Semiautomatische Erstellung semantischer Netze

Dissertation

zur

Erlangung des Doktorgrades (Dr. rer. nat.)

der

Mathematisch-Naturwissenschaftlichen Fakultät

der

Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

Lars Bröcker

aus

Ratzeburg

Bonn, Juni 2008

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Rheinischen Friedrich-Wilhelms-Universität Bonn.

Diese Dissertation ist auf dem Hochschulschriftenserver der ULB Bonn unter `http://hss.ulb.uni-bonn.de/diss_online` elektronisch publiziert.

Erstgutachter: Prof. Dr. Armin B. Cremers

Zweitgutachter: Prof. Dr. Stefan Wrobel

Tag der Promotion: 5. Dezember 2008

Erscheinungsjahr: 2008

Zusammenfassung

Diese Dissertation beschäftigt sich mit der Erforschung von Verfahren zur semiautomatischen Erstellung von Ontologien für digital vorliegende Dokumentensammlungen, um den Prozess der semantischen Erschließung solcher Sammlungen zu erleichtern und so einen besseren Zugang zu den enthaltenen Informationen zu ermöglichen.

Auf Basis der Ergebnisse einer automatischen Eigennamenerkennung für verschiedene Arten von Eigennamen wird ein Verfahren zur unüberwachten statistischen Relationserkennung entwickelt und vorgestellt. Eine vom Nutzer ausgewählte Teilmenge der gefundenen Relationen wird anschließend automatisch zusammen mit den erkannten Eigennamen in eine Ontologie überführt, mit der die Inhalte der Sammlung repräsentiert werden können.

Diese Verfahren werden in eine Systemarchitektur eingebettet, mit der sich eine weitgehend automatisierte semantische Erschließung eines digital vorliegenden Dokumentenbestands durchführen lässt und die die Navigation in der Sammlung über eine Wiki-Schnittstelle ermöglicht. Die Architektur basiert auf Web Services, wodurch sowohl die Erweiterung des Systems vereinfacht als auch die lastabhängige Verteilung der einzelnen Komponenten ermöglicht wird.

Die drei Hauptbeiträge dieser Arbeit sind: Ein Verfahren zur automatischen Relationserkennung auf annotierten Textdokumenten, ein Verfahren zur semiautomatischen Umwandlung erkannter Relationsinstanzen in eine Ontologie sowie eine Systemarchitektur, die den gesamten Arbeitsablauf vom Import der Dokumente bis zur Web-Präsentation abdeckt.

Danksagungen

Keine wissenschaftliche Arbeit entsteht im luftleeren Raum - diese Dissertation macht da keine Ausnahme. An dieser Stelle möchte ich mich bei den Personen bedanken, ohne deren Unterstützung diese Arbeit nicht hätte geschrieben werden können.

Zuvorderst gebührt Dank Professor Armin B. Cremers, Professor Stefan Wrobel, Professor Wolfgang Hoeppner und Professor Bernhard Schröder für die Übernahme der Betreuung.

Danach möchte ich mich bei meinen Kollegen im Fraunhofer IAIS bedanken: Bei Dr. Joachim Köhler und Peter Wunderling für die mir für die Forschung eröffneten Freiräume, bei Marion Borowski, Thomas Tikwinski und Dr. Gerhard Paaß für die stete Bereitschaft zur fachlichen Diskussion und ihre inhaltlichen Anregungen und bei Dieter Strecker und Stefan Paal für ihre Unterstützung bei der Implementierung der verschiedenen Systeme. An den Arbeiten hat eine Reihe von Studierenden Teil gehabt, an dieser Stelle Dank an Andreas Bertram, Markus Birnbaum, David Greiff, Barbara Krausz, Marina Reinus, Maren Scheffel, und Kerstin Schmidt.

Desweiteren möchte ich mich bei den Projektpartnern des WIKINGER-Projekts bedanken, nämlich bei Dr. Marc Rössler und Dr. Andreas Wagner vom Lehrstuhl für Computerlinguistik der Universität Duisburg-Essen, sowie bei den Mitarbeitern der Kommission für Zeitgeschichte, Dr. Andreas Burtscheidt, Dr. Bernhard Frings, Dr. Christoph Kösters, sowie ihrem Leiter, Dr. Karl-Joseph Hummel.

Schließlich möchte ich mich bei den beiden Menschen bedanken, die wahrscheinlich den meisten Anteil an der Fertigstellung dieser Arbeit haben: Bei meiner Ehefrau Petra, die sich in den letzten drei Jahren damit arrangieren musste, dass die Wochenenden für Ontologien, Web Services und das Semantic Web verplant waren. Trotzdem hat sie mich angespornt, Erklärungsversuche arkaner Technologien ertragen und heldenhaft erste Fassungen der einzelnen Kapitel Korrektur gelesen. Schlussendlich ist da noch Johanna Elisabeth, deren Geburt unsere kleine Familie komplett gemacht hat und ein überwältigendes Incentive für die rasche Fertigstellung der Arbeit darstellte.

Allen Genannten sei herzlich gedankt – sowie all jenen, die ich im Eifer des Gefechts hier aufzuführen vergessen habe. Danke Euch/Ihnen allen!

Bonn, im Juni 2008

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einführung

Im Verlauf der letzten 15 Jahre hat das World Wide Web die Art, in der Wissen erzeugt, konsumiert und nachgefragt wird, dramatisch verändert. Im Jahr 2007 nutzten bereits mehr als 60% der Deutschen das Internet, in der Altersgruppe zwischen 14 und 29 Jahren sogar 88,1% [105] – es entwickelt sich also in der Breite der Bevölkerung zu einem neuen Informationskanal. Klassische Bewahrer von Wissen und Kultur sind von dieser Entwicklung nicht ausgenommen; immer mehr Fachbibliotheken, Sammlungen und Museen gehen den Schritt in die digitale Welt, unter anderem gefördert durch die Europäische Union, die das Projekt "European Digital Library" (kurz EDL) aufgelegt hat[1]. Zu den Beweggründen hierfür heisst es in [46]:

> The heritage of European libraries is unequalled in richness and diversity. But if it is not digitised and made accessible online, this heritage could, tomorrow, fail to fill its just place in the future geography of knowledge.

Das Projekt soll also sicherstellen, dass die Vielfalt des europäischen Kulturerbes auch im Internet angemessen repräsentiert ist. Abgesehen von den geopolitischen Erwägungen, die hinter dem Projekt stecken mögen, ist jedoch der Kern der Sache durchaus begrüßenswert: In der Tat gibt es in Europa eine sehr große Anzahl von Bibliotheken, Archiven und Museen, die äußerst umfangreiche Sammlungen mit kulturell, gesellschaftlich und wissenschaftlich interessanten Inhalten pflegen. Diese vor dem Zerfall zu retten, digital zu sichern und unter Umständen sogar im Internet verfügbar zu machen, sind Ziele, die das kulturelle Erbe Europas erhalten helfen[2].

[1] Siehe http://www.europeana.eu

[2] Gerade angesichts des tragischen Beispiels des Brands der Anna-Amalia-Bibliothek in Weimar am 2. September 2004

Wissenschaftliche Dokumentensammlungen

Auch in der Wissenschaft spielen digital verfügbare Dokumentensammlungen eine immer größere Rolle, nicht nur in den naturwissenschaftlich-technischen Disziplinen, sondern auch in den Geistes- und Gesellschaftswissenschaften. Gerade hier gibt es eine große Anzahl von Fachsammlungen und Archiven, deren Inhalte für die Forschung von großem Wert sind, deren Aufarbeitung aber unter dem bisher nur beschränkt möglichen Zugriff gelitten hat, weswegen die Deutsche Forschungsgemeinschaft seit 2006 ein spezielles Förderprogramm für die Digitalisierung wissenschaftlich interessanter Dokumentensammlungen zum Zweck der gemeinfreien Veröffentlichung unterhält, inklusive des evtl. notwendigen Erwerbs von Nationallizenzen bei solchen Dokumenten, deren Autorenschutz noch nicht ausgelaufen ist.

Allerdings sind die Anforderungen an die Recherchemöglichkeiten für die wissenschaftliche Nutzung deutlich höher als für die reine Präsentation von Sammlungsinhalten für die interessierte Öffentlichkeit. Während letztere mit einer redaktionellen Aufbereitung der Highlights einer Sammlung und einer Volltextsuchmöglichkeit im Allgemeinen ihr Informationsbedürfnis befriedigen kann, benötigt erstere einen umfassenderen Zugriff auf die Inhalte der digitalisierten Kollektionen, der nur durch deren inhaltliche Erschließung realisiert werden kann. Zwar existieren oftmals bereits so genannte “Findmittel”, diese sind jedoch üblicherweise als Hilfsmittel für ausgebildete Archivare und nicht für Fachwissenschaftler gedacht - zumal sie stark auf die lokalen Begebenheiten der analogen Sammlung abheben, sei es durch Organisation nach Räumen, Regalen oder Ähnlichem. Die Verwendung solcher Kategorisierungen ist jedoch im digitalen Umfeld nicht mehr sinnvoll, da hier keinen räumlichen Beschränkungen Rechnung getragen werden muss: Digitale Regale wachsen dynamisch, solange genug Speicherkapazität zur Verfügung gestellt werden kann, und ein digitales Dokument kann in vielen verschiedenen thematischen Regalen gleichzeitig stehen. Im gleichen Maß, in dem andere Kategorisierungen an Bedeutung verlieren, gewinnt in einer digitalen Sammlung die inhaltliche Bedeutung der Dokumente an Bedeutung.

Ontologien und das Semantic Web

Was daher benötigt wird, ist eine Repräsentation der Inhalte und Zusammenhänge von Dokumenten in Fachsammlungen, und zwar so, dass sie von einer möglichst großen Zahl räumlich verteilter Nutzer verstanden und genutzt werden kann. Um zu einer solchen Repräsentation zu gelangen, bedarf es zunächst einer Einigung unter den avisierten Nutzern darüber, welche Sachverhalte des behandelten Themas wie dargestellt bzw. notiert werden sollen. In der Informatik bezeichnet man so eine Beschreibung eines Teils der Welt als *Ontologie* oder auch als *semantisches Netz*. Der Begriff ist der Philosophie entlehnt; seinen Einzug in die Informatik hat er mit einem Artikel von Gruber aus dem Jahr 1993 gehalten, in dem die folgende Definition verwendet wird [51]:

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

Diese Definition fasst prägnant zusammen, worum es sich bei einer Ontologie handelt. Sie ist eine formale Spezifikation, d.h. sie ist nach einem vorgegebenen Regelwerk erstellt und nachvollziehbar. Sie ist explizit formuliert, also weitestgehend eindeutig in ihrer Semantik und schließlich – und das ist der wichtigste Punkt – beschreibt sie eine von *einer Gruppe von Personen* geteilte Sicht auf die Welt. Gerade dieser Aspekt ist entscheidend, verbirgt sich doch dahinter der Abstimmungsprozess, in dem die subjektiven Sichtweisen der einzelnen Gruppenmitglieder aufeinander abgestimmt worden sind. Dadurch gewinnt die in der Ontologie formulierte Weltsicht ein deutlich größeres Gewicht als die einer einzelnen Person.

Damit bleibt nur zu klären, nach welchem Regelwerk Ontologien zu erstellen sind. Hierfür bieten sich Entwicklungen des World Wide Web Consortiums (W3C) an, das im Rahmen seiner Initiative zur Entwicklung des *Semantic Web* Sprachen entwickelt hat, mit denen Ontologien formuliert werden können. Ein Kernzitat zu den Zielen des Semantic Web findet sich in einem Artikel von Tim Berners-Lee im Scientific American vom Mai 2001 [16]:

> The Semantic Web will bring structure to the meaningful content of Web pages, creating an environment where software agents roaming from page to page can readily carry out sophisticated tasks for users. [...] The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation. [...] In the near future, these developments will usher in significant new functionality as machines become much better able to process and "understand" the data that they merely display at present.

Das Semantic Web ist also nicht als völlig neues Angebot neben dem World Wide Web zu sehen, sondern als Erweiterung davon zu verstehen. Diese Erweiterung soll es ermöglichen, die Bedeutung der auf einer Webseite enthaltenen Informationen in einer wohldefinierten Weise abzulegen. Damit wird die Grundlage dafür geschaffen, dass Maschinen nicht mehr nur die Informationen *anzeigen*, sondern auch damit sinnvoll *arbeiten* können. Dies ist gerade für die wissenschaftliche Nutzung interessant, da somit neue Möglichkeiten der Recherche in den Daten möglich werden, die sich mit Volltextsuchen auf dem Dokumentenbestand nicht realisieren ließen.

Obwohl es auf den ersten Blick scheinen mag, dass die Erweiterungen durch das Semantic Web im Wesentlichen der maschinellen Weiterverarbeitung von Webseiteninhalten dienen, sind die dafür entwickelten Sprachen RDF[3] und OWL[4] dennoch nicht an eine Verwendung im Kontext von Webseiten gebunden. Mit ihrer Hilfe lassen sich Ontologien für alle denkbaren Themen erzeugen, unabhängig von deren Repräsentation, womit ein mächti-

[3]Resource Description Framework, siehe Kapitel 3.1.2
[4]Web Ontology Language, siehe Kapitel 3.1.4

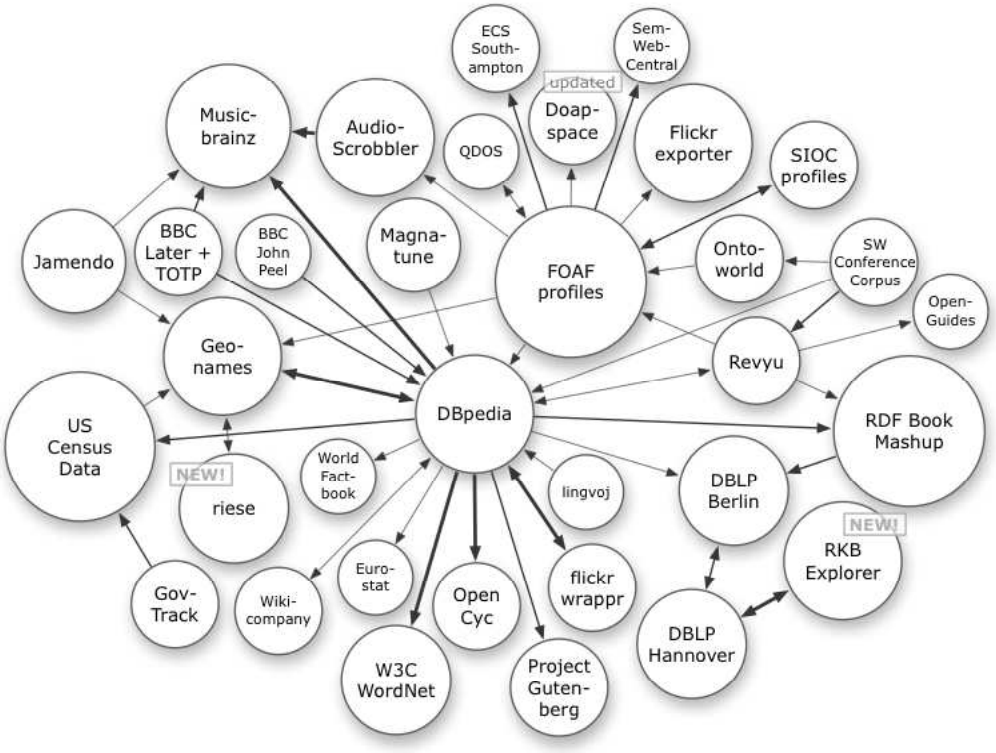


Abbildung 1.1: Aktueller Stand des Projekts Linking Open Data

ges Instrument für die inhaltliche Erschließung thematischer Sammlungen zur Verfügung steht.

Es ist nicht davon auszugehen, dass in absehbarer Zeit eine Ontologie entwickelt werden wird, mit der sich alle Themengebiete gleichermaßen abdecken lassen, dafür ist das Spektrum des Wissens zu breit und der benötigte Abstimmungsaufwand für eine solche, hypothetische "Weltontologie" zu groß. Die Verbreitung semantischer Technologien im WWW wird sich wahrscheinlich eher in der Schaffung von Themeninseln mit spezifischen Ontologien vollziehen. Ein Beispiel für eine aktive Förderung dieses Prozesses ist das Projekt "Linking Open Data", in dem bidirektionale Verbindungen (so genannte *Ontology Mappings*) zwischen verschiedenen offenen und im WWW verfügbaren Datenquellen von Hand erstellt und zur Verfügung gestellt werden[5]. Abbildung 1.1 zeigt den aktuellen Stand des Projekts[6].

Im Zentrum des Projekts steht die DBPedia[7] [7], eine Aufbereitung der Wikipedia für das Semantic Web. Daneben sind weitere große Sammlungen enthalten, unter anderem das

[5]Siehe `http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData`

[6]Die Abbildung wurde von Richard Cyganiak (richard@cyganiak.de) erstellt, verwendet unter Creative Commons License CC-BY-SA

[7]http://www.dbpedia.org

CIA World Factbook[8], das zentrale Fakten zu allen Ländern der Welt enthält, oder die Geonames Ontology[9], die geographische Daten zu vielen Orten der Welt enthält.

Erstellung von Ontologien

Für die Bereitstellung zusätzlicher semantisch erschlossener Datenbestände sind jedoch noch eine Reihe von Herausforderungen zu bewältigen, denn die Erstellung einer Ontologie ist kein einfaches Unterfangen. Die Sammlung und Einigung auf die in der Ontologie zu modellierenden Konzepte und deren Verknüpfungen untereinander ist sehr zeitaufwändig, damit kostenintensiv und benötigt sowohl Fachexperten (auch Domänenexperten genannt) als auch Ontologie-Ingenieure, also Personen, die auf die Erstellung von Ontologien spezialisiert sind. Die Inanspruchnahme ihrer Dienste treibt den Preis zusätzlich in die Höhe. Ein Bericht aus dem Jahr 2003 gibt die Kosten für die manuelle Erstellung eines Konzepts in einer Ontologie mit 40 britischen Pfund (also ungefähr 50 EUR) an [94]. Bereits kleine Ontologien weisen typischerweise mehrere Hundert Konzepte und Verbindungen zwischen diesen Konzepten auf. Die Kosten steigen zusätzlich, wenn Ontologien für Fachgebiete mit geringer Fehlertoleranz erstellt werden sollen, etwa im militärischen oder medizinischen Bereich.

Eine Konsequenz der hohen Kosten und der zu erwartenden langwierigen Diskussionen ist, dass es zu einer Fragmentierung der Anstrengungen zur Ontologiebildung kommen kann. Gerade wenn die erwarteten Kosten hoch sind, hält man die Anzahl der beteiligten Institutionen lieber klein, in der Annahme, dass so eine einfachere Einigung auf eine Ontologie erzielt werden könne. Selbst in hochgradig strukturierten Fachbereichen wie der Medizin gibt z.B. es gleich mehrere, nicht aufeinander abgestimmte, Ontologien, darunter sehr große wie MeSH [78], SNOMED CT [97] und UMLS [107]. Diese Systeme werden unter anderem zur Klassifikation von Krankheitsbildern und zur Beschreibung von Patientendaten in Krankenhäusern verwendet. Damit ergibt sich eine Fülle neuer Probleme, denn problemlos können Patientendaten nur zwischen Krankenhäusern ausgetauscht werden, in denen die gleiche Ontologie verwendet wird. Der Übertrag von Patientendaten von einer Ontologie in eine andere gestaltet sich so schwierig, das sich für den generalisierten Fall eine eigene Forschungsrichtung etabliert hat: Das Ontologie-Mapping.

Vor dem Kostenhintergrund, aber auch um die Verfügbarkeit von Ontologien für die verschiedensten Fachgebiete zu erhöhen, wird seit einigen Jahren an Systemen geforscht, die zumindest Teile der Ontologieerstellung automatisieren helfen. Kapitel 4.1 gibt einen Überblick über solche Systeme. Ihnen ist gemein, dass sie als Hilfsmittel für einen Ontologie-Ingenieur bei der Definition einer Ontologie gedacht sind. Daher wird nur ein Teil der Aufgaben automatisiert, insbesondere die, für die eine Interaktion mit den Fachexperten nicht erforderlich ist. Dazu gehören allerdings weder die Definition der Konzepte, noch die

[8]Siehe `http://www.cia.gov/library/publications/the-world-factbook/`
[9]Siehe http://www.geonames.org/ontology/

der Verknüpfungen zwischen ihnen, so dass so ein System von Fachexperten nicht ohne einen Ontologie-Ingenieur verwendet werden kann.

Zielsetzung

Wünschenswert wäre ein System, das die Erstellung einer Ontologie für einen Dokumentenbestand weitgehend automatisiert – und zwar wesentlich weiter, als dies heutige Systeme tun. Diese beschränken sich im Wesentlichen auf die Populierung von Konzepten mit Instanzen. Für das Konzept *Person* könnte dies z.B. darin bestehen, Namen aus einem internen Adressbuch als Instanzen von *Person* in die Ontologie zu übernehmen oder mittels automatischer Eigennamenerkennung neue, bisher unbekannte, Instanzen eines Konzepts in einer Textmenge zu entdecken. Hilfe bei der Suche nach Beziehungen zwischen den Konzepten wird hingegen heutzutage nicht angeboten.

An dieser Stelle will die vorliegende Dissertation ansetzen, indem Algorithmen entwickelt werden, die mögliche Relationen für eine vorgegebene Menge von Konzepten aus einem annotierten Korpus extrahieren. Damit wird den Fachexperten eine Übersicht über im Material vorkommende Verknüpfungen gegeben, mit der sie eine Entscheidung über den in der Ontologie zu verwendenden Relationenmenge treffen können. Die Umwandlung der Annotationen und der ausgewählten Relationen findet automatisch statt. Die in dieser Arbeit entwickelten Verfahren werden in ein Gesamtsystem integriert, das die Vorverarbeitung der Dokumente und die spätere Nutzung der Ontologie abdeckt. So wird ein System geschaffen, das es Fachleuten erlaubt, in Eigenregie Ontologien für ihr Fachgebiet auf der Basis vorliegender Korpora zu erstellen. Diese Ontologien stellen zudem eine maschinell lesbare inhaltliche Erschließung dieser Korpora dar.

Struktur der Arbeit

Die Dissertation gliedert sich wie folgt: Kapitel 2 definiert die Ziele dieser Arbeit, gibt die damit verbundenen Herausforderungen an und leitet anschließend Anforderungen an eine Lösung ab. Im Anschluss gibt Kapitel 3 Einführungen in verschiedene Standards bzw. Spezifikationen, die im Semantic Web - Umfeld verwendet werden, sowie kurze Erklärungen zu verwendeten Techniken und Technologien, ehe in Kapitel 4 verwandter Arbeiten aus der wissenschaftlichen Literatur besprochen werden, mit besonderem Augenmerk auf die vorher in Kapitel 2 definierten Anforderungen.

Kapitel 5 stellt den Ansatz dar, der zur Zielerreichung unter Berücksichtigung der Anforderungen gewählt worden ist, beschreibt die wissenschaftliche Vorgehensweise, die verwendeten Algorithmen und gibt Implementierungsdetails. In Kapitel 6 wird insbesondere das WIKINGER-Projekt beschrieben, in dem der in Kapitel 5 beschriebene Ansatz exemplarisch umgesetzt worden ist. Darüber hinaus enthält das Kapitel die Beschreibung eines Systems zur vollautomatischen Erstellung leichtgewichtiger Ontologien, das zur Er-

forschung der Möglichkeiten und Grenzen vollautomatisch erstellter Ontologien entwickelt worden ist.

In Kapitel 7 wird die Evaluierung des Ansatzes vorgenommen. Dargestellt werden die Ergebnisse der Evaluierung des Ansatzes anhand der im WIKINGER-Projekt erstellten Referenzimplementierung. Kapitel 8 schließlich bildet den Abschluss der Arbeit und enthält eine Bewertung der Ergebnisse des vorangegangenen Kapitels als auch Ausblicke auf weitere Entwicklungen und Erweiterungsmöglichkeiten des vorgeschlagenen Ansatzes.

Kapitel 2

Problemstellung

Dieses Kapitel zeigt die Ziele dieser Dissertation auf, sowie die Herausforderungen, die es zur Zielerreichung zu meistern gilt. Aus den Zielen und den Herausforderungen werden die Anforderungen abgeleitet, denen eine Lösung genügen muss.

2.1 Ziele der Dissertation

Im letzten Kapitel ist die grundlegende Ausgangssituation bereits geschildert worden. Es gibt bereits eine große Zahl digitaler Sammlungen und Archive im Netz, viele weitere werden in den nächsten Jahren folgen, unter anderem aufgrund der Bemühungen um die EDL. Für die inhaltliche Arbeit mit diesen Beständen werden Erschließungen unerlässlich sein, die weit über die Bereitstellung von Volltextsuchmaschinen hinaus gehen. Für die tiefere inhaltliche Erschließung der Bestände und ihrer Inhalte eignen sich sehr gut die Technologien, die durch das Semantic Web zur Verfügung gestellt werden. Allerdings ist nicht davon auszugehen, dass viele Einrichtungen die Mittel dazu haben werden, ihre Bestände auf herkömmliche Weise für das Semantic Web aufzubereiten. Um dieser Technologie zum Durchbruch zu verhelfen, ist es daher unerlässlich, den Prozess der Erstellung von Ontologien soweit zu automatisieren, dass Fachleute in die Lage versetzt werden, eine Fachontologie für ihre Sammlungen in Eigenregie zu erstellen.

Mit dieser Dissertation soll dazu beigetragen werden, Wege aufzuzeigen, über die digitale Datenbestände aus Fachbibliotheken oder Archiven für das Semantic Web erschlossen werden können – und zwar mit so wenig manueller Arbeit wie möglich. Dazu wird untersucht, inwieweit sich der Prozess der Erschließung eines solchen Bestandes für das Semantic Web automatisieren lässt, welche Rahmenbedingungen dazu nötig sind und inwieweit sich eine semantische Repräsentation des Bestandes anschließend tatsächlich zum Zugang zu den Inhalten des Bestands nutzen lässt.

Die Ziele lassen sich in drei Bereiche unterteilen: Die Erforschung neuer Algorithmen

zur Unterstützung des Prozesses, die Bereitstellung einer adäquaten Infrastruktur zur Einbettung dieser Algorithmen und schließlich Szenarien zur Verwendung der inhaltlichen Erschließung aufzuzeigen. Diese werden in den nachfolgenden Abschnitten näher ausgeführt.

2.1.1 Semi-automatische Erstellung semantischer Netze

Vorab ein kleiner Einschub zur Begriffsklärung. Der Begriff der Ontologie ist kurz Die vollautomatische Erstellung eines hochwertigen semantischen Netzes aus einer Datensammlung wird noch für absehbare Zeit auf sich warten lassen. Jedenfalls, solange man den kompletten Erstellungsweg von einer mehr oder weniger strukturierten Dokumentenmenge hin zum komplett autonom und automatisch erstellten semantischen Netz als Endprodukt betrachtet.

Die intellektuelle Arbeit der Bestimmung von Klassen mit ihren jeweils für den Anwendungsfall wichtigen und interessanten Attributen, sowie die anschließende Festlegung der möglichen semantischen Verknüpfungen zwischen Instanzen dieser Klassen, überfordert klar die Fähigkeiten heute denkbarer Softwaresysteme. Die Erzeugung von Instanzen solcherart definierter Klassen ist hingegen bereits heute möglich, geeignete Trainingsdaten als Beispiele für das maschinelle Erlernen der Charakteristika dieser Instanzen vorausgesetzt. Das ist jedoch der Teil der Ontologieerstellung, der im Wesentlichen einer Fleißarbeit gleich kommt.

Die vollautomatische Entdeckung komplexer Relationen hingegen ist deutlich außerhalb der maschinellen Möglichkeiten. Insofern ist auszuschließen, dass ein System komplett ohne menschliche Einflussnahme ein nicht triviales semantisches Netz eines komplexeren Datenbestands erzeugen können wird.

Wenn man jedoch von der Idee der Vollautomatik Abstand nimmt, so lassen sich Prozessschritte identifizieren, die sich weitestgehend ohne menschlichen Eingriff erledigen lassen. Dazu zählen unter anderem die Klassifikation von Texten bzgl. einer bestehenden Ontologie und die bereits angesprochene Überführung von Daten stark strukturierter Datenbestände in Instanzen vorgegebener Klassen einer Ontologie. Gerade diese Arbeiten sind manuell sehr zeitaufwändig und fehleranfällig. Gut trainierte Klassifikationssysteme können von der Qualität durchaus mit menschlicher Klassifikation mithalten – ihre Zuverlässigkeit ist zudem nicht tagesformabhängig. In diesen Bereichen ist eine Automatisierung also sehr wünschenswert.

In der Dissertation wird ein der Teilautomatisierung untersucht werden, der über solche, eher handwerklichen Erleichterungen hinausgeht. Hierbei soll ein Softwaresystem entwickelt werden, das Kandidaten für Relationen des semantischen Netzes erzeugt, die menschlichen Experten zur Durchsicht vorgelegt werden. Erst bei einem positiven Votum der Experten werden die Kandidaten der Relationsmenge des semantischen Netzes hinzugefügt.

Der Themenkomplex der semi-automatischen Erstellung von Inhalten für semantische

Netze bildet den Hauptteil der Arbeit, da sich hier das meiste Potenzial zur Verringerung der erforderlichen manuellen Arbeit findet. Je einfacher der Prozess der Erstellung semantischer Netze für bereits bestehende oder neu zu erstellende Sammlungen gestaltet werden kann, desto eher steht zu erwarten, dass sich die Anzahl solcher Angebote im Internet erhöht. Das wiederum eröffnet neue Perspektiven zur Verknüpfung verschiedener Datenbestände. Ultimativ ließe sich so ein weiterer Fortschritt auf dem Weg zum Semantic Web erzielen.

2.1.2 Infrastruktur

Der Übergang von einem analog vorliegenden zu einem digital verfügbaren Bestand erfordert eine bestimmte technische Infrastruktur. So ist üblicherweise die Einrichtung einer relationalen oder objektorientierten Datenbank, eines Webservers und eines Web Application Servers, sowie die Erstellung einer Webpräsenz für das digitale Angebot notwendig. Diese Bestandteile sind entweder vor Ort zu integrieren oder sind bereits in Teilen oder komplett in einer Anwendung gekapselt. Diese Anwendungen werden unter dem Begriff *Content-Management-Systeme*, abgekürzt CMS, geführt. Falls sie sich auf die Erstellung, Wartung und Pflege von Webpräsenzen konzentrieren, werden sie auch als *Web-Content-Management-Systeme* bezeichnet, wobei viele Hersteller auf die spezialisierte Bezeichnung verzichten, um breitere Einsatzfähigkeit ihrer Systeme zu suggerieren. (Web-)Content-Management-Systeme sind weit verbreitet, praktisch jeder größerer Web-Auftritt wird heutzutage mit einem erstellt und verwaltet. Die Verwendung eines CMS erhöht jedoch die Kosten für Änderungen am Gesamtsystem, da sie einen gewissen Arbeitsablauf mit festgelegten Komponenten voraussetzen.

