–131.

[28] Takahira Yamaguchi. „Acquiring Conceptual Relationships from Domain-Specific Texts“. In: *Workshop on Ontology Learning*. (Seattle, USA). Hrsg. von Alexander Maedche u. a. Aachen: CEUR-WS.org, Aug. 2001, S. 13–18.

[29] Jiang-Ming Yang u. a. „Incorporating site-level knowledge for incremental crawling of web forums: a list-wise strategy“. In: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. (Paris). Hrsg. von John F. Elder IV u. a. New York: ACM, Juni 2009, S. 1375–1384.

[30] Lina Zhou. „Ontology learning: state of the art and open issues“. In: *Inf. Technol. and Management* 8.3 (2007), S. 241–252.

Online-Quellen

[31] Kopie auf DVD (Datei 16_extraktion_von_ontologien_aus_natürlichsprachlichen_texten). URL: http://logopolis.de/docs/ontomining.pdf (besucht am 13.07.2012).

[32] Kopie auf DVD (Datei 35_owl_overview.pdf). URL: http://www.w3.org/TR/owl-features/ (besucht am 22.04.2012).

[33] Kopie auf DVD (Datei 36_owl_guide.pdf). URL: http://www.w3.org/TR/owl-guide/ (besucht am 22.04.2012).

[34] Kopie auf DVD (Datei 11_Ontology_Learning_from_Textual_Web_Documents). URL: http://www.claes.sci.eg/coe_wm/Infos_08NLP_17_P113-120.pdf (besucht am 13.07.2012).

[35] Kopie auf DVD (Datei 37_introduction_xml_schema.pdf). URL: http://www.w3schools.com/schema/schema_intro.asp (besucht am 25.04.2012).

[36] Kopie auf DVD (Datei 38_introduction_dtd.pdf). URL: http://www.w3schools.com/dtd/dtd_intro.asp (besucht am 25.04.2012).

[37] URL: http://gate.ac.uk/ (besucht am 25.04.2012).

[38] Kopie auf DVD (Datei 40_wordnet.pdf). URL: http://wordnet.princeton.edu/ (besucht am 28.04.2012).

[39] Kopie auf DVD (Datei 09_Clustering_Method_for_Verb_Frames_and_Ontology.pdf). URL: http://alice.nc.huji.ac.il/~dmitry/Reading/People/LanguageDeveloplment/FaureDavid/CorpusBasedClusteringForVebFramecAndOntologyAcquisition.pdf (besucht am 13.07.2012).

[40] URL: http://www.isquare.de/web/products.php?p=smartsearchmenu\&l=en (besucht am 21.05.2012).

[41] Kopie auf DVD (Datei 42_steinmann_forum.pdf). URL: http://aktuell.de.selfhtml.org/artikel/projekt/forum/index.htm (besucht am 21.05.2012).

[42] Kopie auf DVD (Datei 43_android_hilfe_forum_nutzen.pdf). URL: http://www.android-hilfe.de/faq.php?faq=vb3_board_usage#faq_vb3_tags (besucht am 21.05.2012).

[43] Kopie auf DVD (Datei 44_forenundboards.pdf). URL: http://aktuell.de.selfhtml.org/artikel/gedanken/foren-boards/ (besucht am 21.05.2012).

[44] URL: http://www.vbulletin-germany.com/ (besucht am 21.05.2012).

[45] Kopie auf DVD (Datei 46_composite_pattern.pdf). URL: http://www.philipphauer.de/study/se/design-pattern/composite.php (besucht am 22.05.2012).

[46] Kopie auf DVD (Datei 12_lehrbuch_information_retrieval.pdf). URL: http://www.uni-bamberg.de/fileadmin/uni/fakultaeten/wiai_lehrstuehle/medieninformatik/Dateien/Publikationen/2008/henrich-ir1-1.2.pdf (besucht am 13.07.2012).

[47] Kopie auf DVD (Datei 47_crawler4j.pdf). URL: http://code.google.com/p/crawler4j/ (besucht am 22.04.2012).

[48] Kopie auf DVD (Datei 48_crawl_depth.pdf). URL: http://www.seoengine.com/crawl-depth.htm (besucht am 03.06.2012).

[49] Kopie auf DVD (Datei 49_jsoup_cookbook.pdf). URL: http://jsoup.org/ (besucht am 03.06.2012).

[50] Kopie auf DVD (Datei 50_java_sun_dao_pattern.pdf). URL: http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html (besucht am 06.06.2012).

[51] Kopie auf DVD (Datei 51_opennlp_manual.pdf). URL: http://opennlp.apache.org/ (besucht am 03.06.2012).

[52] Kopie auf DVD (Datei 52_stts.pdf). URL: http://www.fi.muni.cz/~xnemcik/nlp/sarrebrugge/handout.pdf (besucht am 10.06.2012).

[53] Kopie auf DVD (Datei 53_snowball_stemming_table.pdf). URL: http://snowball.tartarus.org/algorithms/german/diffs.txt (besucht am 10.06.2012).