Das wirft die Frage auf, inwieweit sich die Anforderungen an die Infrastruktur eines digitalen Archivs ändern, sobald es Komponenten des Semantic Web beinhaltet. Durch die erweiterten Vernetzungsmöglichkeiten der Dokumente untereinander, aber auch durch die Berücksichtigung der Ebene der Bedeutung der Dokument*inhalte*, werden viel höhere Anforderungen an die Dienste gestellt, die der Web Application Server zur Verfügung stellen muss. Hyperlinks haben auf einmal eine semantische Qualität, die Komposition der dynamischen Webseiten wird komplexer und falls das Angebot tatsächlich auch für Softwareagenten verwendbar sein soll, ist eine weitere Schnittstelle zu bedienen, die zwar keine Anforderungen an die graphische Ausgabe stellt, aber die Einhaltung von Spezifikationen erfordert, die für die Ausgabe semantischer Netzes geschaffen worden sind.

Für die Zusammenstellung webbasierter Systeme ist in den letzten Jahren eine neue Technologie entwickelt worden: Die so genannten Web Services. Ihnen liegt die Idee zu Grunde, dass Dienste auf einem Web Server verfügbar gemacht werden, auf die über klar definierte Schnittstellen (z.B. SOAP [43], REST [49] oder XML-Remote Procedure Call [112]) von außerhalb zugegriffen werden kann. Die Ausgabe der Ergebnisse dieser Anfragen erfolgt dabei nicht über Webseiten, sondern in Form von XML-Nachrichten, die maschinell

auswertbar sind.

Mit Hilfe von Web Services kann eine Vielzahl kleiner, spezialisierter Dienste zur Verfügung gestellt werden, aus denen je nach Anwendungsszenario größere Dienste zusammengestellt werden können (man spricht hier auch von Orchestrierung). Vor dem Hintergrund der Vision des Semantic Web, dass Informationen auch von Maschinen interpretiert werden können sollen, verdienen Web Services beim Architekturdesign eine genauere Betrachtung, zumal es Mechanismen zum semantischen Annotieren von Web Services gibt [47].

Die Suche nach einer passenden Architektur für die Einbettung der in dieser Arbeit erforschten Algorithmen und Verfahren ist eine weitere Aufgabe dieser Dissertation.

2.1.3 Nutzung der semantischen Netze

Der vorangegangene Abschnitt mag die Frage aufgeworfen haben, warum man sein System für die Verwendung des Semantic Web vorbereiten sollte, wenn sich dadurch die Anforderungen an die Infrastruktur deutlich erhöhen. Natürlich hängt die Art der Antwort auf diese Frage von einer Vielzahl von Faktoren ab, die nicht zuletzt finanzieller oder organisatorischer Natur sind. Es lässt sich jedoch generell sagen, dass sich durch den Einsatz von Komponenten und Techniken des Semantic Web die Möglichkeiten zur Präsentation der in einem Datenbestand enthaltenen Informationen vervielfältigen: In herkömmlichen digitalen Archiven sind Dokumente nach einigen wenigen Kategorien gruppiert, etwa nach Autoren, Dokumentenarten oder Zeitintervallen. Dies sind alles Daten, die in einer relationalen Datenbank typischerweise zu einzelnen Dokumenten erfasst sind und sich insofern mit heutiger Technik einfach abrufen lassen. Dadurch bleibt jedoch die Navigation in den Beständen sehr eng an die Daten der einzelnen Dokumente gebunden. Weiterführende, eher inhaltlich geprägte Einstiege, die als solche in der Bestandsdatenbank nicht erfasst sind, können jedoch nachträglich auf so einer Basis nicht ohne Weiteres realisiert werden und finden sich demzufolge eher selten.

Sobald jedoch eine maschinell auswertbare, semantische Beschreibung der Inhalte dieser Dokumente vorliegt, kann eine Vielzahl weiterer Zugänge zum Material angeboten werden. Dabei können die Konzeptklassen zur Gruppierung ähnlicher Inhalte des Netzes verwendet werden, Instanzen zur Ansicht von Detailinformationen ausgewertet werden. Zu der reinen Navigation über Dokumente gesellt sich also die Navigation über die restlichen modellierten Konzepte. Gegenüber einer Bestandsdatenbank hat das semantische Netz jedoch den Vorteil, dass es sich jederzeit um weitere Relationen und Konzeptklassen erweitern lässt. Es erlaubt also die spätere Erfassung weiterer Zusatzdaten, ja sogar die nachgelagerte Modellierung völlig neuer Aspekte der Sammlung. Und das alles, ohne die zu Grunde liegende Ontologie verändern zu müssen, denn die Erweiterungen können separat erfasst und abgelegt werden. Ihren Bezug zu bereits bestehenden Datenmodellen erhalten sie über Hyperlinks, auf die gleiche Art und Weise wie im bereits in Kapitel 1 erwähnten Projekt Linking Open Data.

Das zentrale Einsatzgebiet für semantische Netze ist also die Organisation und Beschreibung von Datenbeständen. In der bisher beschriebenen Art handelt es sich dabei um einen Prozess, der im Wesentlichen ohne Interaktion mit den späteren Nutzern abläuft. Idealerweise müssen die Nutzer nicht einmal wissen, welche Technik für die Organisation der Daten eingesetzt worden ist.

Daneben gibt es aber noch weitere Einsatzgebiete, die ein deutlich höheres Interaktionspotenzial mit den Nutzern eines Datenbestands aufweisen. Dazu gehört insbesondere das Angebot von Suchfunktionen, die sich das zusätzliche Wissen zu Nutze machen, das im semantischen Netz manifestiert ist. Mit den darin enthaltenen inhaltlichen Verbindungen lassen sich Anfragen beantworten, für die eine herkömmliche Volltextsuche keine befriedigenden Antworten liefern kann.

Speziell für diesen Zweck ist im Rahmen des Semantic Web Projekts eine Anfragesprache entwickelt worden, die Sprache SPARQL. Sie ist das Schlüsselelement für die Nutzung semantischer Netze sowohl in Sachen Strukturierung, als auch als Technik hinter einer direkten Suchschnittstelle für die Nutzer eines Angebots. Die Untersuchung der notwendigen Schritte und Mittel zur Einbindung einer SPARQL-Suche in das geplante System stellen das Hauptziel des Bereichs Nutzung dar. Eine Einführung in die Sprache SPARQL ist in Abschnitt 3.1.5 zu finden.

Für die Visualisierung von Datenbeständen in der Form semantischer Netze gibt es in der Literatur verschiedene Ansätze. Diese Dissertation hat ihren Fokus nicht auf Themen der Computergraphik, allerdings ist die Visualisierung der im Netz enthaltenen Informationen ein Thema, das nicht völlig ausgeklammert werden sollte. Interessant für diese Arbeit ist das Aufbereiten der Daten dergestalt, dass externe Visualisierungskomponenten bedient werden können. Das gilt besonders für solche Visualisierungen, die Spezifika der anzuzeigenden Daten ausnutzen, etwa Zeitstrahlen für temporale Daten.

2.2 Herausforderungen

Der vorangegangene Abschnitt hat die Ziele skizziert, die mit dieser Dissertation verfolgt werden. Auf dem Weg zu ihrer Erreichung wartet eine Reihe von Herausforderungen, die es zu überwinden gilt. In diesem Abschnitt werden die verschiedenen Herausforderungen herausgearbeitet und näher beschrieben. Der Abschnitt ist analog zu den Unterabschnitten von Abschnitt 2.1 unterteilt, um eine direkte Gegenüberstellung der Ziele mit den dazu gehörenden Herausforderungen zu ermöglichen.

2.2.1 Netzerstellung

In Abschnitt 2.1.1 sind bereits einige der Herausforderungen behandelt worden, die auf dem Gebiet der (semi-) automatischen Erstellung semantischer Netze existieren. Eine vollautomatische Erstellung ist sicher möglich, allerdings sind die dabei automatisch zu extrahierenden Relationen auf keinen Fall von der Qualität wie sie in manuell erzeugten Netzen erreichbar ist. Der Grund dafür liegt im Kontextwissen derjenigen, die so ein Netz erzeugen. Sie können Verbindungen zwischen Entitäten des Netzes herstellen, die keine explizite Entsprechung im vorliegenden Textkorpus haben, insofern auch nicht aus diesem extrahiert werden können. Darüber hinaus haben Menschen üblicherweise keine Probleme damit, eine Entität im Text zu identifizieren, auch wenn die Bezeichnung eine Variation oder ein Synonym des Namens ist, sie mit einem ihrer Attribute bezeichnet wird oder nur in Form eines Pronomens vorkommt. In solchen Fällen zu erkennen, dass eine anaphorische Verbindung zu der ersten Nennung besteht, ist ein aktiv beforschtes Problem aus der Computerlinguistik [1].

Es ist nicht zu erwarten, dass für nichttriviale Domänen eine vollautomatische Erstellung Netze in der erwarteten und benötigten Qualität liefert. Trotzdem gibt es Bedarf für eine algorithmische Unterstützung, um den manuellen Arbeitsaufwand möglichst klein zu halten. Eine dieser Unterstützungsmöglichkeiten besteht in der automatischen Erkennung von Entitäten, die im semantischen Netz berücksichtigt werden sollen.

Eine Entität ist dabei ein Vorkommen eines Begriffs im Datenmaterial, der innerhalb der Ontologie modelliert werden soll. Im Regelfall wird eine Entität als Instanz eines abstrakten Konzepts[2] modelliert werden, zum Beispiel wäre “Heiner Müller” eine Instanz des Konzepts *Person*. Je nach Domäne kann es eine recht große Anzahl von Entitäten geben, deren manuelle Entdeckung im Quellmaterial zur späteren Übertragung in die Strukturen der Ontologie eine zeitaufwändige und auch fehlerträchtige Aufgabe darstellt. Eine Möglichkeit, diese Entitäten automatisch zu entdecken, zu deduplizieren und zu disambiguieren, kann eine Menge Geld und Aufwand sparen, sowie gleichzeitig die Fehlerquote erheblich senken. Eine Vorbedingung für diese Art der Unterstützung ist zumindest die Verfügbarkeit von Beispielinstanzen der Konzepte.

Die Beschreibung eines Konzepts enthält eine Menge von Attributen, die dieses Konzept auszeichnen, sowie etwaige Beschränkungen, denen gültige Belegungen dieser Attribute unterliegen. Ein Beispiel für so ein Konzept zeigt Tabelle 2.1. Hier wird das Konzept Person definiert. Es zeichnet sich durch eine Reihe von Attributen aus, die seine Instanzen besitzen können bzw. müssen. Die erste Spalte enthält die Bezeichnung des Attributs, die zweite Spalte die Angabe, ob es sich bei dem Attribut um ein Pflichtfeld handelt oder nicht, die dritte gibt schließlich die Kardinalität des Felds an, d.h. ob es mehrfach für eine Instanz

[1] Siehe 3.6.2 in [64] für eine nähere Definition des Problems. Einen Überblick über wissenschaftliche Arbeiten dazu gibt [92].

[2] Im Umfeld des Semantic Web werden Konzepte auch als Klassen bezeichnet, weswegen dieser Begriff im weiteren Verlauf synonym verwendet werden wird.

Attribut	Erforderlich	Kardinalität
Nachname	ja	1
Vorname	ja	1
Titel	nein	0-n
Geburtsdatum	ja	1
Todesdatum	nein	0-1

Tabelle 2.1: Beispiel für das Konzept *Person*

vorhanden sein darf oder nicht. Optionale Attribute dürfen auch unbesetzt bleiben, für diesen Fall gilt dann Kardinalität 0. Am Beispiel sind einige Pflichtfelder zu erkennen, so sind Nachname, Vorname und Geburtsdatum zwingend erforderlich, um eine Instanz von Person erzeugen zu können. Dagegen ist die Angabe von Titel oder Todesdatum optional, d.h. ihre Kardinalität kann auch null sein. Im Fall des Todesdatums kann man sehen, dass es eine Kardinalitätseinschränkung gibt: Eine Person hat entweder kein Todesdatum (d.h. sie lebt noch) oder genau eins. Titelangaben hingegen kann es zu einer Person keine oder aber beliebig viele geben.

Schon dieses einfache Beispiel zeigt ansatzweise die komplexen Bedingungen, die sich in diesen Klassen modellieren lassen. Gleichzeitig sind Ähnlichkeiten zur Definition z.B. von Tabellen relationaler Datenbanken erkennbar. In der Tat eignen sich solche Datenbanken sehr gut als Quelle für Entitäten, denn die einzelnen Datensätze der enthaltenen Tabellen beschreiben üblicherweise je eine Entität, d.h. man gewinnt nicht nur einen Namen einer möglichen Instanz einer Klasse, sondern auch direkt verschiedene Attribute dieser Instanz. Dazu kommt, dass das Problem doppelter Nennungen und verschiedener Nennungsarten in Datenbanken üblicherweise nicht besteht. Zudem lassen sich aus den für einen Datensatz definierten Feldern die für die Entitätserstellung benötigten recht einfach auswählen – und der Auswahlvorgang muss nur einmal erfolgen, egal wie viele Datensätze die Tabelle enthält. Diese Eigenschaften machen Datenbanken zu den Quellen, aus denen sich am einfachsten Instanzen für ein semantisches Netz gewinnen lassen. Generell gilt das für alle tabellarische Quellen, auch wenn außerhalb von Datenbanken die Bedingungen für die Feldwerte üblicherweise weniger stringent gehandhabt werden.

Anders sieht das jedoch bei Textdokumenten aus. Hier ist von einem extrem niedrigen Grad an Strukturierung auszugehen. Unter Umständen lassen sich die Texte in Abschnitte zerlegen, jedoch variieren diese Abschnitte in ihrer Länge, außerdem enthält nicht jeder Abschnitt nur Daten zu einem bestimmten Thema oder einer bestimmten Entität. Vielmehr können beliebig viele Entitäten in einem Abschnitt vorkommen, wodurch sich die Komplexität der Extraktion stark erhöht. Dies gilt in gesteigertem Maß für die Attribute der Entitäten, da nicht sichergestellt ist, dass jedes Attribut einer Entität überhaupt im Text vorkommt. Das kann zu einer gesteigerten Rate von irrtümlichen Zurückweisungen vorhandener Entitäten führen, wenn als erforderlich markierte Attribute nicht belegt werden können (*false negative – Problem*). Sind hingegen die Attribute optional angelegt, kann es andererseits zu einer verstärkten Anzahl fehlerhafter Entitäten kommen, da Irrläufer in

der Klassifikation nicht durch Prüfungen der Erforderlichkeitsbedingung zurückgewiesen werden können (*false positive – Problem*).

Im Allgemeinen wird es innerhalb der Klassen eine kleine Anzahl von Attributen geben, die zur Identifikation späterer Instanzen unabdingbar sind und deswegen als erforderlich deklariert werden. Diese Attribute sind auch diejenigen, über die automatische Extraktionsverfahren Entitäten im Text erkennen sollen. Das reicht von relativ einfach zu erstellenden Extraktoren auf der Basis regulärer Ausdrücke bis zur Integration komplexer NLP[3]-Verfahren, die in der Lage sind, Satzbestandteilen ihre Funktion im Satz zuzuordnen, so Entitäten zu finden und diese über einen Text zu verfolgen. Eine Voraussetzung für den Einsatz von maschinellen Lernverfahren ist allerdings das Vorhandensein von Beispielen der Klassen, da die Verfahren erst für die Erkennung der gewünschten Klassen trainiert werden müssen. Abbildung 2.1 zeigt den typischen Ablauf des Trainingsverfahrens solcher automa-

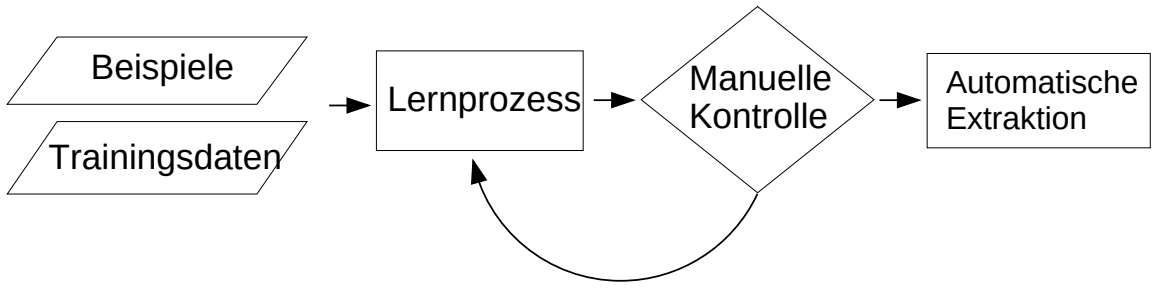


Abbildung 2.1: Trainingsprozess für automatische Entitäts-Extraktionsverfahren

tischen Extraktionsverfahren. Der Prozess benötigt für das Training manuell ausgezeichnete Beispiele. Diese werden zu Beginn in zwei Gruppen unterteilt: Die erste Gruppe wird mit den Auszeichnungen an das Lernverfahren übergeben, die korrekten Identifikationen sind also bekannt. Die zweite Gruppe wird ohne Auszeichnung präsentiert, die Klassifikation wird nicht mitgeliefert. Diese zweite Gruppe wird zum Training des Verfahrens verwendet. Anhand der bekannten Beispiele zeichnet das Verfahren die Trainingsdaten aus und die Resultate werden überprüft. Solange der Grad der Übereinstimmung zwischen dem Verfahren und der Klassifikation der Daten nicht gut genug ist, wird mit Korrekturen iteriert. Je nach Verfahren ist die Anzahl der benötigten Beispiele und Iterationen unterschiedlich. Die Überprüfung des Lernfortschritts lässt sich automatisieren, da die korrekte Klassifikation der Trainingsdaten bekannt ist, eine manuelle Auszeichnung der Trainingsdaten ist jedoch unumgänglich. Erst wenn die Qualität der Ergebnisse hinreichend für eine unbeobachtete Anwendung des Verfahrens ist, kann die eigentliche, automatische Erkennung von Entitäten statt finden.

Die automatische Erkennung der Entitäten erspart eine Menge manueller Arbeit auf dem Weg zu einem semantischen Netz des Bestands. Was noch fehlt, sind die Verbindun-

[3]Natural Language Processing

gen, die aus den isolierten Entitäten erst ein Netz machen. Eine vollautomatische Erledigung dieses Arbeitsvorgangs ist zwar möglich, allerdings beschränken sich die erzeugten Relationen dann auf diejenigen, die sich sicher aus dem Ausgangsmaterial ableiten lassen. Das sind in im Wesentlichen die Klassen-Subklassen-Beziehungen aus den Ontologien, die als initialer Input verwendet wurden. Besteht das Datenmaterial wenigstens zum Teil aus strukturierten Daten, so kann die daraus ableitbare Struktur ebenfalls automatisch auf das Netz übertragen werden. Ein so erzeugtes Netz zeichnet sich dadurch aus, dass es die hierarchischen Strukturen von Superklassen zu Subklassen und Instanzen gut abbildet, allerdings wenige bis gar keine Verbindungen auf Instanzenebene enthält. Man könnte also eher von einer Taxonomie als von einem Netz sprechen. Die Stärken eines semantischen Netzes, nämlich thematische Verbindungen zwischen typisierten Konzepten, werden so aber nicht ausgenutzt. Allerdings kann es Anwendungsfälle geben, in denen so eine Struktur schon ausreicht, etwa um eine Ontologie zur Klassifikation von Dokumenten einzusetzen. Zur Erschließung und Erkundung von Datenbeständen ist diese Art von Netzen jedoch nicht gut geeignet, da die inhaltlichen Verbindungen fehlen.

Um ein Netz zu schaffen, das außer den hierarchischen Strukturen auch inhaltliche Verbindungen enthält, sollte also ein Verfahren verwendet werden, das die Etablierung inhaltlicher Relationen auf Instanzebene ermöglicht. Im Gegensatz zu Verknüpfungen auf Klassenebene können inhaltliche Verknüpfungen zwischen Instanzen nicht global vorgegeben werden. Man kann nicht automatisch davon ausgehen, dass jede Instanz die gleichen Relationen aufweisen wird. Manche werden deutlich mehr Beziehungen zu anderen Instanzen aufweisen als andere. Mehrdeutigkeiten im Textmaterial erschweren die Arbeit für maschinelle Lernverfahren zusätzlich. Hier spielt die Kontrolle der Ergebnisse eine große Rolle. Daher ist ein iterativer Prozess erforderlich, in dem menschliche Experten die Möglichkeit zur Korrektur der aktuellen Struktur des Netzes erhalten und dem System so Hinweise geben können, um zu besseren Hypothesen zu gelangen.

2.2.2 Infrastruktur

Der Aufbau digitaler Archive folgt üblicherweise dem Schema, das in Abbildung 2.2 gezeigt ist. Zentral für dieses System ist ein Web Server, auf dem sowohl die statischen Seiten des Web-Auftritts, als auch die dynamischen Anteile enthalten sind, die unter anderem für die Anzeige des digitalen Archivs benötigt werden. Diese dynamischen Seiten werden üblicherweise innerhalb eines sogenannten Application Servers verwaltet, der als Modul des Web Servers läuft. Der Application Server enthält die Progammlogik, mit der die dynamischen Seiten zusammengesetzt werden. Dazu greift er auf Ressourcen außerhalb des Web Servers zu, zum Beispiel auf Datenbanken oder angeschlossene Dateisysteme. Eine Wartungsschnittstelle ist an den Web Server angeschlossen, über sie werden die Webseiten gepflegt, sowie Web Server und Application Server gewartet. Typischerweise kommt an dieser Stelle ein Web Content Management System zum Einsatz, in vielen Fällen bildet es auch eine Einheit mit dem Web Server, d.h. auf der Server-Hardware ist das Content Management

System installiert, das einen Web Server enthält und nach außen zur Verfügung stellt. Alle links vom Web Browser dargestellten Komponenten gehören logisch zur Serverseite, auch wenn sie vielleicht über mehrere Rechner und Standorte verteilt sein sind. Zum Zugriff auf die Angebote des Web Servers reicht auf Nutzerseite ein einfacher Webbrowser.

Ein digitales Archiv, das so aufgesetzt ist, bildet ein abgeschlossenes System. Das macht es sehr schwierig, in die Architektur neue Komponenten einzupassen. Das gilt auch für die Integration semantischer Informationen in das System, die zur Navigationsverbesserung oder zur Unterstützung von Software-Agenten verwendet werden sollen. Das macht die Suche nach einer geeigneten Architektur für ein digitales Archiv im Sinne des Semantic Web zu einer großen Herausforderung. In der Zielbeschreibung zur Infrastruktur (s. Abschnitt 2.1.2) ist bereits das Stichwort Web Services gefallen. Diese stellen sicher eine bedenkenswerte Alternative zum heute gebräuchlichen Client-Server-Ansatz dar, eine Architektur auf der Basis von Web Services, auch oft SOA für Service Oriented Architectures abgekürzt, birgt jedoch ihre eigenen Herausforderungen:

Wertübergabe In herkömmlichen Web-Anwendungen können verschiedene Teile der Anwendung Objekte über einen Zwischenspeicher austauschen und direkt verwenden, intern kann also ausschließlich mit Objekten der jeweils gewählten Programmiersprache umgegangen werden. Das geht mit Web Services nicht. Da davon ausgegangen werden muss, dass Web Services auf verschiedenen Servern im Internet verteilt sind, ist ein Datenaustausch über Objektreferenzen nicht vorgesehen, da diese nur pro Server eindeutig vergeben werden könnten. Statt dessen werden zum Austausch von Daten zwischen Web Services XML-Nachrichten verwendet. Dazu ist es notwendig, die zu übertragenden Daten in eine String-Repräsentation zu überführen. Das funktioniert problemlos für primitive Datentypen; für die Serialisierung komplexer Datentypen ist mehr Arbeit zu leisten, da die Modellierung über Objektreferenzen nicht möglich ist. Hier sollte im Vorfeld schon die Verwendung spezieller Identifikatoren eingeplant werden, die auch über textuelle Repräsentationen eine eindeutige Identifikation verschiedener Objekte zulassen. Auf der Gegenseite müssen diese Transformationsschritte anschließend in umgekehrter Reihenfolge nachvollzogen werden. Diese Serialisierung bzw. Deserialisierung verlangsamt den Datenaustausch zwischen den Services und stellt eine nicht zu unterschätzende Fehlerquelle, etwa bei der Übergabe nicht erwartungskonformer XML-Daten, dar.

Versionierung Der Zugriff auf Web Services regelt sich über Konfigurationsdateien in der XML-Sprache WSDL (Web Service Description Language [31]). Diese enthalten Informationen darüber, welche Methoden der Dienst zur Verfügung stellt, welche Parameter gesetzt werden können, welche Eingaben erforderlich sind und schließlich, welche Daten wie als Antwort übermittelt werden. Damit erlauben diese Konfigurationsdateien den entfernten Zugriff auf die Dienste ohne weitere Interaktion mit den Anbietern. Das ist einerseits ein Vorteil, andererseits aber auch ein Nachteil, der nicht zu unterschätzen ist: Sind Änderungen an einem bestimmten Dienst geplant, so können diese nur in einem begrenzten Rahmen durchgeführt werden, ohne

Änderungen an den WSDL-Dateien zu bedingen. Abgesehen von dem Fall, dass nur Funktionen hinzugefügt werden, sollte dann der Dienst versioniert geändert werden, d.h. die Änderungen werden in einer neuen Version durchgeführt, die alte unverändert bestehen gelassen. Dadurch wird vermieden, dass bisherige Nutzer des Dienstes auf einmal ausgesperrt sind und ihre Anwendungen nicht mehr funktionieren. Gleichzeitig handelt man sich damit aber u. U. die Notwendigkeit zum parallelen Pflegen verschiedener Versionen der Dienste ein, mit allem Aufwand, der damit verbunden ist – zumindest solange, bis man alle Nutzer zum Wechsel auf die neue Version bewegen kann.

Orchestrierung Unter Orchestrierung versteht man die Definition des Zusammenspiels verschiedener Web Services zum Erfüllen bestimmter Aufgaben. Dabei können sowohl lokale Services zum Einsatz kommen, als auch solche, die andernorts implementiert und angeboten werden. In vielen Fällen macht gerade dieser Aspekt des Zusammenstellens der am besten passenden Dienste, egal woher sie kommen und wo sie zugreifbar sind, den Reiz von SOA aus. Gleichzeitig steigert es jedoch die Komplexität des eigenen Systems deutlich, da sich die Form der Eingangs- und Ausgangsdaten dieser entfernten Dienste nicht beeinflussen lässt, mithin also mehr Arbeit in die Systemintegration investiert werden muss, unter Umständen sogar mehr als einmal, falls sich die Dienstdefinitionen ändern sollten (s.o.).

Diese Aufstellung zeigt exemplarisch, dass SOA nicht als Antwort für alle Architekturfragen herhalten können oder sollten. Nichtsdestotrotz ist eine sorgfältige Abwägung nützlich, gerade wenn Erweiterbarkeit, Vernetzung mit anderen und die Unterstützung maschinell auswertbarer Schnittstellen zu den Anforderungen an ein Zielsystem gehören.

Ein anderer wichtiger Punkt bei der Abhandlung der Herausforderungen auf Infrastrukturseite ist die Frage der Laufzeit der verwendeten Algorithmen. Ein System, dessen Antworten lange auf sich warten lassen, ist aus Nutzersicht unattraktiv, es sei denn, es ist von vornherein abzusehen, dass es lange dauert *und* es keine andere Möglichkeit gibt, an die Informationen zu kommen. Im Fall von Internetsystemen kommt hinzu, dass sich die Nutzer in den letzten Jahren an kurze Übertragungs- und Antwortzeiten gewöhnt haben, insofern noch viel weniger gewillt sind, auf eine Antwort länger zu warten, als es sie kosten würde, ihre Anfrage bei der Suchmaschine ihrer Wahl einzutippen – selbst wenn die Antworten dadurch schlechter werden! Es reicht also nicht aus, bessere Ergebnisse zu liefern als Suchmaschine X, sie müssen auch mindestens so schnell geliefert werden.

2.2.3 Nutzung der semantischen Netze

Die vorangegangenen Abschnitte haben die Herausforderungen aufgezeigt, die bei der Erzeugung semantischer Netze aus vorliegenden Datenbeständen und dem Design der dafür zu verwendenden Architektur zu meistern sind. Um einen Nutzen aus diesen Netzen zu

ziehen, sind zusätzliche Herausforderungen zu bewältigen, die in diesem Abschnitt aufgezeigt werden sollen. Die Zielbeschreibung in Abschnitt 2.1.3 hat die Unterstützung der Navigation, die zu den Daten passende Visualisierung von Zusammenhängen innerhalb des Netzes und die Ermöglichung der semantischen Suche als Ziele genannt. Die sich dabei ergebenden Herausforderungen werden nachfolgend herausgearbeitet.

Das Gebiet der Informationsvisualisierung ist ein intensiv beforschter Bereich, denn als Nahtstelle zwischen Information Retrieval, Information Extraction und Human Computer Interfaces kommt ihm eine große Bedeutung zu: Ohne eine effiziente Visualisierung können Nutzer das zusätzliche Wissen nicht ausnutzen, das ihnen von den automatischen Verfahren zur Verfügung gestellt wird. Semantische Netze lassen sich im Allgemeinen durch gerichtete Graphen darstellen, so dass hier auf einem reichhaltigen Bestand an Arbeiten aufgebaut werden kann. Aufgrund der Struktur semantischer Netze ist nicht auszuschließen, dass der Graph Zyklen enthält, was bei der Auswahl der Algorithmen und Visualisierungsmethoden zu berücksichtigen ist.

Die Erstellung inhaltlicher Einstiege in einen Datenbestand profitiert enorm von dem Vorhandensein einer Ontologie, die das Themengebiet zumindest grob beschreibt. Die durch sie vorgegebene Strukturierung der Domäne kann direkt für die Strukturierung verschiedener Einstiege in das Datenmaterial genutzt werden. Die Gestaltung dieser Einstiege kann auf verschiedene Arten erfolgen. Klassisch ist eine hierarchische Führung, vom Allgemeinen zum Spezifischen, wie sie auch eine Taxonomie bereitstellen würde. Diese Art der Einstiege eignet sich gut für eine textuelle Darstellung.

Neben der textuellen Ansicht gibt es verschiedene Ansätze zur graphischen Visualisierung der Zusammenhänge, klassisch als zweidimensionale Ansicht eines Graphen mit den Inhalten als Knoten und den Verknüpfungen als Kanten oder als hyperbolischer Baum, einer Ansichtsart, die einen dreidimensionalen Eindruck des Datenmaterials erweckt. Darüber hinaus hat es auch Ansätze gegeben, Daten in virtuellen Umgebungen als Räume zu visualisieren, etwa in einer Art virtueller Stadt oder als 3D-Landkarte. Das generelle Problem solcher Ansichten ist allerdings, dass sie leicht die Nutzer überfordern, da diese sich nicht eingehender mit den verwendeten Metaphern beschäftigt haben. Zudem sind diese Darstellungen eher zum Browsen denn zum spezifischen Suchen nach Informationen geeignet, so dass kommerzielle Systeme immer standardmäßig die textuelle Ansicht verwenden und höchstens als graphische Spielerei auch eine zweidimensionale Ansicht anbieten.

Nichtsdestotrotz ist jedoch die Verwendung spezialisierter Visualisierungen für spezielle Datentypen interessant. So bietet sich für die Darstellung zeitlicher Zusammenhänge die Verwendung von Zeitstrahlen an, da diese Ansicht dabei hilft, die Abfolge verschiedener Ereignisse zu verdeutlichen. Ebenso kann sich die Verwendung echter Landkarten bei der Darstellung geographischer Informationen anbieten, etwa um mit einem Blick lokale Häufungen in der Datenbasis zu verdeutlichen. Solche Visualisierungen lassen sich gut aus den Daten befüllen, die zu Instanzen in den Ontologien abgelegt worden sind – und erlauben eine Verlinkung zu den Gründen, die zu der Anzeige geführt haben.

Damit ist die wichtigste Herausforderung angesprochen worden, die für die Nutzung semantischer Netze sichergestellt werden muss: Die Ermöglichung der semantischen Suche. Das Ziel muss sein, die Beantwortung von Anfragen zu ermöglichen, an denen herkömmliche Volltextsuchmaschinen scheitern. Mit der Qualität der Ergebnisse steigt allerdings auch die Komplexität der Suche auf der Ontologie. Hierzu sind Sprachen in Entwicklung, allerdings sind diese nicht für die Anwendung durch normale Nutzer geeignet. Benötigt werden daher Suchschnittstellen, die komplexe Suchanfragen ermöglichen, ohne die Nutzer zu überfordern. Die Erstellung solcher Schnittstellen wird sicherlich nicht ohne Einschränkungen in der Art der möglichen Anfragen zu machen sein; die Herausforderungen hierbei sind die Wahl der Darstellung und die Wahl der übrig bleibenden Möglichkeiten.

Die letzte Herausforderung in diesem Bereich betrifft die Art der Nutzung. Die bisherigen Szenarien gingen implizit von einem feststehenden Datenbestand aus, der zu Beginn des gesamten Erstellungsprozesses bekannt ist. Für Anwendungen, die lediglich die Recherche in einem statischen Bestand erleichtern sollen, ist das auch tragfähig. Hat man es allerdings mit einem dynamischen Datenbestand zu tun, so verändert sich während der Lebensdauer des Systems die Datenbasis. Um den Datenbestand abzubilden, muss sich somit auch das semantische Netz ändern.

Szenarien hierfür sind Dokumentenserver, Wiki-Systeme (siehe hierzu Abschnitt 3.3.4), ja sogar die Arbeit eines Nutzers auf einem lokalen Rechner mit ihren Auswirkungen auf seine Dateien. Um zu verhindern, dass das Netz und der Datenbestand sich auseinander entwickeln und so das Netz unbrauchbar wird, sind Maßnahmen zu treffen, die die Konkordanz zwischen dem semantischen Netz und dem dynamischen Datenbestand sicherstellen.

2.3 Anforderungen an eine Lösung

Im vorangegangenen Abschnitt sind die Herausforderungen herausgearbeitet worden, denen sich ein Ansatz stellen muss, um die Ziele zu erfüllen, die in Abschnitt 2.1 aufgeführt worden sind. In diesem Abschnitt werden aus ihnen Anforderungen abgeleitet, denen ein Ansatz zur Erfüllung der Ziele genügen muss. Die Anforderungen sind in Unterabschnitte analog zu den vorhergehenden Abschnitten dieses Kapitels gruppiert und werden aufsteigend durchnummeriert präsentiert, da nachfolgende Abschnitte und Kapitel noch auf sie Bezug nehmen werden.

2.3.1 Anforderungen im Bereich Netzerstellung

Die entscheidenden Herausforderungen sind diejenigen aus Abschnitt 2.2.1, da sie direkt die Hauptziele betreffen. Daher entstammen die meisten Anforderungen an eine Lösung auch aus diesem Abschnitt.

Die erste Anforderung beschäftigt sich mit der Gewinnung der Konzepte, die für die automatische Auswertung von Inputdaten essentiell sind. Ohne eine grundlegende Spezifikation der gesuchten Konzepte und ihrer Attribute können keine maschinellen Lernverfahren eingesetzt werden. Um diese auch im späteren Netz einsetzen zu können. sollte die Definition direkt in Sprachen des Semantic Web erfolgen. Es ist anzumerken, dass über diese Anforderung auch der Import stützender Ontologien, Thesauri oder Taxonomien abgedeckt ist, selbst wenn diese nicht als Input für automatische Lernverfahren dienen sollen.

Anforderung 1 (Import und Verarbeitung von Konzepten)
Eine Lösung muss in der Lage sein, Beschreibungen von Konzepten in einer Beschreibungssprache des Semantic Web entgegenzunehmen und auf das Quellmaterial anzuwenden.

Digitale Datenbestände können in einer Vielzahl von Formen vorliegen, sei es tabellarisch, als Text, als Bild, Video oder Musik- bzw. Sprachmitschnitt. Zu jeder dieser Formen gibt es eine Vielzahl von Formaten, in denen die Daten codiert sein können. Daher ist es illusorisch, zu fordern, dass ein System Daten in jedem Format importieren, verstehen und verarbeiten können sollte. Andererseits ist ein System, das den Import nur eines bestimmten Formats verlangt, stark eingeschränkt in seinen Anwendungsmöglichkeiten. Also sollte ein Lösungsansatz in der Lage sein, zumindest einen (möglichst repräsentativen) Vertreter jeder Datenform zu unterstützen, deren Integration im jeweiligen Anwendungsszenario sinnvoll ist. Wird dabei zusätzlich auf die Unterstützung eines verbreiteten, vielleicht sogar offenen Formats geachtet, können Daten in anderen Formaten einfacher in das jeweilige Zielformat übertragen werden. Daher wird die folgende Anforderung aufgenommen.

Anforderung 2 (Import unterschiedlicher Datenformate)
Eine Lösung muss in der Lage sein, verschiedene Datenformate zu verarbeiten. Dazu gehören sowohl unstrukturierte Daten, z.B. Volltexte, als auch strukturiertere in Tabellenform, bzw. in der Form relationaler Datenbanken.

In den vorangegangenen Abschnitten sind die maschinellen Lernverfahren bereits vielfach erwähnt worden, daher braucht die nachfolgende Anforderung auch nicht motiviert zu werden. Die Entitätengewinnung ist entscheidend für die weiteren Arbeitsschritte, ihre Qualität bestimmt die Qualität des resultierenden Netzes.

Anforderung 3 (Automatische Extraktion von Konzeptinstanzen)
Eine Lösung muss Funktionalitäten zur Verfügung stellen, die eine automatische Extraktion von Instanzen der vorher definierten Konzepte ermöglichen.

Beinahe ein Korollar der vorhergehenden Anforderung stellt die nun folgende dar: Damit die Lernverfahren ihre Arbeit verrichten können, benötigen sie Vorgaben für die zu lernenden Klassen. In der Trainingsphase des Systems muss also eine Möglichkeit zur Verfügung stehen, diese Beispiele manuell zu generieren und dem System mitzuteilen. Zum Festhalten dieses Sachverhalts ist die folgende Anforderung gedacht.

Anforderung 4 (Annotierung von Beispielen)
Eine Lösung muss Funktionalitäten zur Verfügung stellen, die eine manuelle Annotierung von Beispielen für Instanzen der verschiedenen Konzepte erlauben.

Wenn ein System die bisherigen Anforderungen erfüllt, so sind zwar die möglichen Instanzen gefunden und klassifiziert worden, damit besteht aber noch kein semantisches Netz im Sinne des Semantic Web. Das bisher bestehende Netz hilft primär bei der Verschlagwortung, bzw. der Klassifikation, der enthaltenen Dokumente. Um ein reichhaltiges Netz zu erhalten, ist ein weiterer Schritt nötig, der Verknüpfungen auf Instanzebene anlegen kann. Dies wird in der nachfolgenden Anforderung festgehalten.

Anforderung 5 (Ermöglichen von Verknüpfungen auf Instanzebene)
Eine Lösung muss semantische Netze erzeugen können, die inhaltliche Verknüpfungen auf Instanzebene aufweisen.

Die automatische inhaltliche Vernetzung auf Instanzebene, die in dieser Anforderung postuliert wird, geht über die Fähigkeiten heutiger Computer hinaus, wenn mehr als nur eine rudimentäre Vernetzung erreicht werden soll. Für diesen Schritt sind deshalb semiautomatische Verfahren erforderlich, die Korrekturen und Erweiterungen durch menschliche Experten zulassen. Das impliziert einen Prozess, in dem Ergebnisse der Verarbeitung begutachtet, bei Bedarf korrigiert und in die weitere Verarbeitung integriert werden können.

Anforderung 6 (Semiautomatisches Verfahren)
Die Erstellung des Netzes muss in einem Prozess ablaufen, der Korrekturen und Erweiterungen durch menschliche Experten ermöglicht und aufgreift.

Diese sechs Punkte legen die Anforderungen an ein System fest, das die semiautomatische Erstellung semantischer Netze aus digitalen Datenbeständen verschiedener Formate erlaubt. Diese Anforderungen betreffen das Vorgehen und die Funktionalitäten eines geeigneten Prozesses hierzu. Spezifische fachliche Rahmenbedingungen der verschiedenen Anwendungsszenarien, in denen solch ein System eingesetzt werden soll, können zusätzliche Anforderungen an das System bedingen.

2.3.2 Anforderungen im Bereich Infrastruktur

Die Umwandlung der Herausforderungen aus Abschnitt 2.2.2 in Anforderungen an das zu schaffende System gestaltet sich deutlich schwieriger, als das für die Anforderungen aus Abschnitt 2.2.1 der Fall war. Der Grund hierfür liegt in der deutlich stärkeren Abhängigkeit der passenden Architektur vom jeweiligen Anwendungsszenario. Dennoch lassen sich zwei Anforderungen extrahieren, die generell relevant sind.

Die erste Anforderung in diesem Abschnitt unterstreicht eine der Kernannahmen des Semantic Web: Dort ist die Vernetzung mit anderen Einrichtungen eine zentrale Idee,

die Erweiterung und Einbettung bestehender Arbeiten (sprich Ontologien) erklärtes Ziel. Übertragen auf Softwaresysteme, die sich Techniken des Semantic Web zunutze machen, heisst das, dass ein solches System in der Lage sein muss, Erweiterungen zu integrieren, die erst zur Laufzeit des fertigen Systems entstehen, bzw. entstanden sind, aber zur Verarbeitung neuer Datentypen oder bzgl. neuer Ontologie-Erweiterungen nötig sind.

Anforderung 7 (Erweiterbare Architektur)
Die Systemarchitektur muss die Erweiterung um neue Schnittstellen und Module erlauben.

Die zweite Anforderung dieses Kapitels greift die Diskussion um die Laufzeit, also die Performanz eines geplanten Systems auf. Hierbei spielt besonders die gefühlte Performanz von interaktiven Prozessen eine Rolle. Das sind zumeist Anfragen an ein solches System sowie die Trainingsphasen bei der Vorbereitung des Systems. Wenn die semantische Suche deutlich mehr Zeit beansprucht als die Volltextsuche, so wird es schwer, Nutzer für ihre Verwendung zu begeistern - ungeachtet der Vorteile, die sie mit sich bringen mag. Insofern müssen besondere Vorkehrungen getroffen werden, um eine zumindest ähnliche Laufzeit sicherzustellen.

Für die Trainingsphasen des Systems gilt das nicht im gleichen Maße, allerdings sollte auch hier darauf geachtet werden, dass die Expertendie Ergebnisse eines Trainingslaufes in vertretbarer Zeit erhalten.

Anforderung 8 (Performanz)
Suchanfragen an ein semantisches Netz sollten keine spürbar längere Laufzeit haben als vergleichbare Anfragen an eine Volltextsuchmaschine.

Natürlich spielt auch bei nicht-interaktiven Prozessen die Performanz eine wichtige Rolle, allerdings reduziert sich die Bewertung der Performanz hierbei nicht nur auf die reine Antwortzeit, vielmehr sind hier Fragen des Durchsatzes, der Skalierbarkeit und der Parallelisierbarkeit vordergründig.

2.3.3 Anforderungen im Bereich Nutzung

Die in Abschnitt 2.2.3 beschriebenen Herausforderungen decken eine weite Spanne der gewünschten Nutzungsszenarien ab. Besonders auf die Art der vorliegenden Daten angepasste Visualisierungen versprechen Erkenntnisgewinne, die sich nur mit den durch die inhaltliche Erschließung zusätzlich verfügbar gemachten, maschinell interpretierbaren, Daten erzielen lassen. Jedoch sind diese Visualisierungen sehr stark von dem in der jeweiligen Anwendung verfügbaren Material abhängig. Daher muss die Umsetzung spezifischer Visualisierungen den verschiedenen Anwendungen vorbehalten bleiben, die auf dem geplanten System aufsetzen.

In Abschnitt 2.2.3 wurde ebenfalls die semantische Suche angesprochen. Sie ist eine der zentralen Verheissungen des Semantic Web: Suche auf der Basis wohldefinierter Bedeutun-

gen und nicht mehr auf der Basis des Vorkommens verschiedener Stringketten in Texten. Mit der Verbesserung der Suchmöglichkeiten geht jedoch die Verwendung einer wesentlich komplexeren Anfragesprache einher. Wo heutige Suchmaschinen im Wesentlichen Ausdruckslogik verwenden (deren Verkettungsmöglichkeiten jenseits der UND-Verknüpfung von den meisten Nutzern selten benutzt werden), wird für die semantische Suche Prädikatenlogik benötigt. So lehnt sich SPARQL sowohl an SQL als auch an Prolog an, was gleichzeitig bedeutet, dass nur die wenigsten Nutzer in der Lage sein dürften, ihre Anfrage direkt in SPARQL-Syntax zu formulieren.

Aus diesem Grund lässt sich auch die Erstellung von Oberflächen zum Zugriff auf die Suche nicht so verallgemeinern wie das im Fall von Volltextindexen möglich ist. Die Art der Eingabemöglichkeiten muss sich sowohl an den erwarteten Kenntnisständen der Nutzer orientieren, als auch die Entitätsklassen der Ontologie berücksichtigen.

Den beiden vorgestellten Nutzungsszenarien Visualisierung und semantische Suche ist gemein, dass sie einen Zugang zu den Daten der Ontologie benötigen, um daraus die jeweils benötigten Daten zu extrahieren und anzeigen zu können. Als Anforderung lässt sich daraus also ableiten, dass das System Möglichkeiten bereitstellen muss, die es ermöglichen, anwendungsspezifische Ansichten der in der Ontologie enthaltenen Daten zu erstellen.

Anforderung 9 (Ermöglichen unterschiedlicher Sichten auf die Ontologie)
Die Schnittstelle zum semantischen Netz muss die Definition verschiedener Sichten auf das Netz ermöglichen.

Der letzte Aspekt, der in Abschnitt 2.2.3 angesprochen wurde, betraf die Unterstützung dynamischer Datenbestände. Daraus lässt sich eine Anforderung generieren, von deren Erfüllung praktisch alle denkbaren Anwendungsfälle profitieren können.

Anforderung 10 (Unterstützung dynamischer Datenbestände)
Das System muss dynamische Datenbestände als Grundlage des semantischen Netzes unterstützen.

Selbst in Systemen, bei denen mit einem statischen Bestand gestartet wird, ist unter Berücksichtigung von Anforderung 7 die Möglichkeit zumindest vorzusehen, dass irgendwann zur Laufzeit weitere Bestände in das System zu integrieren sind. Damit wäre auch hier ein Zustand dynamischer Datenbasen erreicht, selbst wenn die neuen Bestände selbst alle statisch sein sollten.

Diese Anforderung schließt den Sammlungsprozess ab. Die zehn in diesem Unterkapitel aufgeführten Anforderungen definieren den Rahmen, innerhalb dessen sich eine Lösung im Sinne dieser Dissertation bewegen und bewähren muss.

2.4 Zusammenfassung

Diese Kapitel hat die Domäne abgesteckt, in der sich die Dissertation bewegt. Das Hauptaugenmerk liegt auf der Entwicklung von Verfahren, die bei der Aufarbeitung digitaler Mediensammlungen für das Semantic Web Unterstützung leisten. Dabei wird der gesamte Prozess der Aufarbeitung unterstützt, von der Infrastruktur, die für die Speicherung und Nutzung der Daten und Metadaten nötig ist, über die Erstellung eines semantischen Netzes, das die Inhalte abbildet, bis hin zur Bereitstellung von Schnittstellen, über die die Inhalte in geeigneter Form dargestellt werden können. Dieser Prozess soll in einem integrierten System abgebildet werden.

Die zu meisternden Herausforderungen auf dem Weg zur Zielerreichung sind aufgeführt worden, ebenso wie die sich daraus ergebenden Anforderungen, denen das geplante System genügen muss.

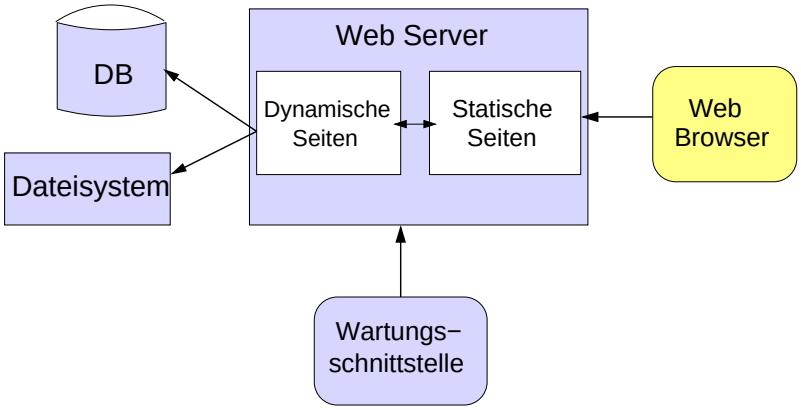


Abbildung 2.2: Klassischer Aufbau digitaler Archive

Kapitel 3

Grundlagen

In diesem Kapitel werden Begriffe und Techniken erläutert, die im Umfeld der Dissertation angesiedelt sind. Abschnitt 3.1 erläutert Standards und Spezifikationen aus dem Umfeld des Semantic Web. Danach folgt in Abschnitt 3.2 die Erläuterung wichtiger Begriffe und Algorithmen. Den Abschluss des Kapitels bildet Abschnitt 3.3, in dem grundlegende Techniken erläutert werden, die innerhalb der Dissertation Anwendung finden.

3.1 Standards und Spezifikationen

Vor der Erläuterung der einzelnen Standards und Spezifikationen soll an dieser Stelle der Unterschied zwischen Standards und Spezifikationen erläutert werden. Viele Organisationen beteiligen sich an der Weiterentwicklung der technischen Grundlagen zum Beispiel des Internets. Die Ergebnisse der Arbeiten dieser Einrichtungen werden oft verallgemeinert als "Standards" bezeichnet. Dabei handelt es sich allerdings um einen terminus technicus für diejenigen Spezifikationen, die von nationalen oder internationalen *staatlichen* Einrichtungen, z.B. dem Deutschen Institut für Normung (DIN) oder der International Standards Organisation (ISO), verabschiedet worden sind und deshalb offiziellen, normativen Charakter haben. Daher sollte in allen anderen Fällen statt dessen von einer Spezifikation oder einer Empfehlung gesprochen werden[1]. Aus einer solchen Spezifikation kann natürlich nachträglich ein Standard werden, wenn sie bei einer entsprechenden Einrichtung eingereicht und von dieser verabschiedet wird.

[1]Das W3C verwendet aus eben diesem Grund für von ihren Gremien verabschiedete Arbeiten die Bezeichnung "Recommendation".

3.1.1 ISO 13250:2003 Topic Maps

Topic Maps, zu deutsch etwa *Themenkarten*, sind eine Technik zur Modellierung von Konzepten, Objekten und deren Beziehungen für eine bestimmte Domäne. Diese Karten haben die Form eines ungerichteten Graphen.

Die Wurzeln von Topic Maps lassen sich bis zur Davenport-Group zurück verfolgen[2]. Dabei handelte es sich um ein unter anderem vom Verlag o'Reilly begründetes Forum für Hersteller von Computer-Fachliteratur. Eine der ersten Aufgaben, der sich die Davenport Group zuwandte, war die Generierung so genannter Master Indices. Dabei handelt es sich um Indices, die eine Schlagwortsuche über mehrere Bücher hinweg erlauben und zwar auf der Grundlage der Indices der einzelnen Bücher. Diese vermeintlich einfache Aufgabe entpuppte sich als überraschend schwierig, da sich die einzelnen Indices nicht problemlos zusammen führen ließen. Die Verwendung von Synonymen, vorkommende Homonyme und die verschiedenen Detaillierungsgrade in der Indexgestaltung stellten die Verlage vor große Probleme.

Als Lösung für dieses Problem wurden Topic Maps entwickelt. Schnell wurde klar, dass sich diese Technik auch außerhalb des Verlagswesens einsetzen ließe, woraufhin bei der International Standards Organisation (ISO) der Prozess der Standardisierung angestoßen wurde. Im Jahr 2000 verabschiedete die International Standards Organisation Topic Maps als Standard 13250:2000 [62]. Der Standard definierte zur Notation HyTM, eine auf HyTime [61] basierende SGML-Sprache. Schon damals war abzusehen, dass im Internet SGML von XML verdrängt werden würde. Um eine XML-Notation für Topic Maps schnell entwickeln zu können, taten sich die Mitglieder des Gremiums für ISO 13250 unter dem Namen TopicMaps.org zusammen.

Im Jahr 2001 wurde dann die erste Version von XTM (für XML Topic Maps) fertig gestellt und anschließend als Ergänzung in den Standard aufgenommen. Mittlerweile gibt es eine neue Fassung des Standards (nunmehr ISO 13250:2003), in der HyTime zu Gunsten von XTM aus dem Standard entfernt worden ist, da sich gezeigt hat, dass HyTime keine Verwendung findet.

Der Aufbau des Standards

Der Standard definiert nur einige wenige Konstrukte, die zur Erstellung einer Topic Map ausreichen. Drei Grundbausteine werden für die Erstellung von Topic Maps benötigt: Topics, Associations und Occurrences. Dazu kommen noch zwei weitere Konstrukte (Scopes und PSI), die zur Erweiterung der Möglichkeiten der Grundbausteine verwendet werden. Nachfolgend werden zuerst die einzelnen Bestandteile des Standards beschrieben, bevor detaillierter auf ihr Zusammenspiel eingegangen wird.

[2]Die besonders in der Linux-Welt oft verwendete DocBook-Spezifikation [41] ist ebenfalls eine Entwicklung der Davenport Group.

Topic Zur Definition von Topics gibt es ein sehr treffendes Zitat von Steve Pepper, einem der Autoren des Standards:

> A topic, in its most generic sense, can be any thing whatsoever a person, an entity, a concept, really anything regardless of whether it exists or has any other specific characteristics, about which anything whatsoever may be asserted by any means whatsoever. [88]

Demzufolge kann ein Topic zur Modellierung *irgend eines* Themas verwendet werden, über das *irgend etwas irgend wie* ausgesagt werden soll. Konform zu dieser ultimativ weit gefassten Definition sind auch die weiteren Charakteristika der Topics. So können einem Topic beliebig viele Namen zugeordnet werden, wodurch verschiedene Varianten von Namen abgebildet werden können.

Topics können typisiert werden, wobei die Typisierung ebenfalls durch Topics vorgenommen wird. Ein Topic kann Instanz beliebig vieler Typen sein, wodurch sich sehr komplexe Hierarchien abbilden lassen.

Für den Verweis auf Ressourcen, die das Thema des Topics näher beschreiben, werden die so genannten Occurrences eingesetzt, die in beliebiger Zahl einem Topic zugewiesen werden können.

Association So wie mit Topics beliebige Themen modelliert werden, so dienen Associations dazu, Beziehungen zwischen einzelnen Topics abzubilden. Ähnlich wie bei Topics gilt auch hier, dass diese Beziehungen jeder beliebigen Form und Aussagekraft sein können. An einer Association sind zwei oder mehr (nicht zwingend verschiedene) Topics beteiligt, wobei jedem teilnehmenden Topic eine Rolle zugewiesen wird. Eine Rolle ist eine spezielle Form der Typisierung von Topics und erlaubt es, innerhalb einer Association fest zu halten, in welcher Funktion ein Topic an einer Association beteiligt ist.

Associations haben keine eigenen Namen. Statt dessen wird der Typ einer Association durch ein Topic festgelegt, auf das die Association sich bezieht. Ihre Namen gewinnt die Association damit durch dieses Topic.

Occurrence Eine Occurrence ist ein Verweis auf eine elektronisches Quelle weiterer Informationen zu einem Thema, das durch ein Topic reifiziert wird. So ein Verweis ist üblicherweise ein Hyperlink auf im Web erreichbare Ressourcen. Prinzipiell kann eine Occurrence aber auch direkt Textinformation aufnehmen, ein Datum oder eine Zusammenfassung eines Dokuments könnte also unmittelbar in einer Occurrence abgelegt werden. Ein Topic kann beliebig viele Occurrences enthalten. Occurrences können typisiert sein, so dass verschiedene Vorkommenstypen besser unterschieden werden können.

Scope Der Begriff des Scope lässt sich sinngemäß mit Gültigkeitsbereich übersetzen. In der Welt der Topic Maps dienen diese Gültigkeitsbereiche für zwei sehr wichtige

Aufgaben: So können sie zur Gruppierung gleichartiger Daten eingesetzt werden, wodurch die Übersichtlichkeit von großen Topic Maps gesteigert wird. Wichtiger jedoch ist ihr Nutzen bei der Auflösung von Homonymen. In einer Topic Map lassen sich mit Hilfe der Scopes beliebig viele verschiedene Gültigkeitsbereiche für Topics und Associations definieren, so dass genau unterschieden werden kann, wann ein Topic in welchem Kontext benutzt wird und welche Verbindungen zu diesem Kontext gehören. Das ist eine Fähigkeit, die praktisch allen aktuellen Suchmaschinen fehlt und deren Fehlen maßgeblich für die große Anzahl irrelevanter Suchergebnisse verantwortlich sein dürfte.

Published Subject Indicator (PSI) Ein PSI ist ein Verweis auf ein andernorts definiertes Thema, der an ein Topic in einer Topic Map angehängt werden kann. Damit ist für Betrachter - und wichtiger, für Computer - klar gestellt, dass an der angegebenen Stelle weitere Informationen zu diesem Thema zu finden sind. Diese PSI unterscheiden sich von Occurrences in der Hinsicht, dass PSI einen deklarativen Charakter haben. Das bedeutet, dass genau das Thema in dem Topic gemeint ist, das durch die Daten an der Stelle, auf die der PSI zeigt, definiert ist. Diese Deklarationen können genutzt werden, um bei der Vereinigung verschiedener Topic Maps Duplikate zu vermeiden. Dann werden alle die Topics zu einem neuen Topic vereinigt, die den gleichen PSI benutzen, unabhängig davon, ob sie gleich benannt sind.

Als Quelle für solche PSI bieten sich Verweise auf Standards von DIN oder ISO an. Eine weitere reichhaltige Quelle stellen Normdateien, wie sie von Bibliotheken erstellt und gepflegt werden, dar. Darin sind zum Beispiel Personen, Werke oder Orte mit ihren verschiedenen Schreibarten verzeichnet.

Der Prozess der Einigung auf weltweit eindeutige PSI ist noch nicht sehr weit gediehen. Bis dato existieren lediglich solche PSI-Sammlungen für Nationen und Sprachen, basierend auf den entsprechenden ISO-Standards (ISO 3166 für Nationen und ISO 639 für Sprachen).

Aus diesen Bausteinen werden Topic Maps aufgebaut. Zusätzlich definiert der Standard noch ein Verfahren, das beim Zusammenführen zweier Topic Maps zu einer neuen anzuwenden ist. Dieses (auf englisch mit *merge*) bezeichnete Verfahren sorgt dafür, dass die neu geschaffene Karte alle Informationen der beiden alten enthält. Dies ist von besonderer Bedeutung, wenn sich die beiden alten Topic Maps mit ähnlichen oder gleichen Themen beschäftigt haben. Dann ist davon auszugehen, dass nicht alle Themen in der gleichen Art und der gleichen Tiefe bearbeitet worden sind, also Unterschiede in der Struktur bestehen.

Das im Standard festgelegte Verfahren setzt an den Topics an, um die Vereinigung durchführen zu können. Dabei sind gleiche Topics zu identifizieren, da diese zu einem Topic in der resultierenden Map verschmolzen werden. Für diese Verschmelzung sind zwei Regeln definiert. Topics werden zusammengeführt, wenn sie mindestens einen Namen in einem Scope teilen, bzw. wenn sie den gleichen PSI enthalten. Die zweite Regel erlaubt die sichere Verschmelzung zweier Topics, wohingegen es bei der ersten nicht ausgeschlossen

Abbildung 3.1: Topics der Topic Map

werden kann, dass es zu falschen Ergebnissen kommt. Das ist ein weiterer Grund, warum die Einrichtung möglichst vieler allgemein bekannter PSI so wichtig ist.

Eine Beispiel für Topic Maps

Zur näheren Erläuterung der vorgestellten Elemente und ihres Zusammenspiels wird an dieser Stelle beispielhaft eine Topic Map aufgebaut. Die Topic Map wird sich mit Ludwig van Beethoven beschäftigen[3]. In ihr sollen die folgenden Aussagen modelliert werden:

1. Ludwig van Beethoven ist in der Stadt Bonn geboren worden.
2. Ludwig van Beethovens Geburtshaus steht in der Bonngasse, Hausnummer 20, in Bonn.
3. Ludwig van Beethoven war Komponist.
4. Ludwig van Beethoven ist von Bonn nach Wien gezogen.
5. Eine der Kompositionen Ludwig van Beethovens ist die Oper mit dem Namen *Fidelio*.
6. In Bonn gibt es eine Oper.

Diese sechs Aussagen bilden die Topic Map. In einem ersten Schritt werden die möglichen Topics identifiziert. Es sind dies *Ludwig van Beethoven, Stadt, Bonn, Geburtshaus, Bonngasse 20, Komponist, Wien, Komposition, Oper* und schließlich *Fidelio*.

[3]Diese Topic Map erhebt keinen Anspruch auf Vollständigkeit. Über Ludwig van Beethoven und sein Schaffen gibt es sicherlich noch viele andere interessante Aussagen, die in einer Topic Map modelliert werden könnten.

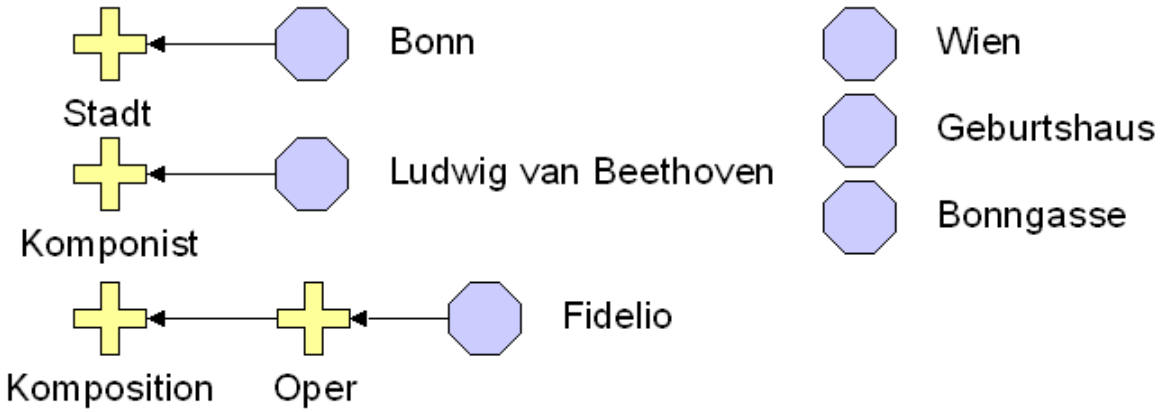


Abbildung 3.2: Typisierte Topics

Abbildung 3.1 zeigt einen ersten Blick auf die Topic Map, so wie sie sich nach dem ersten Schritt darstellt.

Im zweiten Schritt werden die Topics typisiert, soweit das im Kontext der Topic Map angebracht ist. Die erste, dritte und fünfte Aussage enthalten solche Typisierungen. So ist Bonn vom Typ Stadt, Beethoven ist vom Typ Komponist, eine Oper ist vom Typ Komposition und Fidelio ist vom Typ Oper. Abbildung 3.2 zeigt die Topic Map, nachdem die Typisierungen eingefügt worden sind.

Bisher sind in der Topic Map noch keine Associations vorhanden, die Topics sind, abgesehen von der Typisierung, noch nicht semantisch miteinander vernetzt. Daher werden im dritten Schritt die Associations hinzugefügt, die sich aus den Aussagen herleiten lassen. So lässt sich aus der ersten Aussage die Association *geboren in* herleiten, die Ludwig van Beethoven und Bonn miteinander verbinden. Das als Stadt typisierte Topic Bonn übernimmt dabei die Rolle der Geburtsstadt, während Beethoven für diese Association die Rolle einer Person einnimmt.

Die zweite Aussage verbindet das Geburtshaus von Beethoven mit dessen Adresse. Hieraus lässt sich eine Association bilden, die Beethoven, das Geburtshaus, die Stadt Bonn und die Straßenangabe miteinander verbindet. Berücksichtigt man allerdings die bereits erstellte Association *geboren in*, so zeigt sich, dass diese beiden Associations sich in ihrer Aussage ähneln, ohne wirklich wesentlich unterschiedliche Aspekte zu berücksichtigen. Deshalb werden sie zu einer einzigen Association zusammen gefasst: An der Association *geboren in* nehmen drei Topics teil, nämlich *Bonn*, *Beethoven* und *Bonngasse 20*. Die ersten beiden Topics behalten ihre Rolle in der Verbindung, das neu Topic erhält die Rolle Geburtshaus. Auf diese Weise ist das Topic Geburtshaus überflüssig geworden, seine Semantik steckt nunmehr in der Rolle der Association.

Die vierte Aussage lässt sich sehr einfach in einer Association zwischen Bonn, Wien und Beethoven ausdrücken. Diese trägt den Namen *gezogen nach* und enthält Ludwig van

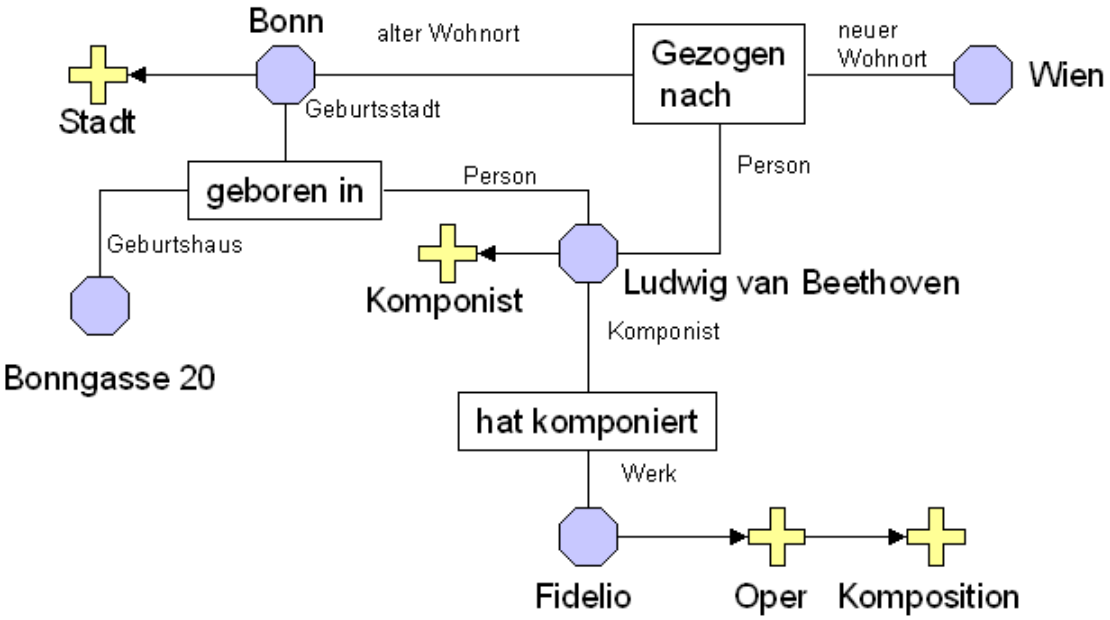


Abbildung 3.3: Associations in der Topic Map

Beethoven in der Rolle Person, Bonn in der Rolle des alten Wohnorts und Wien in der Rolle des neuen Wohnorts. Die fünfte Aussage lässt sich sehr einfach in einer Association *hat komponiert* formulieren, mit Beethoven in der Rolle des Komponisten und Fidelio in der Rolle des Werks. Abbildung 3.3 zeigt die Topic Map mit den neu angelegten Associations. Es ist anzumerken, dass die in den Associations verwendeten Rollen ebenfalls Topics sind und in der Topic Map definiert werden müssen. Dies ist in diesem Beispiel allerdings aus Gründen der Übersichtlichkeit unterblieben. Ebenso verhält es sich mit den Associations. Die Namen der Associations sind eigentlich Topics, die zur Typisierung der Associations verwendet werden. Auch diese sind aus Gründen der Übersichtlichkeit nicht gesondert definiert worden.

Die letzte Aussage ist bisher noch nicht behandelt worden. Darin wird der Fakt ausgedrückt, dass sich in Bonn eine Oper befindet. Dieser Sachverhalt ließe sich recht einfach in einer Association ausdrücken, allerdings ist das Konzept der Oper bereits in einem anderen Zusammenhang verwendet worden, so dass es hier zu Verwirrungen kommen könnte. Zur Auflösung des Homonyms bietet sich daher zusätzlich zu der Association die Verwendung von Scopes an, die die beiden Bedeutungen des Konzepts voneinander trennen. Für unser Beispiel bedeutet das die Einführung zweier Scopes für das Konzept der Oper: Musik und Bauwerk. Zusätzlich wird die Association *befindlich in* geschaffen, die Oper und Bonn unter dem Scope Musik miteinander verbindet.

Daneben müssen auch die Typbeziehungen zu Fidelio und Komposition angepasst werden, damit es nicht zu unangenehmen Nebeneffekten kommt. Abbildung 3.4 zeigt die nun-

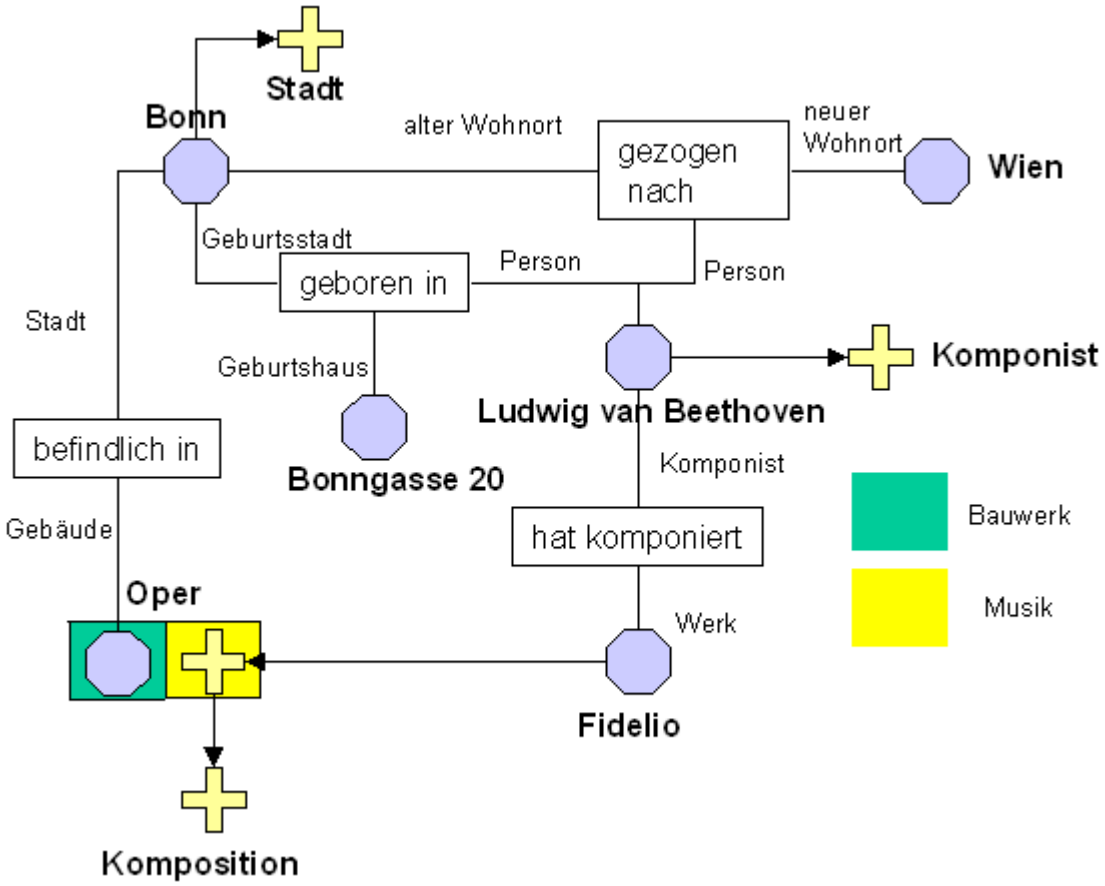


Abbildung 3.4: Die fertige TM mit Scopes

mehr fertig gestellte Topic Map, in der alle Fakten aus dem Beispiel eingetragen sind. Die komplexe Struktur schon so einer einfachen Topic Map verdeutlicht die Wichtigkeit effizienter Darstellungen, damit die Technik im täglichen Einsatz genutzt werden kann.

3.1.2 Resource Description Framework (RDF)

Das Resource Description Framework (RDF) nimmt innerhalb des W3C bei den Plänen für das Semantic Web eine entscheidende Stellung ein: RDF ist die Sprache, mit der die im Netz verfügbaren Ressourcen mit Metadaten versehen werden sollen. Alle weiteren Entwicklungen im Semantic Web nutzen diese Metadaten. Diese fundamentale Stellung wird vom *W3C Semantic Web Activity Statement* aus dem Jahr 2001 untermauert [1]. Dort heisst es zur Rolle von RDF:

> The Resource Description Framework (RDF) is a language designed to support the Semantic Web, in much the same way that HTML is the language that helped the original Web. RDF is a framework for supporting resource description,

> or metadata (data about data), for the Web. RDF provides common structures that can be used for interoperable XML data exchange.

Dieser Absatz zeigt die Erwartung, die an RDF gestellt wird, nämlich die gleiche Rolle im Semantic Web zu übernehmen, die HTML für das World Wide Web hatte. Der letzte Satz dokumentiert die Einbettung von RDF in bereits verabschiedete Empfehlungen des W3C. Dadurch lässt sich RDF mit bestehenden Werkzeugen erzeugen und verarbeiten, wodurch sich die Akzeptanz von RDF erhöht. Tatsächlich findet RDF schon vielerorts Verwendung, zum Beispiel bei Mozilla im Web Browser Firebird und dem Mail Client Thunderbird zur Verwaltung der Benutzerprofile, in Produkten von Adobe zur Einbettung von Metadaten in erzeugten Dateien oder in der Form von RSS (RDF Site Summary) zur Bereitstellung von Artikeln zum automatischen Download.

Der erste Working Draft zu RDF ist bereits 1997 erschienen und 1999 zu einer Empfehlung des W3C geworden, der so genannten *RDF Model and Syntax Specification*. Diese Empfehlung wurde vielfach als zu unübersichtlich angesehen, so dass in den folgenden Jahren daran gearbeitet wurde, eine Überarbeitung der Spezifikation zu erstellen. Nicht zuletzt, um Anforderungen der Gremien zu erfüllen, die sich mit den weiterführenden Aspekten des Semantic Web befassten. Dieser Prozess hat mit der Verabschiedung der neuen RDF-Empfehlung am 10.2.2004 seinen Abschluss gefunden. Das Ergebnis ist nunmehr auf sechs verschiedene Dokumente aufgeteilt, die sich jeweils mit einem bestimmten Aspekt von RDF auseinander setzen:

RDF Concepts and Abstract Syntax Dieses Dokument definiert die Grundlagen von RDF. Nach einer Erläuterung der Designziele wird der RDF-Graph definiert, der das Datenmodell von RDF darstellt. Zu diesem Graphen wird eine abstrakte Syntax definiert. bildet [65].

RDF Semantics Dieses Dokument definiert die Semantik der einzelnen Bestandteile des Datenmodells von RDF [44].

RDF/XML Syntax Specification (revised) In diesem Dokument wird die Syntax von RDF/XML, der vom W3C bevorzugten Notation für Inhalte in RDF, definiert [42].

RDF Vocabulary Description Language 1.0: RDF Schema Dieses Dokument definiert RDF Schema. Diese Sprache wird dazu eingesetzt, sogenannte RDF Vokabulare zu definieren und die Beziehungen der Bestandteile solcher Vokabulare untereinander fest zu legen [17]. Aufgrund ihrer besonderen Bedeutung wird sie in 3.1.3 gesondert behandelt.

RDF Primer Dieses Dokument ist dafür gedacht, einen schnellen Überblick über RDF zu geben. Beispiele und Hinweise zu Einsatzmöglichkeiten von RDF finden sich ebenfalls. Der RDF Primer ist kein normatives Dokument [72].

RDF Test Cases Dieses Dokument enthält eine Sammlung von Problemen mit der ursprünglichen Version von RDF und die dazu entwickelten Lösungen der Arbeitsgruppe. Damit eignet sich das Dokument insbesondere dazu, bereits existierende Implementierungen von RDF mit diesen Problemfällen zu testen und entsprechend der neuen Spezifikationen anzupassen [50].

Das Design von RDF

Das Aufgabe von RDF ist - wie bereits im voran gegangenen Abschnitt erwähnt - Strukturen für die Beschreibung von Ressourcen im Web bereit zu stellen. Beim Design der Sprache galt es, eine Reihe von Anforderungen zu erfüllen. Neben einigen technischen Anforderungen, die im Wesentlichen mit der Kompatibilität zu sowie der Nutzung bereits vorhandener Empfehlungen zu tun haben, waren dies die folgenden (siehe [65], Kapitel 2.2):

1. Jeder soll die Möglichkeit haben, beliebige Aussagen über beliebige Ressourcen machen zu können.
2. Die Benutzung eines einfachen Datenmodells.
3. Die Möglichkeit, eigene Vokabularien definieren zu können.
4. Eine formale Semantik und sowie Inferenzmöglichkeiten.

Die erste Anforderung dokumentiert den Anspruch von RDF: Jedem sollen beliebige Aussagen über beliebige Ressourcen möglich sein. Diese Anforderung ist angesichts des offenen Systems Internet, in dem RDF eingesetzt werden soll, sehr sinnvoll. So werden in RDF selbst keine Annahmen getroffen, was eine vollständige oder richtige Beschreibung einer Ressource ausmacht. Darauf wird in der Empfehlung explizit hingewiesen ([65] Kapitel 2.2.6):

> RDF does not prevent anyone from making assertions that are nonsensical or inconsistent with other statements, or the world as people see it. Designers of applications that use RDF should be aware of this and may design their applications to tolerate incomplete or inconsistent sources of information.

Die zweite Anforderung ist ebenso eine Folge aus dem geplanten Einsatzgebiet Internet. Da im Vorfeld keine Annahmen darüber getroffen werden können, welche Daten mittels RDF erfasst werden sollen, die Einschränkung der möglichen Daten für Anwendungen aber unabdingbar ist, muss es eine Möglichkeit geben, solche Einschränkungen für eine gegebene Anwendung zu definieren. Dafür ist RDF Schema entwickelt worden. Die dritte Anforderung soll sicher stellen, dass Anwendungen große Mengen von Daten in RDF schnell erzeugen und verarbeiten können. Die letzte Anforderung schließlich dient dazu, eine sichere

Grundlage für weitere Bausteine des Semantic Web zu schaffen. Durch die formale Semantik wird es möglich, mit logischen Kalkülen auf den Aussagen in RDF zu arbeiten um z.B. Aussagen per Inferenz zu erschließen.

Das Datenmodell von RDF

Ausgangspunkt für das Datenmodell von RDF ist die Annahme, dass sich jede Aussage als ein Tripel aus Subjekt, Prädikat und Objekt (oder auch Subjekt, Eigenschaft, Wert) ausdrücken lässt. Das Subjekt legt fest, worüber gesprochen wird, das Prädikat bestimmt, über welchen Aspekt des Subjekts gesprochen wird und das Objekt ist der Wert, der zu diesem Prädikat gehört. Eine Aussage in RDF ist genau solch ein Tripel und eine Beschreibung einer Ressource ist eine Menge von RDF-Tripeln. Zu einem Subjekt kann es beliebig viele Aussagen mit dem gleichen Prädikat geben, RDF sieht eine Einschränkung der Kardinalität von Prädikaten nicht vor.

Die formale Repräsentation von RDF-Tripeln erfolgt über gerichtete Graphen. Dabei werden Subjekt und Objekt als Knoten dargestellt, die über das Prädikat, dargestellt als gerichtete Kante von Subjekt zum Objekt, miteinander verbunden sind. Abbildung 3.5 zeigt die Struktur des einfachsten RDF-Graphen.

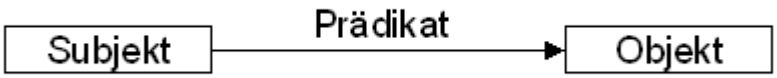


Abbildung 3.5: Graph eines RDF-Tripels

Für die Bestandteile eines RDF-Graphen gibt es einige Einschränkungen. Es werden drei verschiedene Arten von Knoten unterschieden: Solche, die einen URI (Uniform Resource Identifier [15]), ein Literal oder nichts enthalten (sogenannte leere Knoten). Kanten sind grundsätzlich gerichtet und immer mit einem URI belegt. Das Subjekt einer Aussage enthält entweder einen URI oder ist ein leerer Knoten. Letztere haben in RDF eine besondere Aufgabe: Aufgrund der gewählten Struktur lassen sich lediglich binäre Beziehungen abbilden. Es gibt aber viele Fälle, die sich nicht auf diese Weise hinreichend beschreiben lassen. Um solche Sachverhalte adäquat abbilden zu können, sind leere Knoten eingefügt worden. Zur Verdeutlichung soll ein Beispiel dienen, das im Abschnitt über Topic Maps schon verwendet wurde: Der Umzug Beethovens von Bonn nach Wien.

Diese Aussage lässt sich so nicht direkt in RDF tätigen, da sich diese Aussage nicht in das Subjekt - Prädikat - Objekt - Schema pressen lässt. Man kann sie allerdings in die beiden Aussagen “Beethoven zog_von Bonn“ und “Beethoven zog_nach Wien“ mit den beiden Prädikaten zog_von und zog_nach zerlegen (siehe Abb. 3.6). Allerdings geht auf diese Weise der Zusammenhang dieser beiden Aussagen verloren, denn Beethoven ist in

seinem Leben sehr häufig umgezogen, in Bonn allein dreimal. Wünschenswert wäre also ein Mechanismus zum Erhalt des Zusammenhangs der der beiden Aussagen.

Dies lässt sich in RDF mit Hilfe von leeren Knoten erreichen. In unserem Beispiel wird ein leerer Knoten eingefügt, der den Wert eines neuen Prädikats namens Umzug darstellt. Der leere Knoten wiederum ist Subjekt von zwei Prädikaten Umzug_von und Umzug_nach. Abb. 3.7 zeigt den RDF-Graphen dieser neuen Situation. Der Zusammenhang der einzelnen Komponenten des Umzugs von Beethoven bleibt erhalten; den beiden Prädikaten könnten sogar noch weitere hinzu gefügt werden, die diesen Umzug zusätzlich beschreiben helfen.

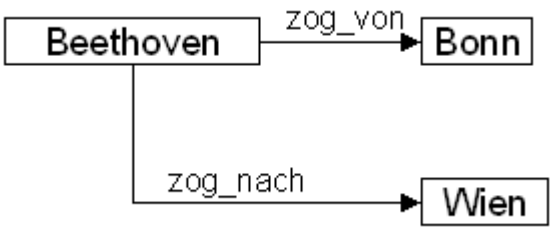


Abbildung 3.6: Auflösung der Aussage “Beethoven zog von Bonn nach Wien“ in RDF unter Verlust des Zusammenhangs

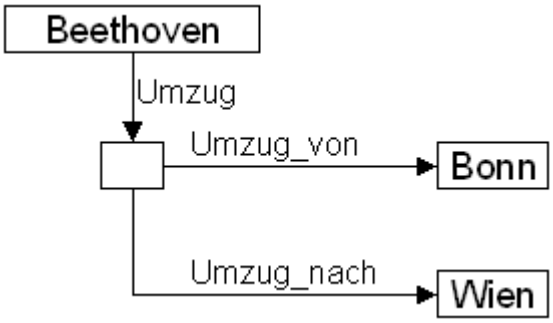


Abbildung 3.7: Auflösung der Aussage “Beethoven zog von Bonn nach Wien“ in RDF unter Verwendung eines leeren Knotens

Im geschilderten Datenmodell lassen sich schon sehr komplexe Aussagen tätigen. Die Ausdrucksstärke von RDF wird allerdings noch dadurch erhöht, dass es möglich ist, Aussagen über Aussagen zu tätigen. Dazu wird einem Tripel ein eigener URI zugeordnet, so dass es als Subjekt oder Objekt eines neuen Tripels verwendet werden kann. Dieser Vorgang wird *Reifikation* genannt. Dadurch lassen sich Aussagen selbst weiter spezifizieren oder in ihrer Bedeutung bewerten. Zum Beispiel kann der Urheber einer Aussage durch Reifikation mit dieser Aussage verbunden werden, so dass ersichtlich ist, wer sie getätigt hat.

Abschließend ist zu der Verwendung von URI zu bemerken, dass diese keine Adresse bezeichnen müssen, die im Internet tatsächlich angesteuert werden kann. Das erscheint

zunächst widersprüchlich, tatsächlich aber ist die einzige Anforderung an einen URI, dass eine Ressource durch ihn eindeutig referenziert wird. Das sich diese Ressource mittels des URI ansteuern lässt, ist nicht verlangt. Dafür gibt es URN (Universal Resource Names) bzw. URL (Uniform Resource Locators). Das ist besonders für den Fall wichtig, dass Aussagen über nicht digital vorliegende Ressourcen gemacht werden sollen. So können Menschen zwar über einen URI identifiziert, aber nicht zwingend auch gefunden werden.

3.1.3 RDF Schema (RDFS)

RDF ist bewusst so entwickelt worden, dass sich damit beliebige Aussagen formulieren lassen. Für viele praktische Anwendungen stellt das jedoch eher ein Hindernis dar: Eben weil mit RDF jede Aussage möglich ist, kann nicht ausgeschlossen werden, dass falsche oder sinnlose Aussagen an eine Applikation weiter gegeben werden. Weiterhin werden in RDF Ressourcen nicht nach Typen unterschieden, daher ist es nicht möglich, Prädikate zu definieren, die hinsichtlich der erlaubten Subjekte und Objekte eingeschränkt sind. Diese Einschränkung ist aber erforderlich, um Begriffshierarchien oder komplexe Zusammenhänge definieren zu können. RDF Schema, kurz RDFS, ist entwickelt worden, um diesem Problem zu begegnen.

RDF Schema ist eine Sprache, mit der Vokabulare für RDF definiert werden können. Diese sind als eigene RDF-Dokumente abgelegt. Dazu werden Mechanismen zur Verfügung gestellt, mit denen sich Klassen von Ressourcen und deren Beziehungen untereinander definieren lassen. Die Vorgehensweise in RDFS unterscheidet sich in einem wesentlichen Punkt von objektorientierten Ansätzen der Modellierung: Während in der objektorientierten Modellierung eine Klasse über ihre Eigenschaften definiert wird (z.B. die Klasse Buch hat als Eigenschaft Autor, vom Typ Person), definiert RDFS Eigenschaften über deren beteiligte Klassen (das Prädikat *hatAutor* hat als Subjekt die Klasse Buch, als Objekt die Klasse Person).

Der Vorteil dieser Art der Modellierung liegt in der Erweiterbarkeit. Wenn einem Vokabular neue Prädikate hinzu gefügt werden, so müssen die Klassendefinitionen nicht angepasst werden. Sie könnten sogar in anderen Vokabularen weiter verwendet werden.

RDFS führt das Konzept der Klasse ein. Auch die Bestandteile von RDF sind in eine Klassenhierarchie eingeordnet. Dadurch wird es möglich, neben Ressourcen auch Prädikaten Eigenschaften zu geben und diese in Hierarchien einzuordnen. Die Zuordnung von Ressourcen zu Klassen und die Ableitung von Klassen voneinander erfolgt über Prädikate, die RDFS zur Verfügung stellt. Außerdem erlaubt RDFS es, ein Prädikat bezüglich der erlaubten Subjekt- und Objektklassen einzuschränken. Dazu werden die beiden Prädikate rdfs:domain und rdfs:range zur Verfügung gestellt, mit denen sich die für die beiden Bereiche gültigen Klassen angeben lassen.

In diesem Zusammenhang sei darauf hingewiesen, dass für Prädikate gilt, dass wenn

zwei Ressourcen über ein Subprädikat verbunden sind, diese Ressourcen implizit auch durch das Superprädikat verbunden sind. Es wird hingegen weder geprüft, ob Domain und Range eines Subprädikats eine Spezialisierung von Domain und Range des Superprädikats sind, noch, ob sie überhaupt paarweise in Beziehung miteinander stehen. Für solche Einschränkungen wird eine semantisch aussagekräftigere Sprache wie OWL benötigt.

3.1.4 OWL - Web Ontology Language

OWL dient zur Definition von Ontologien auf der Basis von RDF und RDF Schema. Ein Zitat aus dem Überblick über die Bestandteile der OWL-Empfehlung zeigt sehr gut die Zusammenhänge mit diesen beiden Sprachen:

> RDF is a datamodel for objects („resources“) and relations between them, provides a simple semantics for this datamodel, and these datamodels can be represented in an XML syntax.
>
> RDF Schema is a vocabulary for describing properties and classes of RDF resources, with a semantics for generalization-hierarchies of such properties and classes.
>
> OWL adds more vocabulary for describing properties and classes: among others, relations between classes (e.g. disjointness), cardinality (e.g. „ëxactly one“), equality, richer typing of properties, characteristics of properties (e.g. symmetry), and enumerated classes.

OWL ist also im Wesentlichen als eine Erweiterung des zur Definition von Ontologien verwendbaren Sprachschatzes zu verstehen, mit der weitere Beschreibungen und Typisierungen von Classen und ihren Relationen möglich werden.

Die Empfehlung zu OWL teilt sich in sechs Unterdokumente:

OWL Overview Überblick über die Sprache sowie eine Übersicht über die in OWL enthaltenen Konstrukte [77]

OWL Guide Beispielbasierte Einführung in OWL sowie ein Glossar der in OWL eingeführten Begrifflichkeiten [96]

OWL Reference Systematische Beschreibung der Sprache und der Konstrukte [11]

OWL Semantics and abstract Syntax Formale und abstrakte Definition der Sprache. Dies ist das einzige normative Dokument der Empfehlungsreihe zu OWL [87]

OWL Test Cases Testfälle mit korrekter Lösung zur Überprüfung von OWL-Implementierungen [29]

OWL Use Cases and Requirements Sammlung von Anwendungsfällen für eine Web Ontologiesprache und daraus abgeleitete Anforderungen an OWL [59]

OWL-Sprachen

OWL besteht aus drei eigenständigen, aufeinander aufbauenden Untersprachen, die sich im Umfang der erlaubten Sprachkonstrukte voneinander unterscheiden. Aufsteigend sind dies OWL Lite, OWL DL und OWL Full. Die Unterschiede lassen sich wie folgt zusammenfassen:

OWL Lite unterstützt die Anlage von Klassenhierarchien mit einfachen Beschränkungen auf den Klassen. So sind Kardinalitäts-Einschränkungen auf Relationen möglich, allerdings nur mit den Werten 0 und 1. Ferner es ist möglich, die Wertebereiche von Relationen komplett auf bestimmte Arten von Klassen einzuschränken(*allValuesFrom*-Property), bzw. das Vorhandensein von Instanzen bestimmter Klassen im Wertebereich zu fordern (*someValuesFrom*-Property).

OWL DL unterstützt die Erstellung von Ontologien, für deren Aussagen sich Berechenbarkeit und Entscheidbarkeit sicherstellen lassen. Dazu enthält OWL DL alle Sprachbausteine von OWL, allerdings mit gewissen Einschränkungen. So kann eine Klasse in OWL DL zwar Subklasse verschiedener Klassen sein, nicht aber die Instanz einer anderen Klasse. Das DL steht für Description Logics, der formalen Grundlage von OWL.

OWL Full bietet die volle Ausdrucksstärke von OWL, ohne Beschränkungen aber auch ohne Garantie, dass sich die Schlüsse der Ontologie immer berechnen lassen. Eine Klasse in OWL Full kann gleichzeitig eine Sammlung von Instanzen, aber auch selbst eine Instanz sein. Mit OWL Full kann die Bedeutung der in RDF definierten Konstrukte angereichert werden. Es ist nicht mit einer Entwicklung von automatischen Reasonern für alle Features von OWL Full zu rechnen.

Die Aufgabe von OWL Lite wird in der Transformation bestehender Taxonomien oder Thesauri in die Welt des Semantic Web gesehen. OWL DL ist deutlich mächtiger, allerdings weiterhin entscheidbar, weswegen es auch automatische Reasoner für OWL DL gibt. In der Praxis sind denn auch überwiegend Ontologien anzutreffen, die höchstens die Komplexität von OWL DL aufweisen.

Mit OWL Full kann man selbst Bestandteile des RDF-Vokabulars einschränken und so die Semantik der ganzen Sprache ändern. Mit OWL Lite und OWL DL geht das nicht, in diesen Sprachen kann nur ein Subset der Einschränkungsmöglichkeiten angewendet werden. Daher ist zwar jedes OWL(Lite, DL, Full)-Dokument ein gültiges RDF-Dokument, umgekehrt gilt das im allgemeinen Fall aber nur für OWL Full. Daher ist beim Wechsel

von RDF zu OWL darauf zu achten, dass durch die Struktur des RDF-Modells nicht unbeabsichtigt der Einsatz von OWL Full notwendig wird. Soll ein RDF-Modell durch die Sprachen OWL Lite oder DL abgedeckt werden, ist daher sicherzustellen, dass die dort geltenden Beschränkungen von dem RDF-Modell eingehalten werden.

Sprachbestandteile

Nachfolgend werden die verschiedenen Bausteine aufgeführt, die OWL zum Modellieren von Ontologien zur Verfügung stellt. Die Gültigkeit für die verschiedenen Sprachstufen wird ebenfalls aufgeführt.

Neue Klassen OWL führt zunächst Klassen ein, die zur Grundierung beim logischen Schließen verwendet werden können: owl:Thing und owl:Nothing. Erstere ist die Oberklasse aller Klassen in OWL, alles ist von owl:Thing als der allgemeinsten Klasse abgeleitet. Auf der anderen Seite des Spektrums gibt es owl:Nothing, die speziellste Klasse, die Subklasse jeder anderen Klasse ist, eine Klasse ohne irgendeine Instanz. Zusätzlich werden Unterklassen von Property eingeführt, die Klassen ObjectProperty und DatatypeProperty. Eine ObjectProperty ist eine Relation zwischen Instanzen von Klassen, wohingegen eine DatatypeProperty eine Relation zwischen einer Instanz und einem Literal, einem XML-Datentyp beschreibt. Damit kann direkt bei der Relationsdefinition eine grobe Festlegung der zulässigen Werte der Relation getroffen werden. Diese vier Klassen sind bereits in OWL Lite verfügbar.

Gleichheit bzw. Ungleichheit von Klassen und Instanzen OWL enthält Relationen, mit denen Klassen oder Relationen als äquivalent (*equivalentClass*, *equivalentRelation*) deklariert werden können. Auf Instanzebene gibt es dazu ebenfalls eine Relation (sameAs), zusätzlich noch die Möglichkeit, Instanzen explizit als verschieden zu deklarieren (*differentFrom*, *allDifferent*). OWL DL und OWL Full können darüber hinaus Klassen mittels *disjointWith* als disjunkt deklarieren, sowie diese Einschränkung außer auf ObjectProperties auch auf DatatypeProperties anwenden.

Relationstypen OWL ermöglicht als erste Sprache im Semantic Web Stack weitere Relationstypen außer der einfachen gerichteten. So können Relationen als transitiv, symmetrisch, invers zu einer anderen, funktional (hat also höchstens einen Wert) oder eineindeutig (invers funktional) definiert werden. Damit sind für Reasoner weitreichende Möglichkeiten zum Erschließen impliziter Fakten gegeben. Ist zum Beispiel die Relation *hatPersonalausweisnummer* als eineindeutig definiert, so kann direkt geschlossen werden, dass erstens das Inverse davon höchstens einen Wert haben kann und zweitens, dass wenn mehrere Instanzen den gleichen Wert für diese Relation haben, es sich eigentlich um die selbe Instanz handeln

muss, diese also mit mehreren Instanzen im System vertreten ist (was ausdrücklich erlaubt ist). Diese Funktionalitäten sind bereits Bestandteil von OWL Lite.

Relationseinschränkungen Wie bereits in der Beschreibung von OWL Lite erwähnt, kann der Wertebereich von Relationen über lokale Einschränkungen weiter beschnitten werden. Dazu dienen die Relationen *allValuesFrom* und *someValuesFrom*. Erstere gibt vor, dass alle Werte aus der bestimmten Klasse stammen müssen, die letztere fordert mindestens einen Wert aus der angegebenen Klasse. Diese Beschränkungen gelten zusammen mit der globalen Beschränkung *rdfs:range* auf jeweils nur der Klasse, auf der sie definiert sind. Für OWL DL und OWL Full gibt es darüber hinaus noch die Einschränkung *hasValue*, die das Vorhandensein einer spezifizierten Instanz einer Klasse erfordert.

Klassenerstellung OWL erlaubt die Definition von Klassen über logische Mengenoperationen, mittels *intersectionOf*, *unionOf* und *complementOf*. Diese ermöglichen die Klassendefinition aus dem Schnitt, der Vereinigung oder der Restemengenbildung von Klassen. OWL Lite erlaubt nur die Klassenerstellung über den Schnitt mehrerer Klassen.

Darüber können in OWL DL und OWL Full die Instanzen von Klassen mittels *oneOf* vorgegeben werden.

Kardinalitätsbeschränkungen OWL DL und OWL Full erlauben die freie Wahl der Kardinalitäten von Relationen. Über die Angabe von minCardinality und maxCardinality in der Relationsdefinition kann ein Bereich, mit Cardinality sogar eine konkrete Anzahl festgelegt werden. Gültige Werte dafür sind alle nichtnegativen ganzen Zahlen.

Zusammenfassung

Mit der Web Ontology Language ist der letzte Baustein für die Ontologie-Modellierungssprachen gelegt. Auf der Basis von OWL lassen sich große Ontologien aufbauen, auch unterteilt in mehrere Module. OWL ist als Zusammenführung der zunächst parallel gestarteten europäischen und amerikanischen Bestrebungen um Ontologiesprachen DAML und OIL entstanden, wodurch eine große Akzeptanz der Sprache von Beginn an gewonnen wurde. OWL ist bereits im Einsatz und auch bisher in RDF oder RDFS formulierte Ontologien sollten dahingehend überprüft werden, ob eine Übertragung nach OWL (unter den bereits geschilderten Rahmenbedingungen) sinnvoll für die Wartbarkeit und den Einsatz von Reasoning Engines ist.

3.1.5 SPARQL

SPARQL ist die vom W3C designierte Anfragesprache für RDF. Sie ist seit Anfang Januar 2008 eine Empfehlung des W3C.

Die Spezifikation besteht aus drei Teilen:

SPARQL Query Language for RDF Die Syntax-Spezifikation der Sprache [89].

SPARQL Protocol for RDF Die Protokolldefinition für die Kommunikation zwischen Diensten, die SPARQL-Anfragen stellen oder auswerten [33].

SPARQL Query Results XML Format Die Definition des XML-Formats, in dem Ergebnisse ausgedrückt werden [13].

Syntax

SPARQL verwendet keine XML-Syntax, sondern lehnt sich in im Aufbau an SQL an. Nachstehend eine einfache SPARQL-Anfrage:

```
PREFIX dc: <http://purl.org/dc/elements/1.1>
SELECT ?title
WHERE
{
    <http://example.org/book/book1> dc:title ?title .
}
```

Diese Anfrage zerfällt in drei Teile: Den PREFIX-, den SELECT- und den WHERE-Block. Im PREFIX-Block werden Bezeichner definiert, die einen Teil eines URI repräsentieren, in diesem Fall wird dem Bezeichner **dc:** der Basis-URI des Dublin Core in der Version 1.1 zugeordnet. Im weiteren Verlauf der Anfrage kann damit der Bezeichner statt des URI verwendet werden, was die Lesbarkeit deutlich erhöht. Es darf beliebig viele PREFIX-Definitionen geben.

Der zweite Block gibt analog zum SELECT bei SQL an, welche Werte im Ergebnis enthalten sein sollen. Im Beispiel soll im Ergebnis nur ein Wertetyp angezeigt werden, nämlich die durch **?title** bezeichneten. Mit vorangestelltem Fragezeichen werden bei SPARQL freie Variablen gekennzeichnet, insofern kann zu dieser Zeit noch keine Aussage über die zu erwartenden Wert-Kategorien getroffen werden; anders bei SQL, wo im SELECT-Teil der Anfrage schon ersichtlich ist, aus welchen Feldern der Datenbank die zu berichtenden Werte stammen.

Im WHERE-Block wird die Anfrage mittels RDF-Tripeln in N3-Syntax spezifiziert. In diesem Fall ist dies die Suche nach dem Titel eines bestimmten Buches aus einem

RDF-Dataset. Die Anfrage besteht nur aus einem Tripel, die Ressource ist vorgegeben, ebenso die Eigenschaft der Ressource, die aus dem im PREFIX-Block definierten Vokabular stammt. An der Objektposition des Tripels ist die bereits im SELECT verwendete Variable eingesetzt, es wird also nach den möglichen Belegungen der Variable unter den gegebenen Rahmenbedingungen gesucht.

In einem WHERE-Block dürfen beliebig viele Tripel verwendet werden, die sich auch auf unterschiedliche RDF-Datasets beziehen dürfen. Das Einsetzen von Variablen ist an jeder Stelle eines Tripels möglich, ebenso das Verwenden von Variablen, die nicht Teil der im SELECT definierten Variablenmenge sind (Projektion).

Das Ergebnis der oben gezeigten Anfrage ist entweder eine leere Antwort, oder eine Liste mit den Titeln, die das angegebene Buch haben kann. Dies entspricht am ehesten den Antwortmengen, die SQL zurückgibt, folgerichtig wird in SPARQL ebenfalls von Result Sets gesprochen.

Weitere Anfragetypen Anstelle eines SELECT-Blocks können auch andere Blöcke verwendet werden, nämlich CONSTRUCT, ASK oder DESCRIBE. Wo SELECT die möglichen Belegungen der angegebenen Variablen als Ergebnis in einem festgelegten Format liefert (siehe unten), gibt CONSTRUCT einen RDF-Graphen zurück. Dafür muss in der Anfrage eine Reihe von Relationen spezifiziert werden, die im Ergebnis enthalten sein sollen, abhängig von weiteren Einschränkungen, die im WHERE-Block vorgenommen werden können.

ASK-Anfragen dienen dazu, herauszufinden, ob ein angegebenes Graph-Muster Teil der angefragten Ontologie ist. Die Antwort auf solche Anfragen ist entweder *yes* oder *no*, die evtl. vorhandenen Belegungen der Variablen sind also nicht Teil der Antwort.

Schließlich gibt es DESCRIBE-Anfragen. Diese liefern als Teil der Antwort weitere Informationen über die Ressourcen und berücksichtigen dabei evtl. in der Ontologie vorhandene Constraints auf den Datentypen und Relationen.

Weitere Blöcke und Modifikatoren Außer den bereits beschriebenen Blöcken gibt es außerdem ORDER-BY, zuständig für die Sortierung der Ergebnisse wie das gleichnamige Konstrukt in SQL. Diese können auf- oder absteigend sortiert werden, die Sortierrichtung lässt sich für jede der angegebenen Variablen einzeln festlegen. Die Verwendung von ORDER-BY ist nur für SELECT-Anfragen sinnvoll.

SELECT-Anfragen können unter Umständen Duplikate auswerfen, abhängig von den Rahmenbedingungen im WHERE-Block und der Art der ausgegebenen Variablen. Um dies zu unterbinden kann SELECT DISTINCT verwendet werden, wodurch Duplikate direkt vom Query Processor entfernt werden.

Mittels LIMIT und OFFSET kann die Anzahl der zurückgegebenen Ergebnisse beein-

flusst werden. LIMIT gibt die maximale Anzahl anzuzeigender Ergebnisse vor, wohingegen OFFSET dazu verwendet wird, die Wiedergabe ab einer bestimmten Position der Ergebnismenge einsetzen zu lassen. Die Verwendung von OFFSET zum Browsen der Ergebnisse einer Anfrage ist nur sinnvoll, wenn durch die gleichzeitige Verwendung von ORDER BY eine gleichbleibende Ordnung der Ergebnisse gewährleistet ist, denn die Abfrage muss für jeden neuen Teil der Ergebnisse wiederholt werden. Der Einsatz von LIMIT und OFFSET ist ebenfalls nur bei SELECT-Anfragen nützlich.

Zur weiteren Eingrenzung der Möglichkeiten im WHERE-Block gibt es die Möglichkeit, Filter zu definieren. Dazu wird ein FILTER-Block in den WHERE-Block eingefügt. Es ist eine Reihe sog. Tests vordefiniert, darunter größer/kleiner-Vergleiche, Sprach- und Literaltyp-Überprüfungen anhand der optionalen XSD-Typ-Definitionen, sowie die Möglichkeit, auf Literale reguläre Ausdrücke anzuwenden. Das macht Filter zu einem mächtigen Hilfsmittel zum Eingrenzen von Ergebnismengen.

Als letztes gibt es noch die Möglichkeit, optionale Bedingungen im WHERE-Block anzugeben. Diese werden zusätzlich zu den anderen Mustern überprüft, ihre Nichterfüllung führt aber nicht zum Ablehnen von Daten. Damit ergänzen sie höchstens die Ergebnismenge.

Einschätzung SPARQL ist eine auf die Graph-Struktur von RDF abgestimmte Anfragesprache. Trotz der Anleihen bei SQL steht hinter SPARQL ein anderes Konzept. Die Verwendung freier Variablen, die optionalen Anfragebedingungen und nicht zuletzt die Möglichkeit, zur Anfragezeit beliebige Quellen aus dem Internet zusammen abzufragen, gehen weit über die Möglichkeiten von SQL hinaus. Die Filtermöglichkeiten sind ebenfalls reichhaltig: Sind die Literale mit XSD-Datentypen versehen, so lassen sich darauf Testkriterien anwenden wie dies auch in einer relationalen Datenbank möglich wäre. Die Filterung über reguläre Ausdrücke eröffnet zusätzliche Möglichkeiten, die SQL nicht ohne den Einsatz von stored procedures bietet.

Der Preis dieser Mächtigkeit ist jedoch eine kompliziertere Syntax, das Zusammensetzen einer SPARQL-Anfrage ist Laien auf keinen Fall zuzumuten. Daher ist die Kapselung eines semantischen Suchdienstes in einer nutzerfreundlicheren Schnittstelle notwendig.

SPARQL-Protokoll

In der Protokoll-Spezifikation von SPARQL werden die Nachrichten definiert, die im Verlauf einer Anfrage zwischen einem anfragenden Service und einem Query Processor ausgetauscht werden können. Dazu wird zunächst die Schnittstelle abstrakt beschrieben. Diese Schnittstelle unterstützt eine Operation, nämlich *query*. Zu ihrer Durchführung wird das In-Out Message Exchange Pattern eingesetzt, d.h. es werden genau zwei Nachrichten im Verlauf einer Suchoperation verwendet: Eine Nachricht, in der die Anfrage und das zu verwendende RDF-Dataset beschrieben werden, wird von einer Stelle N an einen Query

Processor Q geschickt. Dieser verarbeitet die Anfrage und sendet entweder eine Antwort- oder eine Fehlernachricht zurück an N.

Das in der ersten Nachricht angegebene RDF-Dataset kann dabei aus einer beliebig großen Menge von RDF-Graphen bestehen, die vor der Verarbeitung in einem Graphen kumuliert werden. Wird kein Dataset angegeben, so kann der Query Processor ein Standard-Set annehmen, darf aber auch die Anfrage zurückweisen. Ebenso dürfen Anfragen zurückgewiesen werden, die dem Processor nicht genehme Dataset-Bestandteile haben, ein Processor muss also nicht alle Anfragen erfüllen, die ihm gestellt werden und erreichbare Datasets haben. Diese letzte Einschränkung dient dazu, Missbrauch öffentlich verfügbarer Query Processors zu vermeiden, da sie nicht für jedes x-beliebige Dataset Dienste anbieten müssen.

Das Protokoll kennt zwei Fehlermeldungen, nämlich MalformedQuery und QueryRequestRefused. Die erste Nachricht wird gesendet, wenn die Anfrage fehlerhaft ist und deshalb nicht bearbeitet werden kann, die zweite, wenn die Anfrage aus anderen Gründen zurückgewiesen werden muss.

Nach der abstrakten Definition werden noch zwei Implementierungen der Schnittstelle spezifiziert, eine für HTTP und eine für SOAP. Die Spezifikation für HTTP bindet die Nachrichten des SPARQL-Protokolls an die GET- und die POST-Operation des HTTP-Protokolls. Für die Fehlermeldungen MalformedQuery und QueryRequestRefused werden die HTTP-Statuscodes 400 (Bad Request) bzw. 501 (Internal Server Error) verwendet. Die SOAP-Implementierung verwendet SOAP-Envelopes, in denen die HTTP-Anfragen gekapselt sind.

Ergebnis-Format

Das Ergebnis-Format für SPARQL ist in XML spezifiziert. Es ist relevant für die Anfragetypen SELECT und ASK, die anderen beiden Anfragetypen antworten mit einem RDF-Dokument. Das Ergebnisformat sieht einen Header und einen Ergebnisteil vor. Der Header nennt die Feldnamen in der Reihenfolge ihres Auftretens im Ergebnis, optional noch einen Link auf ein Metadatendokument.

Der Ergebnisteil ist ein Container für Resultat-Objekte, pro Zeile des Result Sets ein Resultat. Darin enthalten sind die Variablenwerte für die jeweilige Zeile, versehen mit der Information über den Typ des Werts, also ob es sich um eine blank node, einen URI oder um Literale mit oder ohne angegebenen Typ handelt.

War die Anfrage vom Typ ASK, so ist der Header leer und der Ergebnisteil wird durch ein $<$ *boolean* $>$-Tag ersetzt, das entweder *true* oder *false* als Wert enthält.

3.2 Begriffserklärungen

3.2.1 Ontologien

In Kapitel 1 ist bereits das klassische Zitat von Gruber genannt worden, das sich in einer Vielzahl von Publikationen zum Thema Ontologien in der Informatik findet. Der größte Teil der zu diesem Thema Publizierenden schließt sich dieser Definition an. Interessanterweise sogar dann, wenn sie der Künstlichen Intelligenz, der die Definition entstammt, eher kritisch gegenüber stehen. So findet sich in einem Buch von Johan Hjelm im gleichen Zusammenhang dieses Zitat[4]:

> The AI crowd will tell you that ontologies will bring world peace, save the whales, and generally be the solution to everything. Just like they told you expert systems would do the same last year. An ontology is, of course, nothing that remarkable. It is simply a specification of a conceptualization.

Unabhängig vom Standpunkt bezüglich der Ausmaße des Mehrwerts, der sich mit Ontologien erzielen lässt, besteht also Einigkeit darüber, dass es sich dabei um die Spezifikation einer Konzeptualisierung einer bestimmten Domäne handelt. Gruber geht noch einen Schritt weiter und bezeichnet sie als *geteilte Spezifikation* (shared specification), also eine formalisierte Sicht der Welt, die von einer Gruppe von Personen geteilt wird. Mit dieser gemeinsamen Sicht hat man eine Basis geschaffen, auf der man sich über die Welt unterhalten kann. Dies gilt im speziellen Fall der Informatik auch für die Kommunikation von Menschen mit Maschinen, insbesondere aber für die Kommunikation verschiedener Softwaresysteme untereinander.

Ontologien können in verschiedene Klassen aufgeteilt werden, die Zuordnung einzelner Termsysteme ist dabei durchaus umstritten. In einem Artikel von Deborah McGuiness [39] wird eine Einteilung, ausgehend vom Detaillierunsgrad der Spezifikation von Termsystemen vorgenommen, die gemeinhin als Ontologien bezeichnet werden. Als einfachste Beispiele finden sich kontrollierte Vokabularien wie sie in einfachen Katalogen oder Glossaren Verwendung finden. Thesauri sind bereits eine Stufe komplexer, da in ihnen zumindest die Relation $Oberbegriff \longrightarrow Unterbegriff$ definiert ist, die eine lose Hierarchie der Begriffe induziert.

In der nächsten Stufe finden sich solche Systeme, die strikte Subklassen-Hierarchien erzwingen, in denen also Vererbung klar definiert ist. Ist B Subklasse von A und ist C Subklasse von B, so folgt in solchen Ontologien zwingend, dass C eine Subklasse von A ist.

Die nächste wesentliche Stufe von Ontologien ergänzt die Klassen um Informationen über ihre Eigenschaften (engl. properties, frames). Diese erweitern die Möglichkeiten der

[4] Siehe [60], Seite 202

Wissensmodellierung deutlich. Eigenschaften haben in diesem Modell ihren eigenen Vererbungsmechanismus, d.h. werden von Klassen an ihre Subklassen vererbt und können in diesen spezialisiert werden. Eine Erweiterung dieses Modells erlaubt die Restriktion von gültigen Werten für die Eigenschaften. RDFS steht als Beschreibungssprache zwischen diesen beiden Stufen der Ontologiedefinition, da zwar Wertebereiche für Eigenschaften festgelegt werden können, allerdings nicht, wieviele verschiedene Werte eine Eigenschaft annehmen darf, bzw muss. Das geht erst mit OWL.

Als Mindestanforderung an „echte“ Ontologien werden von McGuiness die drei folgenden Eigenschaften festgelegt:

- Ein endliches, kontrolliertes Vokabular
- Eindeutige Interpretation der Zusammenhänge zwischen Klassen und Termen
- Strenge Klassen-Subklassen-Beziehungen zwischen Klassen

Ontologien, die diese drei Bedingungen erfüllen, werden bei McGuiness *einfache Ontologien* genannt. Nach dieser Kategorisierung sind alle Termsysteme bis einschließlich der Thesauri keine Ontologien. Glossare nicht, da sie (zumindest Maschinen) keine eindeutige Interpretation der Zusammenhänge erlauben, Thesauri nicht, da dort keine strenge Klassen-Subklassen-Beziehung herrscht.

Darüber hinaus wird eine Reihe typischer Merkmale definiert, die allerdings nicht obligatorisch sind.

- Definition von Eigenschaften auf Klassenbasis
- Instanzen als Teil der Ontologie
- Wert-Restriktionen auf Klassenbasis

RDFS erlaubt die Definition von Ontologien, die diese Eigenschaften aufweisen können. Weitere Ausdrucksmöglichkeiten, die zum Teil beim Umstieg auf OWL hinzu kommen, werden in dem Artikel ebenfalls erwähnt, allerdings lediglich als “u.U. wünschenswert, aber nicht typisch“ bezeichnet. Dazu gehören die Definition disjunkter Klassen und die Möglichkeit zum Ausdruck spezieller Beziehungstypen wie *Inverses-von* oder *Teil-Ganzes* auf der Ebene der Ontologiebeschreibungssprache.

3.3 Algorithmen & Technologien

Dieser Abschnitt gibt kurze Erklärungen der wichtigsten Technologien und Algorithmen, die im Rahmen dieser Arbeit Verwendung finden. Der Ablauf von Algorithmen wird sche-

matisch erläutert, zu Begriffen eine Definition der Bedeutung im Rahmen der Dissertation gegeben.

3.3.1 Assoziationsregeln

Assoziationsregeln sind solche Regeln, die eine Implikation der Form $x \Rightarrow y$ ausdrücken. In dieser allgemeinsten Form sind sie aus der Aussagenlogik bekannt. Eine der bekannteren Anwendungen des Data Mining, die Warenkorbanalyse, beschäftigt sich mit der Erstellung genau solcher Assoziationsregeln aus der Analyse von Transaktionen. Hierbei sind x und y jeweils Platzhalter für verschiedene Waren, bzw. Mengen solcher Waren (englisch *Itemsets*). Für Verkäufer stellt das Wissen über Zusammenhänge beim Warenkauf eine interessante Information hinsichtlich der Preisgestaltung, evtl. anstehender Sonderaktionen und sogar der Anordnung der Waren innerhalb des Marktes dar.

Zwei Maße sind für das Aufstellen von Assoziationsregeln aus einer Menge von Transaktionen wesentlich:

Support Der *Support* einer Regel bezeichnet den Anteil derjenigen Transaktionen an der Gesamtmenge, in denen sowohl x als auch y vorkommen.

Confidence Die *Confidence* einer Regel hingegen bezieht sich auf die Teilmenge der Transaktionen, die x enthalten. Sie gibt den Prozentsatz der Transaktionen davon an, in denen ebenfalls y vorkommt.

Für eine Regel $Brot \Rightarrow Butter$ mit Support von 0,6 und Confidence von 0.7 gilt also, dass 80% aller Transaktionen Brot und Butter enthalten. Außerdem gilt, dass in 70% aller Transaktionen, in denen Brot erworben wurde, auch Butter gekauft worden ist.

Ein niedriger Support-Wert für eine Regel ist ein Hinweis auf zufälliges gemeinsames Auftreten der beteiligten Dinge, während ein niedriger Confidence-Wert auf eine schwache Korrelation der beteiligten Dinge hinweist. Damit ergibt sich für die praktische Anwendung die Notwendigkeit, Schwellwerte für die Akzeptanz einer Assoziationsregel vorzugeben. Diese Werte müssen allerdings je nach Anwendung angepasst werden, globale Empfehlungen gibt es hierfür nicht.

Der Algorithmus zur Bestimmung von Assoziationsregeln für eine Transaktionsmenge T besteht im Wesentlichen aus drei Schritten. Bei gegebenem Mindestwerten für Support σ und Confidence γ sind dies:

1. Das Finden von Mengen von Teilen, für die $Support \geq \sigma$ gilt.

2. Die Erzeugung von Kandidaten für Assoziationsregeln, indem jede dieser Mengen in

die zwei Untermengen x und y geteilt wird[5].

3. Die Berechnung der Confidence für jede der Kandidatenregeln. Jede Regel, für die $Confidence \geq \gamma$ gilt, ist eine der gesuchten Assoziationsregeln.

3.3.2 Der Apriori-Algorithmus

Der Apriori-Algorithmus [90] wird zur Bestimmung von Assoziationsregeln eingesetzt. Der Algorithmus wird in den folgenden Schritten durchgeführt:

1. Sammeln aller einelementigen Itemsets und Bestimmung ihres Supportwerts. Alle Mengen, deren Support unterhalb des geforderten Mindestwerts liegen, können im weiteren Verlauf ignoriert werden. Alle anderen seien Teil der Menge I_1.

2. Erzeuge für jede Menge i aus I_n die Mengen mit n+1 Elementen, die sich ergeben, wenn man i um ein Element aus I_1 erweitert. Jede dieser Mengen, die den benötigten Support aufweist, wird Teil von I_{n+1}. Iteriere, bis keine neuen Mengen mehr gebildet werden können, entweder, weil keine der neuen Mengen den nötigen Support aufweist oder die Menge aus allen Teilen aus I_1 gebildet worden ist.

Die aus diesem Algorithmus resultierenden Mengen in den jeweiligen Iterationen ergeben die Ausgangsmengen für die weitere Bestimmung von Assoziationsregeln. Bei der Implementierung des Algorithmus ist beachtenswert, dass nicht alle Möglichkeiten der Kombination getestet werden müssen. Ist z.B. bereits der Support für die Menge $\{A, B\}$ berechnet worden, muss dies nicht für $\{B, A\}$ wiederholt werden, da der Wert gleich sein wird. Hier kann also mit einer geeigneten Datenstruktur zum Nachhalten bereits ermittelter Ergebnisse Zeit gespart werden.

3.3.3 tf*idf

tf*idf ist ein Maß zur Bestimmung von Termgewichten in Vektorraummodellen des Information Retrieval. Dabei wird jedes Dokument einer Sammlung durch einen Wortvektor repräsentiert, einen einspaltigen Vektor, in dem jede Zeile einem Wort aus der Dokumentensammlung zugeordnet ist. Ist ein Wort nicht in einem Dokument enthalten, ist der Wert an der Stelle des Vektors 0. Ein naiver Ansatz zur Bestimmung der Termgewichte wäre, die reine Anzahl der Worte im Dokument zu nehmen, die so genannte Termfrequenz (tf).

Damit würden allerdings die häufig vorkommenden Terme eines Dokuments besonders großes Gewicht für den betreffenden Vektor erhalten, was insbesondere im Fall von Artikeln und Präpositionen nicht erwünscht ist. Außerdem bevorzugt dieses Vorgehen lange

[5]Damit ergibt sich als Randbedingung für die Mengen aus 1., dass sie mindestens zwei verschiedene Dinge enthalten müssen

Dokumente gegenüber kürzeren, diese würden also im Korpus an Sichtbarkeit verlieren. Ein letztes Argument gegen das Verwenden der reinen Termfrequenz liegt in der Informationstheorie begründet: Zur Unterscheidung verschiedener Dokumente eignen sich die Wörter besonders, die eine hohe lokale Bindung aufweisen, deren Vorkommen also auf einige wenige Dokumente beschränkt ist. Hierzu wird die inverse Dokumentfrequenz (idf) verwendet. Dazu wird die Anzahl der Dokumente, in denen ein bestimmtes Wort vorkommt, zur Gesamtanzahl der Dokumente im Korpus in Beziehung gesetzt, die sog. Dokumentfrequenz, und dieser Bruch dann invertiert. Sie ist für ein Wort also um so größer, je kleiner die Anzahl der verschiedenen Dokumente ist, in denen es vorkommt. Um das Wachstum dieses Werts zu begrenzen, wird er logarithmiert.

Um die inverse Dokumentfrequenz in die Berechnung des Termgewichts einfließen zu lassen, kann dieses mit der Termfrequenz multipliziert werden:tf*idf.

Die komplette Formel für die Bestimmung des Gewichts eines Worts w_i ist damit:

$tf * idf(w_i = tf_i * log(\frac{D}{df_i})$

tf_i bezeichnet dabei die Häufigkeit des Vorkommens von w_i im Dokument, D ist die Anzahl der Dokument im Korpus und df_i ist die Dokumentfrequenz von w_i.

Dieses Maß ist angreifbar durch gezieltes Wiederholen vermeintlich wichtiger Begriffe, das so genannte „Keyword Spamming“. Daher werden zumeist zusätzliche Techniken zur Normalisierung der Wortvektoren eingesetzt, z.B. indem die Termfrequenzen durch die höchste im Dokument vorkommende Termfrequenz tf_{max} geteilt wird. In diesem Fall wird tf_i durch $\frac{tf_i}{tf_{max}}$ ersetzt.

3.3.4 Wiki-Systeme

Ein Wiki-System ist ein Web Server, der Nutzern die Möglichkeit gibt, Informationen der enthaltenen Seiten (auch *Wikis* oder *Wiki Webs* genannt) zu ändern, zu ergänzen oder zu löschen - im laufenden Betrieb. Dazu wird eine einfache Schnittstelle angeboten, die das Erfassen und Verändern von Inhalten auch für die Nutzer ermöglichen, die über keine HTML-Kenntnisse verfügen. Diese niedrige Zugangsschwelle ist beabsichtigt, um einen möglichst hohen Grad der Interaktion der Nutzer zu erreichen. Darauf deutet auch die Bezeichnung selbst hin: Das Wort *wiki* stammt aus dem hawaiianischen und bedeutet dort *schnell, einfach*. Wiki-Systeme werden üblicherweise dazu eingesetzt, Informationen zu einem bestimmten Themengebiet zu sammeln und darzustellen. Zentraler Gedanke dabei ist, dass die an diesem Gebiet Interessierten selbst gemeinschaftlich die Daten einpflegen und komplettieren. In vielen solchen Systemen ist nicht einmal die Authentifizierung der einzelnen Nutzer notwendig, Änderungen können also auch anonym erfolgen. Zur Absicherung gegen fehlerhafte oder vorsätzliche Änderungen sind alle Dokumente in einem Wiki versioniert, alte Stände eines Dokuments können also jederzeit angesehen oder wieder hergestellt werden. Viele Wiki-Systeme bieten zudem ein Web-Forum an, in dem über die einzelnen

Dokumente diskutiert werden kann.

Die technischen Hilfsmittel von Wiki-Systemen mögen einfach sein, haben jedoch ein Ziel: Das Erzeugen einer Gemeinschaft rund um das Thema des Wikis. Die Selbstorganisationsfähigkeit der Gemeinschaft der Nutzer bestimmt entscheidend die Qualität des Wikis, weswegen Wiki-Systeme auch unter dem Begriff *social software* geführt werden.

Das bekannteste Wiki Web ist wahrscheinlich die Wikipedia[6], ein internationales Projekt zur kooperativen Erstellung von Enzyklopädien. Wikipedia enthält derzeit[7] unterschiedliche Enzyklopädien in 250 verschiedenen Sprachen, die sich allerdings zum Teil erheblich in ihrem Umfang unterscheiden. So hat die englische Version mit ca. 1,6 Millionen die meisten Artikel (die deutsche Enzyklopädie kommt auf ca. 530 000 Artikel), während die kleinste (auf Herero) nur einen einzigen Artikel aufweist. Die meisten Versionen haben weniger als 5 000 Artikel. Dabei ist aber zu berücksichtigen, dass es sich nicht um Übersetzungen handelt, sondern um eigenständige Fassungen, mit jeweils lokal relevanten Inhalten.

Auch wenn die Wikipedia in der Öffentlichkeit gern mit dem Begriff Wiki gleichgesetzt wird, so ist doch die Geschichte der Wiki-Systeme deutlich länger als die der Wikipedia, die es seit ca. 2001 gibt. Die Erfindung der Wiki-Systeme geht auf Ward Cunningham zurück, der 1995 die erste Software für Wiki-Systeme namens WikiWikiWeb schrieb und damit das erste Wiki aufsetzte[8]. Das Wiki existiert immer noch und sammelt Informationen über Software Design Patterns und Methoden des Software Engineering. Ward Cunningham ist außer für Wiki-Systeme auch für weitere Entwicklungen bekannt geworden: CRC (Class Responsibility Collaboration) Cards [35] und Extreme Programming [12], beide zusammen mit Kent Beck.

Heute gibt es eine Vielzahl verschiedener Wiki-Systeme, am bekanntesten dürfte MediaWiki sein, das System, in dem die Wikipedia gehostet wird. Eine Weiterentwicklung sind die semantischen Wikis, die in Kapitel 4.4 näher beschrieben werden.

[6] www.wikipedia.org
[7] Stand 1.2.2007
[8] Das Portland Pattern Repository, zu finden unter http://c2.com/cgi/wiki?WelcomeVisitors

Kapitel 4

Related Work

Dieses Kapitel widmet sich der Vorstellung wissenschaftlicher Arbeiten, die sich mit Themen beschäftigen, die mit dem Anliegen dieser Dissertation verwandt sind. Die Forschungsgebiete, die von dieser Arbeit berührt werden, sind in Abbildung 4.1 dargestellt. Dabei handelt es sich um die Gebiete Semantic Web, Text Mining und Wiki-Systeme. Die Teilbereiche dieser Gebiete sind direkt mit der Dissertation verbunden. Es handelt sich dabei um Ontologie-Lernsysteme, Relationserkennung, bzw. Relationslernen und semantische Wiki-Systeme. Jedem dieser Themengebiete ist ein eigenes Unterkapitel gewidmet, darüber hinaus werden gängige Ontologie-Editoren behandelt, sowie Arbeiten aus dem weiteren Umfeld der Dissertation in Unterkapitel 4.5 aufgeführt.

4.1 Ontologie-Lernsysteme

Die in diesem Abschnitt vorgestellten Systeme dienen zum (semi-)automatischen Lernen von Ontologien. Damit verfolgen sie das gleiche Ziel wie das in dieser Dissertation zu entwickelnde System. Zur besseren Illustration der Unterschiede werden die vorgestellten Systeme an den in Abschnitt 2.3.1 aufgestellten Anforderungen gemessen.

4.1.1 Text-To-Onto

Text-To-Onto [68, 69, 70, 71] von Alexander Maedche und anderen vom AIFB[1] an der Universität Karlsruhe ist ein sehr bekanntes und oft zitiertes System zum Ontologielernen aus einer gegebenen Textkollektion. Es ist als Erweiterung des OntoEdit-Frameworks entstanden, eines nicht mehr erhältlichen Produkts der OntoPrise GmbH[2], die ihrerseits aus dem

[1] Institut für Angewandte Informatik und formale Beschreibungsverfahren
[2] `http://www.ontoprise.de`

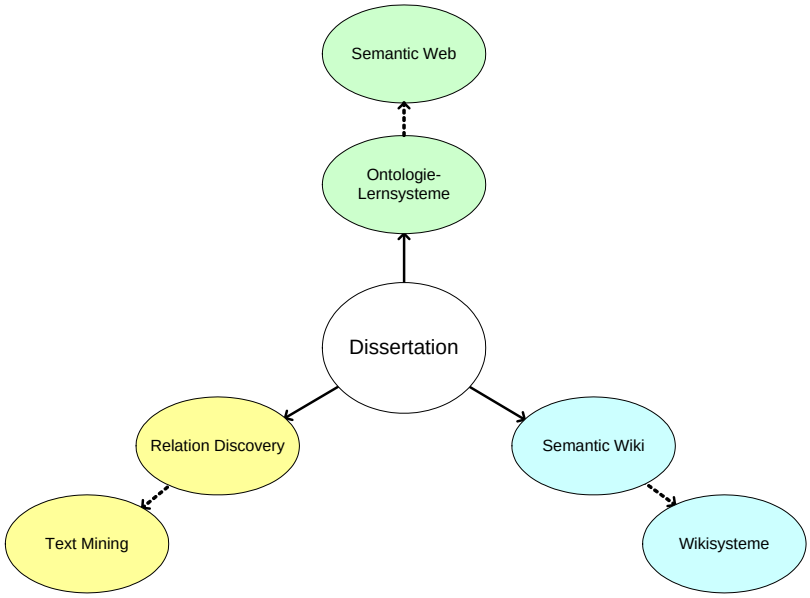


Abbildung 4.1: Einordnung der Arbeit

AIFB hervorgegangen ist. Die Aufgabe von Text-To-Onto ist das Sammeln möglicher Instanzen und Konzepte aus einer festen Dokumentensammlung, um Ontologie-Ingenieuren die Arbeit zu erleichtern. Zur Unterstützung des Sammelvorgangs werden im Vorfeld verschiedene Taxonomien in das System eingegeben, die die maschinellen Lernverfahren unterstützen. Damit ist es möglich, gefundene Instanzen direkt anhand dieser Taxonomien zu klassifizieren. Das Hauptziel liegt in der Erstellung einer Ontologie, die die gefundenen Entitäten möglichst adäquat abbildet. Da diese anhand der Taxonomien gruppiert werden können, verzichtet der Ansatz auf die Suche nach zusätzlichen Relationen zwischen den Entitäten. Dieser Schritt bleibt den Ingenieuren vorbehalten, die mit dem System arbeiten. Die Qualität der automatisch erstellten Ontologie ist damit direkt von der Auffindbarkeit derjenigen Entitäten abhängig, die sich innerhalb der Klassen-Subklassen-Relationen der Taxonomien gut beschreiben lassen. Denn nur diese lassen sich klassifizieren und sind somit über Beziehungen aus der Taxonomie miteinander verknüpft. Zu guter Letzt ist hinzuzufügen, dass das System ohne passende Taxonomien gar nicht funktioniert, also zwingend auf deren Existenz für die jeweilige Domäne angewiesen ist. Damit erfüllt Text-To-Onto die Anforderungen 1, 3 und zum Training der Lernverfahren auch 4, nicht aber die übrigen.

4.1.2 Text-2-Onto

Text-2-Onto [32] ist ein System von Philipp Cimiano und Johanna Völker, ebenfalls vom AIFB in Karlsruhe. Ihr System ist eine Weiterentwicklung des bereits vorgestellten Ansatzes Text-to-Onto, mit drei wesentlichen Änderungen. Erstens: Die zu lernenden Ontologien werden intern nicht in einer der bekannten Formalisierungssprachen verwaltet, sondern in

Form eines sogenannten *Probabilistic Ontology Models*, POM. Darin sind die Konzepte, Instanzen und ihre Relationen enthalten, versehen mit einem Wert, der die Konfidenz des Algorithmus betreffend ihrer Richtigkeit angibt. Zweitens ist eine Schnittstelle zu den eigentlichen Benutzern vorgesehen, damit deren Domänenwissen direkt in die Erstellung der Ontologie einfließen kann und nicht (wie in Text-to-Onto) gefiltert durch Ontologie-Ingenieure, die zwar Expertise im Erstellen von Ontologien, aber eben nicht in der Domäne haben. Drittens enthält das System Funktionalitäten, die es in die Lage versetzen, bei Änderungen im Korpus nicht die gesamte Ontologie zu ersetzen, sondern nur die Teile neu zu berechnen, die von dieser Änderung abhängen.

Zentral für den Ansatz ist jedoch das POM. Dieses erlaubt es, intern beliebig viele verschiedene Hypothesen der Struktur der Ontologie gleichzeitig zu halten, ohne Konsistenzkonflikte bzgl. der Constraints einer Ontologiesprache lösen zu müssen – denn das POM enthält nur Token verschiedener Typen (sog. Modelling Primitives) und Wahrscheinlichkeitswerte. Die Auflösung von Konflikten wird komplett dem Nutzer überlassen. Zur Extraktion der Informationen zum Aufbau der Ontologien aus dem Korpus werden GATE[3] und JAPE[4] verwendet. Die Steuerung des Systems erfolgt über eine graphische Benutzeroberfläche. Darin enthalten sind Funktionalitäten zum Einstellen der Korpora, zum Auswählen und Konfigurieren der Algorithmen und schließlich zur manuellen Manipulation des resultierenden POM. Dazu gibt es zwei verschiedene Ansichten, eine tabellarische und eine in der Form eines zweidimensionalen Graphen. Hinzufügen bzw. Löschen von Relationen, Konzepten und Instanzen sind die angebotenen Möglichkeiten zur Korrektur. Um die manuell korrigierte Ontologie in ein Format zu bringen, das in anderen Programmen verwendet werden kann, muss diese aus dem System exportiert werden. Dazu stehen Transformer für die Sprachen RDFS, OWL und F-Logic zur Verfügung.

Text-2-Onto schlägt einen anderen Weg ein als das Vorgängersystem, insbesondere, was die Nutzerfreundlichkeit angeht. Zwar hatte auch schon Text-to-Onto eine Nutzerschnittstelle, allerdings wandte sich diese noch an Ontologie-Ingenieure und nicht an die eigentlichen Nutzer und Domänenexperten. Das ist definitiv ein Schritt in die richtige Richtung, obwohl auch dieses System noch vom Nutzer verlangt, sich mit der Ontologieerstellung näher beschäftigt zu haben, immerhin müssen die Algorithmen manuell ausgewählt und konfiguriert werden. Ferner lernt das System nicht aus den Änderungen, die von den Nutzern vorgenommen werden, die manuelle Korrektur ist der abschließende Schritt im Arbeitsablauf. Daran schließt sich nur noch die Anpassung der Ontologie an Änderungen im Korpus oder aber der Export der Ontologie an. Diese Anpassung an sich verändernde Datenbestände ist ein sehr positiver Aspekt des Systems, spiegelt es doch stärker den typischen Arbeitsalltag potenzieller Nutzer wider. Ein Schwachpunkt des Systems ist allerdings die Beschränkung auf die Unterstützung englisch-sprachiger Dokumente, bedingt durch die Software, die das Text Processing vornimmt. Ferner werden in der Betrachtung des Systems

[3]General Architecture for Text Engineering [34], entwickelt an der University of Sheffield. Siehe www.gate.ac.uk

[4]Just Another Proof Editor, entwickelt von der University of Oxford, siehe www.jape.org.uk

sowohl die notwendigen Vorarbeiten ausgeklammert, die notwendig sind, um die in GATE enthaltenen Eigennamenerkenner zu trainieren, als auch die Beantwortung der Frage, ob es möglich ist, überhaupt eigene Entitätsklassen trainieren zu lassen. Das berührt die Frage, wie modular Text-2-Onto aufgebaut ist. Es lässt sich also sagen, dass das System die Anforderungen 2, 3, 5 in Sachen Netzerstellung , sowie aus dem Nutzungsbereich die Anforderungen 9 und 10 erfüllt, nicht jedoch die übrigen.

4.1.3 OntoMiner

OntoMiner [37, 38] von Hasan Davulcu, Srinivas Vadrevu und anderen von der Arizona State University und der State University of New York at Stony Brook ist ein System zur automatischen Ontologieerstellung aus Webseiten. Dazu wird vom Nutzer eine Reihe von Webseiten angegeben, die vom System ausgewertet werden sollen. Rahmenbedingungen hierfür sind, dass die Webseiten thematisch verwandt sein müssen und ihre Daten in der Form einer Taxonomie organisiert sind. Diese Taxonomien müssen nicht bekannt sein, wichtig ist nur, dass die Daten in einer Baumstruktur kategorisiert sind. Nach Aussagen der Autoren trifft dies auf den größten Teil aller Seiten aus den Bereichen Wissenschaft, Nachrichten, Finanzen und für Online-Shops zu. Die Auswertung der Seiten erfolgt auf Basis der HTML-Repräsentation, deren strukturierenden Komponenten zur Partitionierung der unterschiedlichen Teile der Seite verwendet werden. In jeder Partition wird anschließend eine Hierarchie der enthaltenen Knoten aufgestellt, die die inhärent vorhandene Taxonomie wieder herstellt. Darauf aufbauend werden Instanzen der in der Taxonomie enthaltenen Konzepte identifiziert. Versuchsergebnisse sind angegeben, für ein Beispiel mit Zeitungswebseiten. Hier wurden Precision- und Recall-Werte zwischen 70% und 90% für die Aufstellung der Taxonomie erreicht.

Zur weiteren Ausgestaltung der Taxonomie wird ein Prozess namens Taxonomy Mining vorgeschlagen, in dem die betreffenden Webseiten nach Vorkommen der im Vorfeld erkannten Konzepte durchsucht werden. Häufig vorkommende Bestandteile der Taxonomie sind in diesem Kontext wichtige Konzepte. Abschließend werden die Konzepte der Webseiten in eine Klassen-Subklassen-Hierarchie einsortiert, basierend auf der logischen Struktur der HTML-Repräsentation. Der gesamte Prozess lässt sich für alle Links einer Website fortsetzen, wobei eine immer feiner granulierte Taxonomie entsteht.

Das System erzeugt Taxonomien, also Ontologien, die strikt hierarchisch aufgebaut sind und im Wesentlichen eine Baumstruktur aufweisen, sowie üblicherweise keine Verbindungen auf der gleichen Ebene des Baums enthalten. Das ist in sich schon ein Unterschied zu den übrigen Systemen und den Zielen dieser Dissertation. Darüber hinaus ist der Ansatz auf die Auswertung von HTML beschränkt, so dass er nur für den speziellen Fall der Auswertung von Webseiten eingesetzt werden kann. Details zur vorgeschlagenen *Nutzung* dieser Ontologien fehlen ebenfalls, so dass nicht ganz nachzuvollziehen ist, wozu das System eigentlich eingesetzt werden soll, zumal die Ausgabe der Algorithmen reines XML ist, also

vor einer Verwendung durch Dritte erst in RDF oder OWL übertragen werden müsste. Festzuhalten bleibt, dass das System lediglich die Anforderung 3 erfüllt.

4.1.4 OntoLearn

OntoLearn [79, 80, 81] ist ein Ontologielernsystem von Roberto Navigli und Paola Velardi von der Università di Roma. Es dient zum Lernen von Ontologien vorher festgelegter Domänen. Dazu wird zu Beginn aus einer Dokumentensammlung ein Grundschatz von relevanten Begriffen mittels statistischer und computerlinguistischer Methoden (unter Verwendung des Systems ARIOSTO [9]) extrahiert. Zu den extrahierten Begriffstypen gehören zusammengesetzte Nomen (z.B. *credit card*), Adjektiv-Nomen-Konstruktionen (*public transport*) und Nominalphrasen (*board of Directors*). Um aus der Menge der Begriffe diejenigen heraus zu filtern, die nicht für die Domäne relevant sind (z.B. *last week*), werden zwei Maße verwendet: Das ist zum Einen der *domain relevance score*, zum Anderen der *domain consensus*. Zur Bestimmung des domain relevance score wird die Häufigkeit eines Terms innerhalb einer Domäne in Beziehung gesetzt mit der Häufigkeit in den bekannten Domänen, er kann also nur dann sinnvoll berechnet werden, wenn es mehrere bekannte Domänenkorpora gibt. Der domain consensus wird mittels der Informationsentropie berechnet, d.h. als Summe der negativen Logarithmen der Wahrscheinlichkeit, dass Term t in einem Dokument aus dem Korpus erscheint. Nur wenn die Summe der beiden Maße über einem bestimmten Schwellwert liegt, wird der Begriff übernommen. Die so gefundenen Begriffe werden anschließend mittels WordNet weiter verarbeitet. Hierbei ist zuerst wichtig, die Begriffe in ihren richtigen Bedeutungen zu erfassen. Diese semantische Disambiguierung wird über die Worteinträge in WordNet realisiert. Dazu wird aus den zu einem Begriff gehörenden SynSets ein semantisches Netz gebildet, bei Komposita als Schnitt der Bedeutungen der Einzelbegriffe. Zur Erstellung dieser Netze wird intensiv von der textuellen Beschreibung der einzelnen Synsets und zusätzlich vom SemCor, einem großen, semantisch mit Synsets aus WordNet getaggten Textkorpus, Gebrauch gemacht. Die solcherart verknüpften Netze werden anschließend so gestutzt, dass nur die Konzepte übrig bleiben, die tatsächlich als Begriffe verwendet werden, bzw. diese über eine Höchstzahl von Relationen miteinander verbinden. Das so entstandene Netz kann dann in anderen Programmen weiter bearbeitet werden. Die Autoren schlagen dazu zwei eigene Programme vor, die eine Diskussion der Ontologie zwischen Domänenexperten und die weitere Bearbeitung des Netzes durch Ontologie-Ingenieuren ermöglichen.

Die Autoren berichten von einem interessanten Aspekt der Verwendung von WordNet: Das Ergebnis der Bearbeitung kann sinnerhaltend über die Synset-Label in andere Sprachen übersetzt werden, für die es eine WordNet-Repräsentation gibt. Das ist gerade für größere, international zu erstellende Ontologien von Vorteil. Im Licht der Anforderungen ist festzuhalten, dass das System die Anforderungen 2, 3, über die angeschlossenen Programme auch 6, nicht aber die übrigen erfüllt. Die Art der Ontologieerzeugung ist eher ein Bootstrapping-Prozess, d.h. es wird ein Startnetz erzeugt, nicht aber kontinuierlich

maschinell überwacht und angepasst. Das ähnelt der Idee von OntoMiner und steht im Gegensatz zu Text-To- bzw. Text-2-Onto und den Zielen dieser Dissertation.

4.1.5 Bewertung

Die vorgestellten Ansätze dienen alle dem gleichen Zweck, dem maschinell unterstützten Lernen von Ontologien. Ist das Vorgehen noch ähnlich, so unterscheiden sich die Systeme doch deutlich in den Resultaten. OntoMiner erzeugt eine Taxonomie aus thematisch ähnlichen Webseiten, die streng hierarchisch ist, recht flach ist und wenig bis gar keine Verbindungen auf einer Ebene innerhalb des Baumes aufweist. Zudem ist der Ansatz auf eine einzige Art semi-strukturierter Dokumente beschränkt und das Ergebnis liegt in einer letztlich proprietären Form vor, ist also ohne Zusatzaufwand nicht extern verwertbar. OntoLearn geht da einen deutlichen Schritt weiter. In der Wahl der Eingangsdaten ist das System nicht beschränkt, verlässt sich allerdings sehr stark auf strukturgebende, manuell erzeugte Quellen, eben WordNet, bzw. SemCor. Diese haben allerdings den Vorteil, dass es sie mittlerweile für mehrere Sprachen gibt, ihr System also zumindest anpassbar auf andere Sprachen wäre, auch wenn es nur für Englisch implementiert worden ist. Dennoch ist das Produkt dieses Systems letztlich nur ein Zwischenergebnis auf dem Weg zu einer Ontologie, allerdings eines, in dem der Versuch unternommen wurde, die semantische Bedeutung gefundener Terme herauszuarbeiten und die Terme sinnvoll zueinander in Beziehung zu setzen.

Text-To-Onto und Text-2-Onto schließlich sind Systeme, die versuchen, eine Ontologie zu erzeugen, rein auf der Basis der vorliegenden Texte mit statistischen Methoden. Das stellt sie in Kontrast zu OntoLearn, das außerdem Verfahren des NLP und eben WordNet berücksichtigt. Damit sind die Relationen aus den beiden Systemen nicht so reichhaltig wie die aus OntoLearn, allerdings kann insbesondere Text-2-Onto auch auf dynamischen Datensammlungen arbeiten, eignet sich daher auch für den kontinuierlichen Einsatz zur Wartung von Ontologien.

Zusammenfassend ist zu sagen, dass die gezeigten Systeme jeweils sehr interessante Aspekte und Verfahren aufweisen, allerdings keins alle Anforderungen erfüllt, die in diesem Kapitel erarbeitet worden sind. Ferner ist zu beobachten, dass es insbesondere auf dem Gebiet der Evaluierung von Lernverfahren für Ontologien noch viel Forschungsbedarf gibt, ehe ein Stand erreicht ist, der einen wirklichen Vergleich verschiedener Lernverfahren und Systeme erlaubt, bzw. ermöglicht. Ansätze dazu finden sich z.B. in [45].

4.2 Ontologie-Editoren

Die Entwicklung von Oberflächen zur Erstellung, Bearbeitung und Wartung von Ontologien ist ein weiterer wichtiger Aspekt der Forschung und Entwicklung auf dem Gebiet des

Semantic Web. Nachstehend werden die Editoren aufgeführt, die in den letzten Jahren innerhalb der wissenschaftlichen Community entwickelt worden sind. Diverse davon sind kostenlos als Open-Source-Software erhältlich und werden mittlerweile auch außerhalb der akademischen Welt eingesetzt.

4.2.1 Protégé

Protégé [82, 84] ist eine Entwicklung der University of Stanford. Der Editor hat zwei grundlegende Modi, in denen er betrieben werden kann: Unter Verwendung von Frame Logic oder unter Verwendung von OWL. Protégé-Frames verwendet intern ein Repräsentationsmodell, das auf dem Open Knowledge Base Connectivity (OKBC) Protocol [30] basiert, während Protégé-OWL das Modell der W3C-Sprache OWL umsetzt. Ontologien können in allen Spielarten von OWL erzeugt werden, also sowohl in OWL Lite, in OWL DL als auch in OWL Full. Damit unterscheidet sich Protégé von der früheren Version Protege2000, die nur Frame Logic beherrschte.

Das Projekt wird als Open-Source-Software entwickelt, über einen Plugin-Mechanismus kann es um zusätzliche Funktionalitäten erweitert werden. Die Website des Projekts[5] enthält eine Liste der momentan rund 90 verfügbaren Plugins[6], darunter auch ein Plugin zum Einlesen von XML Topic Maps.

Die Benutzeroberfläche von Protégé erlaubt nicht nur die Erstellung der Ontologie, sondern ermöglicht über dynamisch erstellte Formulare das Eintragen von Instanzen zu den Konzeptklassen direkt innerhalb des Editors. So kann Protégé auch zur Populierung der Ontologie genutzt werden. Dabei findet eine Plausibilitätskontrolle auf den Formularen statt, d.h. die definierten Einschränkungen auf den Konzeptklassen werden überprüft, auf mögliche Fehler wird graphisch hingewiesen und ein Speichern der Instanz ist nur möglich, wenn die Bedingungen alle erfüllt sind.

Protégé hat eine sehr aktive Entwicklergemeinde, die regelmäßige Treffen durchführt. Die Plugin-Architektur ist eine der Stärken des Systems, da über die Plugins sehr viele zusätzliche Funktionalitäten in Protégé integriert werden können, die andere Editoren nicht anbieten können. So gibt es Plugins für die Anbindung der externen Inferenz-Engines RACER[7][57, 58] und Pellet[8]. Diese können zur Konsistenzüberprüfung oder zur Anfrage der Ontologien eingesetzt werden. Die Anbindung verschiedener Backend-Systeme kann ebenfalls über Plugins realisiert werden. Das macht Protégé zu einer interessanten Grundlage für die Entwicklung eigener Plugins.

Am interessantesten macht Protégé allerdings die Anbindung von GATE über ein Plugin, so dass die Ausgabe von GATE direkt als Input für die Ontologieerstellung in Protégé

[5] Projekt-Homepage: `http://protege.stanford.edu`
[6] Liste der Plugins: `http://protege.cim3.net/cgi-bin/wiki.pl?ProtegePluginsLibraryByType`
[7] http://www.racer-systems.com/products/racerpro/index.phtml
[8] http://www.mindswap.org/2003/pellet/index.shtml

genutzt werden kann.

4.2.2 OntoEdit

OntoEdit [100, 101, 102] ist ein Ontologie-Editor, der am AIFB Karlsruhe im Rahmen eines EU-Projekts entwickelt worden ist. Der Editor erlaubt die Erstellung von Ontologien in einer graphischen Benutzeroberfläche, in der Konzepte, Relationen und Instanzen mit tabellarischen Sichten und Baumsichten modelliert werden. Der Editor ist so angelegt, dass verschiedene Ingenieure zeitgleich an Ontologien arbeiten können. Dazu liegt die zu bearbeitende Ontologie auf einem Server vor, zu dem sich mehrere OntoEdit-Instanzen verbinden. Als Sicherungsmechanismen gegen konkurrierende Updates sind verschiedene Sperrmechanismen integriert, sowohl auf Zweigebene im Ontologiebaum als auch auf Konzeptebene.

Zur weiteren Unterstützung bei der Anforderungssammlung sind zwei Plugins integriert, namens OntoKick und Mind2Onto. OntoKick dient zum Sammeln von Kompetenzfragen. Diese werden in Interviews mit Domänenexperten gesammelt und sollen potenziell interessante Fragen an die Ontologie identifizieren, da sich daraus Relationen, Konzepte und Konzeptattribute ableiten lassen, die in ihr zur Beantwortung enthalten sein müssen. Die so gesammelten Fragen lassen sich im Editor mit den modellierten Konzepten verbinden, damit andere Ingenieure besser nachvollziehen können, warum bestimmte Elemente hinzugefügt worden sind. Mind2Onto ist ein Tool, das Mind Maps aus einem kommerziellen Tool namens MindManager in den Editor importieren kann. Dieser Importweg ist für Domänenexperten interessant, die nicht mit einem Ontologie-Editor umgehen können, aber mit Mind Maps vertraut sind.

Die Zielnutzer für OntoEdit sind Ontologie-Ingenieure, die sich in Sitzungen mit den Domänenexperten ein Bild der Domäne zu machen versuchen und sich dann zur Ontologieerstellung zurückziehen. Die Verteilung der Software in einen Server- und einen Clientteil führt zwar dazu, dass mehrere Nutzer an einer Ontologie arbeiten können und ist interessant, was die Übertragung von Locking-Mechanismen aus der Datenbankwelt auf die Ontologieerstellung angeht, im Kern ist OntoEdit aber weiterhin ein Tool, das zwar Kooperation der Ingenieure miteinander, nicht aber von Domänenexperten und Ingenieuren erlaubt.

OntoEdit ist nicht mehr erhältlich, das AIFB hat die Verbreitung dieser Software mit Beginn der Verfügbarkeit von Kaon (s.u.) eingestellt. Es ist in dieser Auflistung zur Komplettierung der Übersicht trotzdem enthalten, zumal Teile von OntoEdit in Kaon eingeflossen sind.

4.2.3 OilEd

OilEd[9][10] ist eine Entwicklung der University of Manchester, die in der bisher beschriebenen Menge von Editoren eine Sonderstellung einnimmt: OilEd ist ein Ontologie-Editor, der als Grundlage die Sprache OIL (Ontology Inference Layer) verwendet. OIL ist eine europäische Entwicklung, die parallel zu der amerikanischen von DAML (DARPA Agent Markup Language) statt fand. Die beiden Sprachen verschmolzen erst zu DAML+OIL, ehe im Zuge der Beratungen des W3C schließlich OWL daraus entstand. OilEd beherrscht neben OIL auch DAML+OIL. OilEd ist in seinem Funktionsumfang vergleichbar mit Protege2000, das ungefähr um die gleiche Zeit entstand, damals aber nur Unterstützung für das OKBC anbot, das letztlich ebenfalls einen Frame Logic - Ansatz verfolgt. Die wesentliche Neuerung von OilEd bestand allerdings in der Tatsache, dass in dem Editor bereits eine Reasoning-Engine auf Basis von SHIQ enthalten war, mit der Ontologien direkt auf Konsistenz überprüft bzw. angefragt werden konnten. SHIQ ist eine Description Logic - Sprache, in die sich OIL-Ontologien transformieren lassen.

OilEd wird nicht mehr von der University of Manchester weiterentwickelt, da die zu Grunde liegende Sprache OIL nach ihrem Aufgehen in OWL nicht mehr verwendet wird.

4.2.4 Bewertung

Die Liste der über die Jahre entwickelten und in der Literatur besprochenen Ontologie-Editoren ist sehr lang. Von den meisten dieser Editoren ist jedoch nach der ersten Veröffentlichung nicht weiter berichtet worden, insofern ist anzunehmen, dass sie außerhalb des speziellen Anwendungsfalls, für den sie ursprünglich geschrieben wurden, nicht weiter entwickelt worden sind. Impulse für das Gebiet sind von ihnen jedenfalls nicht ausgegangen. Die Liste der Editoren, die sich in der Literatur immer wieder finden, ist ungleich kürzer. Die hier detailliert aufgeführten gehören alle dazu und sind deshalb ausgewählt worden. Von ihnen ist nur noch Protégé in Gebrauch, da die Entwicklung an OilEd eingestellt wurde, als sich abzeichnete, dass weder OIL noch Daml+OIL als Sprache Bestand haben würden. Die gesonderte Entwicklung von OntoEdit hingegen ist vom AIFB eingestellt worden, in Zusammenarbeit mit dem FZI Karlsruhe wird dort nunmehr an KAON2 gearbeitet. Dabei handelt es sich um einen integrierten Werkzeugkasten, der die Entwicklungen des AIFB und des FZI enthält. Darin ist auch eine Editorkomponente auf Basis von OntoEdit enthalten. Ob KAON2 außerhalb der beiden Einrichtungen Verbreitung finden wird, ist abzuwarten, die Community rings um Protégé ist indes sehr groß und sehr aktiv; sowohl in Sachen Nutzung als auch in Sachen Weiterentwicklung von Plattform und Plugins.

[9] Siehe http://oiled.man.ac.uk/

4.3 Verfahren zum Relationslernen

Relationslernen (engl. *Relation learning*) ist eine Disziplin der Informatik, die sowohl in der Computerlinguistik, als auch im Machine Learning angesiedelt werden kann. Beide Teilbereiche suchen nach maschinellen Möglichkeiten, Relationen zwischen verschiedenen Entitäten zu finden und zu benennen. Man kann zwei Aufgaben unterscheiden, die Extraktion und die Entdeckung von Relationen. Im ersten Fall geht es um die Entscheidung, ob ein vorgelegtes Textstück eine Relation aus einem Vorrat vorher bestimmter Relationen enthält, also eine Klassifikation. Im zweiten Fall geht es darum, Relationen im Text zu entdecken. Bei dieser Aufgabe gibt es üblicherweise keine vorher festgelegte Menge von Zielrelationen. In der Computerlinguistik ist in der Vergangenheit oftmals das Interesse an Aufgaben der ersten Gruppe stärker ausgeprägt gewesen als im Machine Learning. Heutzutage ist jedoch eine verstärkte Anwendung statistischer Lernverfahren zu beobachten.

Die bestehenden Ansätze zum Relationslernen lassen sich in verschiedene Richtungen unterteilen:

1. Regelbasierte Algorithmen, die Strukturen natürlicher Sprache ausnutzen,
2. Auswertung des gemeinsamen Auftretens von Entitäten (co-occurrences)
3. Maschinelle Lernverfahren

Die erste Richtung ist klassisch für die bisherige Computerlinguistik; hier wurde seit Jahrzehnten an effizienten Parsern für die syntaktische und semantische Struktur von Texten gearbeitet. Diese Parser liefern eine baumartige Repräsentation des Textes, in denen die Knoten einzelne Wörter sind, die mit Informationen über ihre Funktion und Typisierungen im Satz angereichert sind. Diesen Algorithmen ist gemein, dass sie eine intensive Vorverarbeitung der Texte erfordern und ein möglichst großes Lexikon der verwendeten Sprache benötigen, um die jeweiligen Token richtig zuordnen zu können. Ein typischer Ansatz in dieser Gruppe ist, die Verben der Sätze zu analysieren und zwischen den Subjekt und Objekten die entsprechenden Beziehungen zu etablieren. Meist ist dieser Ansatz eingeschränkt auf eine kleine Anzahl von Verben, um den Verarbeitungsaufwand klein zu halten. Daher sind diese Algorithmen für die Verwendung im geplanten System nicht geeignet, denn sowohl die im Vorhinein über Lexika und Verbauswahlen zu modellierende Vorkenntnis über die Domäne als auch die Einschränkung auf bestimmte Verbmengen übersteigen den angestrebten Aufwand für die Anwendung auf Fachkorpora deutlich. Aus diesem Grund sind sie in der Betrachtung der verwandten Arbeiten nicht weiter berücksichtigt worden.

Die Ansätze der zweiten Richtung wenden statistische Verfahren an, um Relationen zwischen Entitäten zu finden. Dabei wird besonderen Wert auf sogenannte Co-Occurrences von Entitäten gelegt. Kommen Gruppen von Entitäten statistisch signifikant oft zusammen vor, so wird eine Verbindung zwischen diesen Entitäten postuliert. Ob eine weitere Spezifizierung dieser Verbindung statt findet, ist anwendungsabhängig. Falls dies geschieht,

wird aus einem vorher definierten Vorrat an Relationsarten die passendste ausgewählt. Ein Nachteil dieser Methode liegt darin, dass hierbei solche potenziellen Verknüpfungen, die absolut gesehen selten vorkommen, keine Berücksichtigung finden, auch wenn zwischen ihnen eine signifikante Abhängigkeit bestehen könnte und sie somit für den Informationsgewinn vielleicht am interessantesten wären. Darüber hinaus ist festzuhalten, dass die meisten Systeme lediglich *binäre* Relationen lernen. Inwieweit das der realen Welt gerecht wird, ist zumindest diskussionswürdig.

In der dritten Gruppe schließlich finden sich Anwendungen und Erweiterungen klassischer Machine Learning Verfahren. Hierbei finden die verschiedensten Verfahren Einsatz: Hidden Markov Modelle, Support Vector Machines und Kernel Methods werden oft eingesetzt, naive Base Classifier finden Verwendung als Baseline für die Evaluation. Die Zusammenstellung der Feature-Vektoren für den Lernvorgang wird oft mit der Unterstützung von (im Vergleich zu den Parsern der ersten Gruppe) einfachen Textparsern realisiert, indem einfache syntaktische Zusatzinformationen erzeugt oder direkt die Ergebnisse einer Eigennamenerkennung verwendet werden. Dabei besteht allerdings die Gefahr, dass die Auswahl der Trainingsbeispiele zu sehr davon abhängig ist, welche Entitäten in den Beispielen vorkommen, so dass das fertige System lediglich diese Entitäten wiedererkennt und nicht die Relationen berücksichtigt, an denen man interessiert ist. Auch hier wird oft nur mit binären Relationen gearbeitet.

4.3.1 Co-Occurrence Methoden

Hasegawa, Sekine und Grishman berichten in *Discovering Relations among Named Entities from Large Corpora* [103] von ihrem Ansatz zum Finden von Relationen in Texten. Sie verwenden einen vorhandenen Eigennamenerkenner, um Entitäten verschiedener Kategorien zu markieren (verwendet wurden Personen, GPE[10] und Organisationen). Anschließend wird der Text satzweise bearbeitet. Stehen zwei Entitäten in einem Satz und haben sie zusätzlich nicht mehr als fünf Worte Abstand, werden die Entitäten mit den dazwischen liegenden Wörtern weiter verarbeitet. Für jede Klasse von Paaren (also z.B. PERSON–GPE) wird nach der Textdurchsicht eine Clusteranalyse durchgeführt. Dies geschieht auf der Basis der Kontextinformationen, die durch den Text zwischen den Entitäten der einzelnen Paare gegeben ist. Zur Vorbereitung der Feature-Vektoren werden Stoppworte entfernt, die übrigen anschließend mit einem Parts-Of-Speech-Tagger (abgekürzt: POS-Tagger) bearbeitet und anschließend lemmatisiert. Die Feature-Vektoren enthalten dann den tf*idf-Wert der Wörter. Die Ähnlichkeit der Vektoren wird mittels Kosinus-Ähnlichkeit bestimmt, Zugehörigkeit zu einem Cluster bestimmt sich nach einem Schwellwert für den Unterschied der Vektoren. Als Bezeichner der einzelnen Cluster werden die häufigsten Wörter innerhalb der Vektoren der Klasse verwendet.

[10] steht für Geo-Political Entities, hier definiert als eine geographische Entität, die eindeutig verortbar ist und Regierung aufweist. Darunter fallen z.B. Länder, Bundesländer, Kreise, Städte und Orte

Zur Evaluation des Ansatzes wurde ein Jahr der New York Times zuerst manuell bearbeitet, um eine Bewertungsgrundlage zu erhalten. Dabei beschränkten sich die Autoren auf die Domänen PERSON–GPE und COMPANY–COMPANY. Für Erstere fanden sie 177 verschiedene Paare, die sie manuell in 38 Cluster unterteilten. Letztere enthielt 65 verschiedene Paare, die in 10 Cluster gruppiert wurden. Anschließend wurden diese Domänen dann automatisch bearbeitet. Die Qualität der automatischen Ergebnisse ist direkt abhängig vom Schwellwert für die Kosinus-Ähnlichkeit. In der besten Einstellung fand der Algorithmus 34 Cluster für PERSON–GPE und 15 für COMPANY–COMPANY. Recall und Precision lagen dabei bei 79% und 83% für die erste, 76% und 74% für die zweite Domäne[11]. Die Qualität der erreichten Ergebnisse wird in der Veröffentlichung nicht mit anderen Ansätzen verglichen, eine Einordnung findet nicht statt. Als Grund für die schlechtere Leistung im zweiten Fall wird die geringere Anzahl an Fundstellen und die Häufung asymmetrischer Relationen in diesem Gebiet angeführt. Als Erweiterungsmöglichkeiten für die Zukunft werden die Einbeziehung von mehr Kontextinformationen, speziell direkt vor dem ersten und hinter der zweiten Entität genannt. Als Schwachstelle des Algorithmus wird angesehen, dass er lediglich solche Relationen findet, die auch häufig im Text vorkommen. Ein größerer Informationsgewinn würde erreicht, könnten auch selten auftretende Relationen entdeckt werden. Insofern ist eine Verbindung des Algorithmus mit einem anderen System angedacht, aber leider nicht verfolgt worden.

In *Preemptive Information Extraction using Unrestricted Relation Discovery* [114] berichten Sekine und Shinyama über ein System zum Finden beliebiger Relationen aus Nachrichtenartikeln. Dabei verwenden sie den Cluster-Ansatz aus dem weiter oben beschriebenen Artikel in abgewandelter Form. Die Aufgabe des Algorithmus ist es, aus einer Menge von Artikeln Daten zu extrahieren, die es erlauben, Tabellen zu erstellen, die Vorkommen der gleichen Relationen enthalten. Als Beispiel wird eine Tabelle angeführt, die in der ersten Spalte die Namen von Stürmen enthält und in der zweiten Spalte die Orte, an denen sie Schaden angerichtet haben. Diese Aufgabe geht über die in [103] berichtete dahingehend hinaus, dass die Cluster nicht durch die Typen der Entitäten vorbestimmt sind, gleichwohl werden wieder nur Paare von Entitäten betrachtet.

Die Gewinnung der Feature-Vektoren für die Cluster-Erstellung wird nunmehr auf der Basis sogenannter *basis patterns* durchgeführt. Das basis pattern einer gefundenen Entität ist der Teil des Textes, der mit dieser Entität syntaktisch verbunden ist. Daraus lassen sich dann Rückschlüsse auf die Rolle der Entität im Text ziehen. Deswegen ist allerdings die Performanz auch von der Formulierung abhängig. Zu Beginn des Algorithmus werden aus einer Artikelmenge Cluster gebildet, die Artikel zum gleichen Thema enthalten. Dies erfolgt über herkömmliche Ähnlichkeitsmaße anhand der Wortvektoren der Artikel. In diesen, in der Arbeit *basic clusters* genannten Artikelgruppierungen werden dann mit einem auf Hidden Markov Modellen basierenden Eigennamenerkenner Entitäten der Typen Person, Organisation, GPE, Ort und Einrichtung erst erkannt und dann im nächsten Schritt die basic patterns der Entitäten in den einzelnen Basisclustern bestimmt. Aus diesen Ba-

[11] F-Measures damit 80 bzw. 75

sisclustern werden dann in einem letzten Schritt Metacluster gebildet. Diese Metacluster sind die weiter oben erwähnten Tabellen, enthalten also alle Vorkommen einer Relation in allen Artikeln.

Die Datenbasis der Evaluierung des Ansatzes erfolgte mit einer Artikelmenge, die aus zwölf amerikanischen Zeitungen über einen Zeitraum von zwei Monaten per Web Crawling der Webseiten der Zeitungen generiert wurde. Aus den derart gesammelten 28.000 Artikel wurden 5500 Basiscluster gebildet, mit ca. 8000 basic patterns. Die Metaclustering-Phase ergab 302 Tabellen, von denen alle verworfen wurden, die weniger als 3 Zeilen aufwiesen. Letztlich blieben rund 100 Tabellen übrig. Von diesen wurde eine zufällig ausgewählte Menge (48 Stück) manuell überprüft. 75% der Tabellen wurden dabei als konsistent betrachtet, d.h. mindestens die Hälfte der Zeilen der Tabelle beschrieben das gleiche Relationsmuster. Insgesamt beschrieben 73% der Zeilen das jeweils von ihnen erwartete Muster. Diese Zahlen sind nicht wirklich aussagefähig, eine normierte Bewertung der Ergebnisse bleiben die Autoren schuldig.

4.3.2 Machine Learning

Zelenko, Aone und Richardella beschreiben in ihrer Arbeit *Kernel Methods for Relation Extraction* [115]aus dem Jahr 2003 einen Ansatz zum Lernen von Relationen, der auf der Verwendung von Kernel-Methoden beruht. Im Gegensatz zu den bisher beschriebenen Ansätzen handelt es sich hier um eine Arbeit zur Extraktion vordefinierter Relationen, also eine Klassifikationsaufgabe. In der Arbeit werden Kernel eingeführt, die auf der Basis von einfachem semantischen Parsing (sog. *shallow parsing*) der Eingangstexte erzeugt und berechnet werden können. Ausgabe des shallow parsing ist ein Ableitungsbaum des Satzes, in dem die einzelnen Worte mit ihren Bedeutungen versehen sind. Für die Kernel werden Beispielsätze erst geparsed und dann die Bestandteile der Bäume mit ihrer Rolle für die zu lernende Relation beschriftet. Mit der Herleitung von Matching- und Ähnlichkeitsfunktionen auf diesen erweiterten Bäumen wird der Kernel definiert.

Kernel für zwei verschiedene Relationen (person-affiliation und organization-location) werden in der Arbeit mit zwei verschiedenen Lernalgorithmen verwendet, nämlich mit Support Vector Machines (SVM) und mit dem Voted Perceptron [113]. Beide Algorithmen werden mit verschiedenen Ausprägungen der Kernel getestet, aber auch mit einem Naive Bayes-Ansatz, dem Winnow-Algorithmus und einer Support Vector Machine verglichen, die herkömmliche Feature-Vektoren verwendet. Das Testmaterial besteht aus 200 Artikeln aus verschiedenen amerikanischen Zeitungen. Im Vergleich zeigt sich, dass die Kernel-basierten Verfahren ähnliche, oft auch bessere Ergebnisse erzielen, als die Vergleichsalgorithmen. Am besten unter allen Verfahren schneidet die SVM mit der spärlich besetzten Variante der Kernel ab, mit einem F-Measure von 86,8% bzw. 83,3%. Im Vergleich der Lernkurven zeigt sich, dass die Kernel-Methoden sowohl steiler ansteigen als auch schneller konvergieren als die anderen Methoden. Dabei verlaufen die Kurven für die SVM oberhalb der Kurven der

Voted Perceptron Varianten.

Culotta und Sorensen stellen in *Dependency Tree Kernels for Relation Extraction* [6] aus dem Jahr 2004 einen ähnlichen Ansatz zu dem von Zelenko vor, verwenden allerdings eine reichhaltigere Repräsentation der Satzstruktur, indem sie nicht auf dem Ableitungsbaum des Parsings arbeiten, sondern auf Abhängigkeitsbäumen, die einen engeren Zusammenhang zwischen den beiden an der Relation beteiligten Entitäten herstellen. Dadurch erhoffen sie sich eine deutlichere Verringerung von Störungen durch Eliminierung von Rauscheffekten. Verwendet wird der kleinste Abhängigkeitsbaum, der beide Entitäten enthält, unter der Annahme, dass dieser Baum nur die gemeinsamen Textabhängigkeiten enthält. Die Arbeit versucht zuerst, mit den Kernen aus der Menge der erzeugten Abhängigkeitsbäume diejenigen heraus zu filtern, in denen tatsächlich eine der gewünschten Relationen besteht und anschließend erst, diese auch zu klassifizieren. Das geht über die Arbeit von Zelenko hinaus, der direkt mit einem sehr kleinen Satz von Zielrelationen startet, diese trainiert und anschließend nur noch klassifiziert.

In ihrer Arbeit vergleichen sie den Dependency Tree Kernel mit einem Bag-of-Words-Kernel (Ähnlichkeit wird definiert über Vorkommen von Worten. Je mehr gemeinsame Wörter zwei Teilsätze haben, desto ähnlicher sind sie). Darüber hinaus führen sie Komposit-Kernel ein, Kombinationen unterschiedlicher Kernel. Die Evaluierung wird anhand des ACE(Automatic Content Extraction)-Korpus des National Institutes for Standards and Technology[12] durchgeführt. Dieser enthält 800 Dokumente verschiedener Quellen mit gekennzeichneten Entitätstypen (fünf verschiedene) und Relationsarten (24 verschiedene). Die Kernel werden in einer SVM gelernt. Auf diesem Korpus erreicht ihr Komposit-Kernel aus Bag-of-Words und Continuous Kernel als bester 70,3% Precision bei 26,3% Recall. In den Versuchen wird deutlich, dass die Dependency-Tree Kernel deutlich besser arbeiten als der Bag-of-Words und das sich diese Kernel im Allgemeinen besser zur Klassifikation denn zur Detektion eignen. Der Grund wird in der Inhomogenität der negativen Beispiele gesehen, die eine klare Organisation dieser Gruppe sehr erschweren. Zur Behebung dieses Problems wird vorgeschlagen, die Vorauswahl der Kandidaten durch ein zur Detektion besser geeignetes Verfahren erledigen zu lassen und Kernel erst zur Klassifikation einzusetzen.

Bunescu und Mooney stellen in *A Shortest Path Dependency Kernel for Relation Extraction* [91] aus dem Jahr 2005 ebenfalls einen Ansatz auf der Basis von Kerneln vor. Dieser baut direkt auf den Ergebnissen von Zelenko und Culotta auf, die bereits vorgestellt worden sind. Dabei verwenden die Autoren ebenfalls Dependency Tree Kernel, benutzen allerdings nicht den kleinsten gemeinsamen Teilbaum der beiden beteiligten Entitäten wie es Culotta und Sorensen tun, sondern lediglich den Teil dieses Baums, der den kürzesten Pfad zwischen den beiden Entitäten beschreibt. Der Großteil der Arbeit beschreibt die Logik hinter diesem Ansatz. Die Evaluierung wird anhand der gleichen Daten durchgeführt, die auch Culotta und Sorensen verwendet haben, so dass die Ergebnisse verglichen werden können. Gelernt wird auch hier mit einer SVM. Es zeigt sich, dass der vorgestellte Ansatz

[12] US-Einrichtung vergleichbar mit dem DIN

vergleichbare Precision (71,1%) aber mit 39,2% deutlich höheren Recall erreicht. Damit zeigt sich, dass die kürzere Repräsentation sogar noch besser funktioniert, wahrscheinlich aufgrund der weiteren Reduktion von Rauschquellen in den Beispielen.

Culotta, McCallum und Betz beschreiben in *Integrating Probabilistic Extraction Models and Data Mining to Discover Relations and Patterns in Text* [5] aus dem Jahr 2006 einen weiteren Ansatz zur automatischen Extraktion von Relationen aus Texten, dieses Mal allerdings unter Verwendung von Conditional Random Fields (abgekürzt CRF)Im Ansatz wird versucht, neben expliziten Relationen auch implizite Relationen zu entdecken, also solche, die nicht direkt im Text erwähnt werden, aber nichtsdestotrotz bestehen und aus den im Text vorhandenen Informationen auch abgeleitet werden können. Gegeben diese Beispielsätze:

- X ist der Vater von Y.
- X ist der Bruder von W.
- Z ist der Sohn von W.

so sind die Relationen Vater-Kind und Geschwister-Geschwister direkt im Text erwähnt, allerdings existiert zwischen Y und Z auch die Cousin-Relation. Diese Art von Relationen lässt sich zwar theoretisch in einem Modell ablegen, um auch erkannt werden zu können, aber dies für alle möglichen Relationen zu tun, ist (zumindest heute noch) illusorisch. Zudem sind im Data Mining gerade die impliziten Relationen interessant, von denen man vorher nicht einmal weiss, dass sie existieren.

In Ihrem Ansatz betrachten Culotta et al. nur biographische Texte, in denen alle vorkommenden Entitäten in Bezug zu einer Hauptentität stehen, also derjenigen, über deren Leben berichtet wird. Das erlaubt eine Reformulierung des Problems. Denn nunmehr geht es darum, zu jeder darüber hinaus im Text auftretenden Entität den Grad der Beziehung zur Hauptentität zu bestimmen. Das ist deutlich einfacher, als alle möglichen Paare betrachten zu müssen, es wird gleichsam ein Teil des Paares direkt festgehalten. Darüber hinaus werden die Entitäten nicht nach ihrem Typ klassifiziert, sondern nach ihrer Beziehung zur Hauptentität. Das ergibt ein sog. Sequenz-Modell, in dem alle Entitäten in Abhängigkeit von der Hauptentität stehen. Der Ansatz verwendet eine Datenbank, in der Fakten zu den einzelnen Entitäten gesammelt werden. Diese Datenbank stellt Features für die Random Fields zur Verfügung und ermöglicht darüber hinaus, zusätzliche Muster zu lernen, etwa das Cousin-Muster, dass sich aus dem Pfad Sohn-Vater-Bruder-Sohn ableiten lässt. Relationsmuster können mit unterschiedlichen Wahrscheinlichkeiten belegt werden, die wiedergeben, wieviel sie jeweils zur Findung des Ergebnisses der CRF beitragen.

Die Evaluation besteht aus einer Sammlung von 1127 Absätzen aus 271 biographischen Artikeln der Wikipedia. Darin sind 4701 Vorkommen von Relationen aus 53 verschiedenen Typen markiert. Die Evaluierung wurde für verschiedene Lernalgorithmen durchgeführt. Als Baseline dienen ein Maximum Entropy Classifier und ein CRF, das die relationalen

Merkmale (wie das Cousin-Muster) nicht verwendet. Demgegenüber stehen drei Versionen des CRF mit den relationalen Merkmalen, jeweils mit unterschiedlichen Schwellwerten für die benötigte Mindestkonfidenz, das ein Merkmal aufweisen muss, um verwendet zu werden. Der Aufbau der Merkmalsdatenbank wird durch diesen Schwellwert maßgeblich beeinflusst, ist aber selbst schon ein komplizierter Prozess: Dabei werden mehrfache Verarbeitungen des Materials mal mit einem CRF ohne relationale Features durchgeführt, immer im Wechsel mit einem Schritt, der Muster erzeugt und diese bewertet. Diese Bewertungen ergeben letztlich den Wert, gegen den der Schwellwert gemessen wird.

Die Ergebnisse der Evaluierung zeigen, dass das CRF ohne Verwendung der Datenbank besser funktioniert als der Maximum Entropy Classifier (F-Measure von 59,9% gegen 54,8%), aber ungefähr gleichauf mit der Version liegt, in der die Datenbank Verwendung findet (F-Measure von 61,3%, bei einem Schwellwert von 0.5). Diese geringe Verbesserung wird auf Fehler in der Datenbank zurückgeführt. Das untermauert der Versuch, der mit einer manuell korrigierten Datenbank durchgeführt wurde. Dabei erreichte das beste Verfahren dann 67,9% F-Measure, interessanterweise ist das aber eine Version der CRF, das keinen Schwellwert benutzt, sondern alle Muster akzeptiert. Die Messung der Performanz bzgl. impliziter Relationen ist naturgemäß schwierig, das gewählte Verfahren besteht im Durchführen von Tests mit synthetischen Testsätzen und der Überprüfung, ob es zum Erreichen der erwarteten Ergebnisse reicht, entsprechende weitere Fakten in die Datenbank einzupflegen, die ein "Erraten" der gesuchten Relation erlaubt. Dazu wurden von den Autoren zwei Relationen ausgewählt. sie kommen zum Ergebnis, dass ihr Ansatz funktioniert, solange sichergestellt ist, dass der Text kontextuelle Hinweise enthält und die Datenbank Fakten aufweist, die diese Hinweise nutzbar machen. Es wird allerdings recht deutlich, dass das Aufspüren impliziter Relationen sehr viel manuelle Vorarbeit, sehr gutes Material und eine gezielte Interpretation der Ausgaben erfordert.

4.3.3 Bewertung

Die vorgestellten Arbeiten geben einen Querschnitt aktuellerer Arbeiten zum Relationslernen aus Texten. Dabei unterscheiden sich die einzelnen Ansätze zum Teil zwar deutlich, sind aber in der Qualität der Ausgaben durchaus vergleichbar. Mehrere Schlüsse lassen sich aus den Ansätzen ziehen: So sind Clustering-Methoden grundsätzlich angebracht, wenn man nicht im Vorhinein die Relationen festlegen kann oder möchte, da sie allein auf der Basis der vorgelegten Texte eine Unterscheidung vornehmen. Die Arbeiten von Sekine und Hasegawa zeigen, dass durchaus ein Zusammenhang zwischen den ausgedrückten Relationen und den syntaktischen Ähnlichkeiten besteht; dieser Ansatz ist also tragfähig. Worin sich diese Ansätze jedoch schwer tun, ist die Benennung der Cluster, so dass hier oft manuell nachgearbeitet werden muss. Die andere Gruppe wird dagegen häufig dafür eingesetzt, im Vorhinein bekannte Relationsarten zu lernen und neue Texte auf der Basis des Gelernten zu klassifizieren.

Alle vorgestellten Arbeiten benötigen dafür allerdings zumindest eine hinreichende Anzahl positiver Beispiele. Das bedeutet einen nicht unbeträchtlichen manuellen Aufwand für jede Relation. Das, gekoppelt mit dem Problem, dass es im Vorfeld sehr schwierig ist, alle Relationen zu klassifizieren, die interessant sind, lässt diese Algorithmen für das vorliegende Problem als nicht sehr gut geeignet erscheinen, obgleich sie für hinreichend einschränkbare Problemdomänen sicher interessant sind. Für den generellen Ansatz dieser Arbeit erscheint eine Verfolgung der Cluster-Algorithmen hingegen erfolgversprechender, zumal sie unbeaufsichtigt ablaufen können und im Allgemeinen als Vorverarbeitungsschritt lediglich eine Eigennamenerkennung erfordern, deren Klassen sich deutlich einfacher spezifizieren lassen als die Relationen zwischen ihnen. Gegen die Arbeiten dieser Gruppe spricht allerdings, dass sie ausnahmslos binäre Relationen betrachten. Das greift im praktischen Einsatz, gerade in semantischen Netzen eindeutig zu kurz. Ein Algorithmus, der auch n-äre Relationen erkennen kann, wäre hier wünschenswert. Lobenswert ist der Versuch von Culotta, implizite Relationen automatisch aufzuspüren. Sein Ansatz kann allerdings nur als Experiment gewertet werden, das die grundsätzliche Möglichkeit zeigt. Das Maß an notwendigen Vorarbeiten ist so groß, dass das System für einen praktischen Einsatz nicht taugt. Was die Geschwindigkeit der Algorithmen angeht, werden leider grundsätzlich von den Autoren keine Angabe gemacht. Das wäre allerdings eine sinnvolle Zusatzinformation, um die Einsatzmöglichkeiten für die Arbeiten besser einschätzen zu können.

4.4 Semantische Wiki-Systeme

Semantische Wikis erweitern Wiki-Systeme (siehe Kapitel 3.3.4 für eine Einführung) um Techniken des Semantic Web. Primär geht es dabei um die Verbesserung auf zwei Gebieten: Die Datenrepräsentation und die Verknüpfung der einzelnen Artikel innerhalb der Wikis. Herkömmliche Wikis halten den Text der in ihnen enthaltenen Artikel in einer Datenbank und verwenden diese zum dynamischen Erstellen der Seiten.

Um aus Semantic Web - Anwendungen auf Inhalte aus Wikis zugreifen zu können, werden Repräsentationen benötigt, die sich maschinell sinnvoll auswerten lassen. Das zweite Verbesserungsfeld betrifft die Verknüpfung der einzelnen Artikel untereinander. Herkömmliche Wikis verwenden dafür Hyperlinks, die keine weiteren semantischen Qualitäten besitzen. Dadurch können thematische Bezüge zwischen verschiedenen Seiten nur ungenügend ausgedrückt werden, insbesondere lassen sie sich nicht automatisch auswerten. Hier setzen verschiedene semantische Wiki-Systeme an, indem sie die Markup-Sprache des Wikis um Funktionalitäten erweitern, die das Ausdrücken unterschiedlicher Relationen erlauben.

Nachfolgend wird ein Überblick über aktuelle semantische Wiki-Systeme gegeben. Dabei ist zu beachten, dass es sich bei diesen Systemen um Forschungssysteme handelt, die üblicherweise noch in Veränderung begriffen sind.

4.4.1 Semantic MediaWiki

Semantic MediaWiki [66, 108] ist eine Erweiterung zu MediaWiki, dem Wiki-System, das auch von der Wikipedia verwendet wird. Das Ziel der Erweiterung ist die Integration von typisierten Verknüpfungen in bestehende MediaWiki-Installationen. Eine Forschergruppe des AIFB um Max Völkel und Markus Krötzsch in Karlsruhe operiert dabei mit einer recht einfachen Idee: Die Erweiterung der bestehenden Syntax zur Verlinkung von Artikeln um einen optionalen Part, der den Typ der Verknüpfung angibt. Links zu anderen Wiki-Artikeln werden in MediaWiki mittels [[Seitenname]] dargestellt. Will man den Verknüpfungstyp spezifizieren, schreibt man in Semantic MediaWiki [[relationstyp::Seitenname]]. Zusätzlich ist die Typisierung von Informationen innerhalb der Artikel möglich. Dies geschieht mit einer ähnlichen Syntax: [[Attribut:=Wert]], zum Beispiel [[height:=8848 m]]. Diese Angaben können automatisch ausgewertet und bei Anzeige und Suchanfragen gesondert berücksichtigt werden. Semantic MediaWiki ist sogar in der Lage, zwischen verschiedenen Maßeinheiten umzurechnen, z.B. Meter in Fuß.

Die zusätzlichen Daten werden in einem RDF-Speicher verwaltet, der neben der Artikeldatenbank im Wiki-System eingerichtet wird. Die Identifikation der Ressourcen erfolgt über die URI der Wiki-Artikel, die von den Nutzern vergebenen Typisierungen und Attribute dienen als Relationsbezeichner und die Objekte der RDF-Tripel sind andere URI bzw. die Literale, die in einer Attributszuweisung verwendet worden sind. Damit ist jeder Wiki-Artikel in einem solchen System auch als Sammlung von RDF-Statements zugreifbar. Für die externe Nutzung ist das ein nicht zu unterschätzender Vorteil, da sich so jedes Wiki als Informationsquelle für beliebige Semantic Web Anwendungen nutzen ließe. Darüber hinaus erlaubt es aber auch die Etablierung interessanter Zusatzfunktionalitäten innerhalb des Wiki-Systems selbst. Die Suchmöglichkeiten innerhalb der Wikis kann deutlich verbessert werden, da neben der Volltextsuche auch eine semantische Suche möglich wird. Semantic MediaWiki baut dabei auf SPARQL [89], die vom W3C entwickelte RDF-Anfragesprache.

Die Erweiterung von MediaWiki ist eine gute Idee, angesichts der großen Zahl bestehender Installationen in der Folge des Erfolgs der Wikipedia. Allerdings nur, sofern es tatsächlich möglich ist, die Änderungen an der Systemsoftware einem bereits laufenden System hinzuzufügen. Die Änderungen an der Syntax der Markup-Sprache sind hinreichend einfach, dass auch ungeübte Nutzer sie einfach einsetzen können. Es bleibt allerdings zu evaluieren, inwieweit Nutzer bereit sind, die neuen Möglichkeiten zu nutzen und ob sich ein tatsächlich messbarer praktischer Nutzen ergibt. Auf den ersten Punkt gehen die Autoren bereits ein. In [108] heisst es dazu:

> In order for such extensions to be used by editors, there must be new features that provide some form of *instant gratification*.

Diese sofortige Belohnung soll darüber erreicht werden, dass sich die zusätzlichen Eingaben direkt in der Seite niederschlagen, die im Wiki zu sehen ist. Dazu werden typisierte Daten

und Relationen in einer speziellen Box noch einmal gesondert ausgewiesen, so dass sie einen prominenten Platz im Artikel einnehmen.

Ob das allerdings bei der Beantwortung der zweiten Frage hilft, bleibt abzuwarten. Es ist davon auszugehen, dass sich eine Vielzahl von Typisierungen ansammelt, die sich zwar nicht semantisch, aber sehr wohl in der verwendeten Syntax unterscheiden. Dadurch ist mit einem großen Aufkommen semantischer Dubletten zu rechnen. Das Problem ist den Autoren bewusst, sie vertrauen aber auf die Selbstorganisation der Wiki-Communities, die schon dafür sorgen werde, dass aus dem initialen Chaos eine Ontologie entsteht.

4.4.2 Rhizome

Rhizome [98, 99] ist eine Entwicklung von Adam Souzis. Das System ermöglicht die Erstellung von semantischen Wikis, aber ebenso anderer Applikationen, die von der Unterstützung durch semantische Technologien profitieren können. Zu den avisierten Applikationen zählen persönliche Informationsmanager und Forensysteme. Die einzige in der Literatur näher beschriebene Anwendung ist allerdings das Rhizome-Wiki.

Das Ziel bei der Entwicklung von Rhizome ist es, die Erstellung semantischer Informationen und Anwendungen deutlich zu vereinfachen. Souzis sieht dabei in [99] die mangelnde Nutzerfreundlichkeit des Semantic Web als größtes Hindernis auf dem Weg zur generellen Akzeptanz an:

> The second is Semantic Web technology's ease of use. We need to make this technology dramatically easier to use, for both developers and end users. Semantic Web technologies are difficult conceptually, even for experienced software developers. It's difficult to imagine nontechnical end users effectively authoring RDF statements without software assistance. Even for sufficiently trained users, writing the required precise statements takes much more time and effort than writing informal text. Thus [...] the less we need precise, consistent statements, the easier it is to create semantic content.

Dieses Ziel versucht Souzis dadurch zu erreichen, dass sowohl Nutzer als auch Entwickler soweit als möglich von der Notwendigkeit entbunden werden, Aussagen in RDF zu verfassen. Intern arbeitet das System komplett mit selbst entwickelten Beschreibungssprachen, angefangen von der Markup-Sprache für das Wiki (ZML), über ein alternatives Format für RDF (RxML), das angeblich die Erstellung von Aussagen auch für Neulinge erlauben soll, bis hin zu eigenen Versionen von XPath (RxPath) und XSLT (RxSLT). Das für ein Wiki-System eine eigene Markup-Sprache für das Erstellen der Artikel entwickelt wird, ist nicht weiter ungewöhnlich, dass ein Wiki-System jedoch sämtliche benötigten Sprachen selbst definiert, ist mehr als ungewöhnlich. Folgerichtig benötigt Rhizome auch einen Application Server, der diese Sprachen ebenfalls zu verarbeiten in der Lage ist. Den, mit Namen

Raccoon, hat Souzis ebenfalls selbst geschrieben. Dessen Aufgabe ist es, die Datenbasis zu verwalten (die aus einer Menge von RDF-Aussagen besteht) und auf Anfragen mit dem passenden Subset von Aussagen zu antworten, bzw. die geforderten Veränderungen an der Basis vorzunehmen.

Nutzer sollen in einem Rhizome-System im Wesentlichen mit der Markup-Sprache ZML arbeiten. Diese geht nicht über die Fähigkeiten anderer Markup-Sprachen herkömmlicher Wikis heraus, erlaubt somit auch keine freie Definition von Relationsarten bei der Verlinkung mit anderen Seiten. ZML lässt sich mit Hilfe geeigneter Skripte nach XML wandeln, XML-Dokumente wiederum nach ZML. So könnten theoretisch beliebige XML- bzw. HTML-Dokumente in eine Form gebracht werden, die einfaches Editieren durch Nutzer erlaubt, ehe sie wieder zurückgewandelt werden. RDF kann man nicht direkt aus ZML erzeugen, dazu ist ein Schritt nötig, den der Autor *shredding* nennt. Darunter versteht er die Extraktion von RDF-Daten aus beliebigen Dateiformaten. ZML wird dazu zuerst nach XML gewandelt, die Metadaten von MP3-Dateien würden direkt aus den ID3-Tags extrahiert, usw.

Mit Rhizome verfolgt Adam Souzis ein sehr ambitioniertes Ziel. In der Tat krankt das Semantic Web daran, dass es sehr schwierig ist, sich in die Verwendung der einzelnen Sprachen einzuarbeiten. Für Laien ist die Formulierung von Metadaten in RDF viel zu schwierig, als dass sich die manuelle Annotierung von Inhalten auf breiter Front durchsetzen könnte. Ob allerdings RxML einfacher zu verstehen ist als RDF, mag zumindest bezweifelt werden. Als Illustration soll dafür das folgende Beispiel dienen, das zur *einfachen* Einführung von RxML in der Dokumentation von Rhizome dienen soll:

```
 prefixes:
    dc: "http://purl.org/dc/elements/1.1/"
    mysite: "http://example.org/mysite/"

mysite:page1.html
  a: {http://purl.org/dc/dcmitype/Text}
  dc:creator: {bnode:alice}
  dc:title: "The first Page"
```

Wer so tief in die Materie eingestiegen ist, sich freiwillig als Nutzer mit dem Dublin Core vertraut zu machen, dem ist genauso gut zuzutrauen, äquivalente Statements zu denen des Beispiels in einer der Serialisierungen von RDF anzulegen.[13] Zudem ist das Format sehr fehleranfällig, z.B. ist die Einrückung der Statements mit Semantik behaftet. Für Programmier-Laien erschließt sich das auch nicht einfacher als die Verwendung der spitzen Klammern in RDF/XML.

Auf der Rhizome-Seite findet sich leider keine Information über aktuelle Arbeiten an

[13]Die N3-Serialisierung von RDF sieht der im Beispiel sehr ähnlich, kann aber ohne Umwege von jedem RDF-Parser verarbeitet werden.

dem System. Selbst wenn daran nicht mehr gearbeitet werden sollte, ist jedoch der Versuch zu würdigen, ein System zu schaffen, das nach Außen zwar RDF anbietet, in der täglichen Arbeit damit aber ein deutlich einfacheres Format verwendet – auch wenn in diesem Fall das Format nicht gut genug für das selbst gesteckte Ziel gewesen zu sein scheint. Schade ist jedoch, dass durch die konsequente Verwendung proprietärer Formate die Chance vertan wurde, ein wirklich offenes System zu entwickeln. So können nicht einmal die gelungenen Teile des Systems in anderen Arbeiten Einzug halten, ohne dass viele eigentlich unnötige Transformatoren zwischengeschaltet werden, die für die Umwandlung der angeblich äquivalenten Formate in die allgemein verwendeten sorgen.

4.4.3 OntoWiki

OntoWiki [74, 75] ist eine Entwicklung einer Forschungsgruppe um Martin Hepp vom DERI[14] an der Universität Innsbruck. Dabei handelt es sich um einen Prototypen, mit dem untersucht werden soll, inwieweit sich Wikis zur gemeinschaftlichen Erstellung einer Ontologie eignen.

Ausgangspunkt der Entwicklung war die Erkenntnis, dass der Prozess zur Ontologienerstellung ohne den Großteil der adressierten Nutzer statt findet. In den Augen der Autoren stellt eine Ontologie aber nicht nur eine Formalbeschreibung einer Weltsicht dar, sondern insbesondere einen Gemeinschaftsvertrag über diese Weltsicht. So ein Gemeinschaftsvertrag ist aber den Änderungen der Weltsicht der Gemeinschaft unterworfen, so dass eine Ontologie, die von einem kleinen Kreis von Experten gewartet wird, irgendwann unweigerlich die Situation der Gemeinschaft drumherum nicht mehr abbildet und darum von dieser nicht mehr akzeptiert wird. Hinzu kommt, dass in Ontologien zwar die Verwendung eines Konzepts spezifiziert, aber üblicherweise keine Definition des Konzepts gegeben wird.

In Wiki-Systemen und den dazu gehörenden Nutzergemeinschaften wird die Lösung für die angeführten Probleme gesehen, da dort eine ähnliche Situation vorliegt. Jeder Nutzer hat die Möglichkeit, sich bei der Neuanlage von Artikeln einzubringen, Fehler zu korrigieren oder Ergänzungen vorzunehmen. Der Gesamtzustand des Wikis spiegelt den aktuellen Konsens der Gemeinschaft über ihre Sicht auf die behandelten Themen wider.

Mit OntoWiki schlagen sie ein System vor, dass sich der Techniken von Wiki-Systemen bedient, um in einer Gemeinschaft eine Ontologie zu erstellen. Die logische Ausdrucksstärke solcher Ontologien schätzen sie selbst als eher gering ein (in [74] ist von *flat ontologies* die Rede), halten sie aber trotzdem für nützlich, weil sie gemeinschaftlich erstellt werden und sich im Revisionssystem des Wikis die Entstehungsgeschichte der einzelnen Konzepte ablesen lässt. Zusätzlich tritt das Problem der mangelnden Definition der Konzepte gar nicht erst auf, da es sich um Wiki-Artikel handelt.

Der von Hepp verfolgte Ansatz ist durchaus interessant. In der Tat ist es sinnvoll,

[14] Digital Enterprise Research Institute

die jeweilige Community stärker an der Erstellung der Ontologien zu beteiligen, als es bisher üblicherweise der Fall ist. Bei dieser Arbeit können Wikis sicherlich nützlich sein. Allerdings wird ein Nachteil dieses Vorgehens bereits in [74] als Herausforderung formuliert: Der Drift von Bedeutungen ist zwar durchaus gewünscht, stellt aber eine Komplikation dar, wenn extern auf ein Konzept verwiesen wird. Sobald sich dessen Bedeutung ändert, ist der Verweis darauf erst einmal potenziell fehlerbehaftet. Hier versuchen die Autoren mit der Versionierung zu argumentieren, das erscheint jedoch kein tragfähiger Weg. Folgerichtig schlägt Hepp in [75] vor, die Wikipedia nicht nur als Definitionsquelle, sondern gleich als Entwicklungswiki zu benutzen, eine Abkehr vom eigenen System. Damit geht aber einher, dass die Zielontologie keinerlei Hierarchien mehr aufweisen können, da sich die dafür benötigten Relationen nicht in der Wikipedia als URI definieren lassen. Insofern kann man den Versuch als gescheitert ansehen, da solche Ontologien nur sehr begrenzt für den täglichen Einsatz nützlich sind.

4.4.4 Platypus Wiki

Platypus Wiki [27, 104] ist eines der ältesten semantischen Wiki-Systeme, über die in der Literatur berichtet wird. Das Projekt von Stefano Campanini, Paolo Castagna und Roberto Tazzoli stammt aus dem Jahr 2003 und ist geradlinig in seiner Verknüpfung von Semantic Web und Wiki-Systemen. Platypus ist zuallererst ein Wiki-System, mit dem Unterschied, dass die Datenhaltung nicht in einer relationalen Datenbank auf dem Server erfolgt, sondern in einem RDF-Speicher. Alle Metadaten des Wikis werden in RDF vorgehalten, die Art der Ausgabe wird über die angeforderte URL gesteuert. Platypus erlaubt die Definition beliebiger Relationen durch die Nutzer, die Verknüpfungen sind explizit bidirektional ausgelegt, um eine Rückkehr von einer Seite zur vorhergehenden einfach zu ermöglichen. Auf Benutzerseite trennt Platypus die Eingabe des Artikeltexts von der Eingabe semantischer Informationen dadurch, dass sie in verschiedenen Eingabefeldern erfolgen. RDF-Aussagen werden in der N3-Notation [14] eingetragen.

Platypus hat, als eines der ersten Systeme dieser Art, einige Eigenheiten, die nachfolgende Systeme zu vermeiden versucht haben. So ist es sehr RDF-zentrisch angelegt: Nutzer, die mit RDF nichts anfangen können, werden wahrscheinlich nicht mit der Art der Eingabe der semantischen Metadaten zurecht kommen. Darüber hinaus verwendet das System eine recht einfache Markupsprache für die Artikel, die wenige Möglichkeiten für die Textgestaltung anbietet. Beides sind Gründe dafür, dass sich das System keiner großen Verbreitung erfreut. Zudem scheint die Weiterentwicklung schon im Jahr 2005 eingeschlafen zu sein. Vom wissenschaftlichen Standpunkt handelt es sich um ein interessantes System, das als erstes zeigte, dass die Verbindung der Technologien Wiki und Semantic Web echten Mehrwert produziert.

4.4.5 Bewertung

Die Verbindung zwischen Semantic Web und Wiki-Systemen wird erst seit einigen Jahren näher untersucht. Dennoch gibt es bereits eine recht große Spannbreite von verschiedenen Systemen und Ansätzen. In dieser Übersicht ist nur ein Teil der in der Literatur erwähnten Veröffentlichungen berücksichtigt worden. Entscheidend für die Aufnahme war, ob es sich bei dem vorgestellten Ansatz nur um *Paperware* oder tatsächlich um eine Arbeit, die über längere Zeit aktiv verfolgt, bzw. weiter entwickelt wurde. Die Auswahl zeigt, dass die Wikipedia im Bereich der Forschung über Wikis einen breiten Raum einnimmt. Semantic MediaWiki und OntoWiki beziehen sich immer wieder auf dieses Projekt. Dahinter steckt wahrscheinlich der Traum, unter Verwendung der Artikel in der Enzyklopädie eine Art Weltontologie erstellen zu können.

Vorausgesetzt, innerhalb der immensen Artikelmengen[15] ließe sich wirklich sinnvoll von allen Nutzern gemeinsam eine semantische Vernetzung der Artikel erreichen, wäre das in der Tat das Ergebnis: Eine immense Ontologie, die alle Themengebiete berührt, für die sich Menschen hinreichend begeistern können, dass sie ihre Zeit darin investieren, Artikel dazu ins Internet zu stellen. Gerade darin liegt aber auch die Crux dieses Ansatzes: Die Breite der behandelten Themen und die große Zahl der beteiligten Autoren lassen es als sehr unwahrscheinlich erscheinen, dass tatsächlich eine Einigung über das Aussehen und die für eine Ontologie notwendigen Beschränkungen erzielt werden kann. Für einzelne Themen lässt sich sicher so eine Einigung erzielen, da hier die Zahl der Beteiligten (und ihr Kenntnisstand zum Thema) eine sinnvolle Diskussion zulassen. Aber über die Gesamtheit aller Themen? Gleichzeitig? Das ist mehr als zweifelhaft. Eher erscheint es sinnvoll, spezialisierte Wikis zu unterhalten, die sich, bei Bedarf und wo angebracht, untereinander referenzieren. In solchen spezialisierten Gemeinschaften ist dann auch das Nachdenken über eine Ansatz ähnlich zu OntoWiki wieder sinnvoll.

Rhizome wiederum steht für einen Ansatz, der die mangelnde Nutzerfreundlichkeit des Semantic Web als Hauptproblem sieht und in dem dementsprechend versucht wird, die Mechanismen desselben vor dem Nutzer und den Entwicklern solcher Anwendungen transparent zu halten. Diese Idee ist prinzipiell gut, allerdings in der speziellen Ausprägung von Rhizome nicht gut gelöst worden. Leider hat Souzis die Komplexität allgemein anerkannter Formate gegen vermeintlich einfachere ausgetauscht, die nur sein System tatsächlich beherrscht. Sein System adressiert aber eine Frage, die bedenkenswert erscheint: Ist die geringe Ausbreitung des Semantic Web außerhalb der akademischen Welt vielleicht auch darauf zurückzuführen, dass selbst viele Entwickler und Web-Designer die Technologie des Semantic Web für zu unhandlich oder unverständlich halten?

Darüber hinaus gibt es einige Systeme, die auch als *personalized semantic Wikis* bezeichnet werden. Hierbei handelt es sich um Wikis, die explizit für die Verwendung durch *einen* Nutzer vorgesehen sind. Dabei handelt es sich allerdings mehr um Werkzeuge für

[15] Allein die englische Fassung bringt es mit Stand Juni 2008 auf fast 2,4 Mio Artikel, die deutsche auf rund 760.000

Spezialisten, die bereits gute Kenntnisse der Materie des Semantic Web haben. Vertreter dieser Art sind zum Beispiel die Systeme WikSAR [8] und SemPerWiki[16] [85, 86].

4.5 Weiteres Umfeld

Neben den geschilderten Feldern gibt es weitere, deren Arbeiten für den in dieser Dissertation behandelten Gegenstand interessant sind. Eines dieser Felder ist *Ontology Evaluation*. Das Arbeitsgebiet dieses Feldes ist die Suche nach Methoden und Maßen für die Gütebewertung von Ontologien. Dabei spielen Fragen der logischen Widerspruchsfreiheit ebenso eine Rolle wie die Abdeckung der Domäne durch die Konzepte oder die Gewichtung der einzelnen Konzepte untereinander. Die bekannteste Evaluierungsmethode ist OntoClean [52]. Dabei werden die Konzepte der zu untersuchenden Ontologie um Eigenschaften erweitert, die bei der Analyse helfen sollen, inkonsistente Modellierungsentscheidungen aufzudecken. Die Methode genießt einen guten Ruf, ist allerdings voll manuell durchzuführen. In [109] wird ein Ansatz namens AEON vorgestellt, der einige der Metadaten automatisch erzeugen kann. Auf der Basis von OntoClean wurde 2006 ein Ansatz namens CleanOnto vorgestellt, der ebenfalls Inkonsistenzen in Ontologien aufspüren und automatisch beheben soll [95]. Dazu ist allerdings eine Upper-Level-Ontologie erforderlich, also eine hierarchische Sicht auf die ganze Welt. In den Beispielen in der Arbeit wird WordNet verwendet, obwohl es sich dabei eher um eine linguistische denn eine ontologische Ressource handelt.

Zu den eher technisch-mathematisch orientierten Maßen und Methodiken gibt es allerdings auch kritische Meinungen. So vertritt Natalya Noy von der University of Stanford die These, dass die meisten Maße die Bedürfnisse der Nutzer völlig außer Acht ließen, d.h. sie anhand der ermittelten Merkmale nicht entscheiden könnten, welche Ontologie für ihre tatsächlichen Belange gut oder weniger gut geeignet sei [83]. Demzufolge sagten empirische Berichte über Ontologien wesentlich mehr über deren Qualität aus, als Maßzahlen, die aus einem Algorithmus resultierten. Um den Nutzern mehr verständlichere Informationen an die Hand zu geben, sei es daher besser, eine Art Zusammenfassung von Ontologien zur Verfügung zu stellen, etwa in Form ihrer wichtigsten Konzepte (auch *Hub Concepts* oder *Ontology Backbone* genannt). Noy vertritt zwar eine provokante These, überlegenswert ist sie jedoch allemal – denn der Erfolg einer Ontologie ist letztlich von den Nutzern bestimmt. Wenn sie zwar rein formal korrekt modelliert ist, aber darüber für die Nutzer zu unverständlich geworden ist, werden diese eine andere Formalisierung ihrer Domäne wählen, bzw. dem Semantic Web gleich komplett den Rücken kehren.

[16] Siehe http://www.semperwiki.org

4.6 Zusammenfassung

Dieses Kapitel hat Gebiete der Forschung betrachtet, die von den Zielen dieser Dissertation berührt werden. Der Schwerpunkt hierbei liegt auf den Gebieten, die sich mit der Extraktion semantischer Zusammenhänge aus Textsammlungen beschäftigen, sei es in der Form integrierter Ontologie-Lernsysteme oder in der Form losgelöster Algorithmen, die sich in einen Workflow einpassen lassen. Hierbei hat sich gezeigt, dass sich die in der wissenschaftlichen Literatur erwähnten Ontologie-Lernsysteme nur eingeschränkt für die Erfüllung der Ziele dieser Arbeit einsetzen lassen. Die Algorithmen zum Relationslernen weisen ebenfalls Einschränkungen auf, die sie zur Erfüllung der Anforderungen aus Kapitel 2.2 ungeeignet werden lassen.

Ergänzend wurde eine Übersicht über aktuell verfügbare und in der Wissenschaft behandelte Editoren zur Ontologieerstellung gegeben. Diese weisen üblicherweise keine automatischen Fähigkeiten zur Unterstützung bei der Modellierung auf, sind jedoch für die manuelle Modellierung kleinerer Bereiche der Ontologie durchaus hilfreich. Insbesondere Protégé erfreut sich einer weiten Verbreitung innerhalb der wissenschaftlichen Gemeinde: In einer Umfrage im Dezember 2006 gaben 68,2% der Befragten an, Protégé als Ontologie-Editor zu verwenden, weit dahinter folgte OntoEdit mit 12,2%, dann OilEd mit 7,3 % [28].

Im Anschluss wurde das relativ junge Gebiet der semantischen Wiki-Systeme behandelt, da die Verbindung der beiden Technologien Semantic Web und Wiki-Systeme auch im weiteren Kontext dieser Arbeit (siehe Kapitel 6) eine wichtige Rolle spielt. Wiki-Systeme bieten sich als Nutzeroberfläche für solche Datenbestände an, bei denen eine Interaktion der Nutzer mit dem Material explizit gewünscht ist. Die Einbindung der semantischen Verknüpfungen direkt in die Syntax der Markupsprache der Wiki-Systeme eröffnet eine Fülle von Möglichkeiten – auch wenn der Beweis aussteht, dass die freie Erstellung semantischer Verknüpfungen die Nutzer nicht überfordert und das Gesamtsystem nicht ins Chaos stürzt.

Den Abschluss bildete eine Umschau in angrenzenden Gebieten. Speziell im Bereich der Ontology Evaluation zeigt sich, dass die Frage nach dem praktischen Nutzen der Ergebnisse für Endanwender nicht unterschätzt werden darf. Die besten Methodiken und Maßwerte werden sinnlos, wenn sie von den Anwendern weder angewendet noch verstanden werden können, sondern immer einen Experten erfordern, der sich in den Erstellungsprozess einbringt. Diese Erkenntnis lässt sich so auch im Bereich der Ontologie-Lernsysteme machen.

Kapitel 5

Semiautomatische Erstellung semantischer Netze

Dieses Kapitel greift die in Abschnitt 2.3 aufgestellten Anforderungen auf und stellt ihnen Lösungsansätze gegenüber. Diese sind analog zur bisherigen Struktur gruppiert, also in die Bereiche Netzerstellung, Infrastruktur und Netznutzung unterteilt, die nachfolgend in den Abschnitten 5.2 bis 5.4 vorgestellt werden. Das Kapitel schließt mit einer Zusammenfassung des Vorgestellten.

5.1 Vorbemerkungen

Zum Erstellen einer Ontologie ist es unerlässlich, sich im Vorhinein Gedanken über die Welt zu machen, die modelliert werden soll. Denn ihre Entitäten und deren Relationen untereinander bilden den Rahmen dessen, worüber man sich später einvernehmlich mit denen unterhalten kann, die ebenfalls die durch die Ontologie definierte Weltsicht teilen.

Für nichttriviale Themengebiete ist die erschöpfende Aufzählung der möglichen Entitäten nicht wünschenswert, da es einerseits zu aufwändig wäre, andererseits aber auch die Welt in einem in der Zeit fixierten Zustand abbildet. Zur Umgehung dieses Problems werden statt dessen Entitätsklassen eingeführt: abstrakte Obermengen von Entitäten, die gewisse Eigenschaften teilen. Allein auf deren Basis wird die Ontologie definiert, die Entitäten selbst kommen darin idealerweise nicht mehr vor. Die Vorteile dieses Vorgehens liegen darin, dass die in der Ontologie erlaubten Relationen zwischen den Entitätsklassen definiert werden – und nicht mehr für jede Entität einzeln. Entitätstypen dienen als Schablonen für konkrete Entitäten, die damit nach Belieben erzeugt und im Rahmen der Ontologie verwendet werden können.

Die Aufgabe des hier vorzustellenden Ansatzes ist es, den Prozess der Erzeugung einer

Ontologie aus einer Dokumentensammlung weitestgehend zu automatisieren. Dies betrifft sowohl die Suche nach Entitäten verschiedener Entitätsklassen, um aus diesen Instanzen für die Ontologie zu gewinnen, als auch die Bestimmung der möglichen Relationen zwischen den Klassen.

In Abschnitt 2.1.1 wurden die Ziele dieser Arbeit festgelegt, gefolgt von einer Reihe von Herausforderungen auf dem Weg dorthin und daraus resultierenden Anforderungen an ein Software-System, das diesen Weg beschreitet. Die Komplexität der einzelnen Anforderungen ist dabei zum Teil deutlich unterschiedlich, sowohl bezogen auf die Implementierungsarbeit, als auch in Sachen der wissenschaftlichen Arbeit. Für den Fokus dieser Arbeit sind sicherlich die Anforderungen A5 und A6 am interessantesten, definieren sie doch den Kernbereich des eigentlichen Knüpfens eines semantischen Netzes aus einer Fülle an Einzelfakten: Die maschinell unterstützte Relationserstellung.

Die eingangs skizzierte Hauptarbeit bei der Ontologieerstellung zerfällt in die beiden Teile Klassen- und Relationsdefinition. Während der erste sich noch manuell auch für umfangreichere Themengebiete erledigen lässt, sieht es bei der Definition der erlaubten Relationen zwischen den verschiedenen Klassen schon anders aus. Bereits zwischen lediglich zwei Klassen kann es mehrere Dutzend unterschiedliche Relationen geben[1] – und es gibt nicht nur binäre Relationen.

Als Grundlage für die Ontologien dienen große Mengen digitaler Dokumente. Die Vielfalt der darin ausgedrückten Relationen lässt sich nicht ohne große Informationsverluste in eine Relationsmenge abbilden, die im Vorhinein manuell festgelegt werden könnte. Das Ziel muss hier sein, von den zwischen Entitätsklassen existierenden Relationen so viele wie möglich automatisch aufzuspüren, um den Experten eine qualifizierte Auswahl der für sie relevanten Relationen für die Ontologie zu ermöglichen.

Neben diesem zentralen Problem sind außerdem die Fragen der dafür notwendigen, bzw. wünschenswerten Infrastruktur sowie der Möglichkeiten der Nutzung des so erzeugten semantischen Netzes von Interesse. Deshalb folgt dieses Kapitel der Strukturierung in Netzerstellung, Infrastruktur und Nutzung. Die in diese Bereiche fallenden Ansätze werden im Folgenden detailliert vorgestellt.

5.2 Netzerstellung

Der Löwenanteil der Anforderungen in Abschnitt 2.1 konzentrierte sich auf die Spezifika der Netzerstellung. Folgerichtig nehmen auch die Lösungsansätze zu diesem Thema den Großteil dieses Kapitels ein. Da sie im Einzelnen in diesem Zusammenhang noch öfter Erwähnung finden werden, führt Tabelle 5.1 die zugehörigen Anforderungen noch einmal auf.

[1] Zum Beispiel in dem in Abschnitt 4.3.1 beschriebenen Algorithmus von Hasegawa 38 Relationen zwischen Person und Ort

Nr	Anforderung
A1	Import und Verarbeitung von Konzepten
A2	Import unterschiedlicher Datenformate
A3	Automatische Extraktion von Konzeptinstanzen
A4	Annotierung von Beispielen
A5	Ermöglichen von Verknüpfungen auf Instanzebene
A6	Semiautomatisches Verfahren zur Netzerstellung

Tabelle 5.1: Die Anforderungen aus dem Bereich Netzerstellung

5.2.1 Formale Beschreibung

Um zu einem tieferen Verständnis der eigentlich zu Grunde liegenden Problematiken zu gelangen, empfiehlt sich zunächst eine formale Betrachtung des Problemraums. Diesem bewährten Ansatz folgend, werden nachfolgend Definitionen der beteiligten Konzepte gegeben und daraus die Aufgabenstellung formal abgeleitet.

Definitionen

Als Erstes ist es wichtig, den Begriff des *semantischen Netzes* zu konkretisieren. Zunächst gilt es daher, zu definieren, was ein allgemeines semantisches Netz ist.

Definition 1 (Allgemeines semantisches Netz)
Ein allgemeines semantisches Netz ist ein Tupel, bestehend aus einer Menge von Knoten K und einer Menge von Relationen R, die verschiedene Knoten miteinander verbinden. Diese Relationen sind ungerichtet, mindestens 2-stellig und in ihrer Stelligkeit nicht nach oben beschränkt.

Damit ist das semantisches Netz grundlegend definiert. Für den Kontext dieser Arbeit sind jedoch noch einige ergänzende Definitionen notwendig. So gibt es zusätzlichen Klärungsbedarf im Bereich der Konzepte und der Relationen, ehe wir eine finale Definition semantischer Netze im Arbeitskontext geben und aufbauend darauf die Aufgabe definieren können. Wir beginnen mit der Definition des Konzepts.

Definition 2 (Konzept)
Ein Konzept definiert das ideale Bild eines Gegenstands, einer Klassifikation, eines Sachverhalts oder einer Idee.

Damit ist geklärt, worum es sich bei einem Konzept handelt. Ein Konzept definiert gleichsam das Ideal, analog zu Platos Höhlengleichnis. Die von diesem Ideal geworfenen Schatten sind die so genannten Konzeptinstanzen.

Definition 3 (Konzeptinstanz)
Eine Konzeptinstanz (auch Entität, Instanz), ist ein konkretes, individuelles Vorkommen eines Konzepts, d.h. die Konzeptinstanz ist anhand ihrer Eigenschaften und Merkmale als zu einem Konzept zugehörig identifizierbar, kann aber hinreichend von anderen Instanzen dieses Konzepts unterschieden werden.

Mit der letzten Definition im Bereich der Knoten des semantischen Netzes wird die Brücke zu dem im Ontologieumfeld eher gebräuchlichen Begriff der *Klasse* geschlagen.

Definition 4 (Konzeptklasse)
Eine Konzeptklasse (auch Entitätsklasse, Klasse) $\mathcal{K}$ ist ein Paar (K,I), bestehend aus einem Konzept K und einer Menge I, die ausschließlich Instanzen von K oder aber Subklassen von $\mathcal{K}$ enthält.

$\mathcal{K}'$ ist genau dann eine Subklasse von $\mathcal{K}$, wenn $K' \lesssim K$ und $I' \subset I$ gelten. In so einem Fall nennt man $\mathcal{K}$ dann auch die Superklasse von $\mathcal{K}'$.

Damit ist die Grundlage für den Aufbau von Subklassenhierarchien gelegt. Das ist wichtig für die Abbildung von Ontologien.

Außerdem ist eine weitere Spezifikation des Relationsbegriffs notwendig, um zu der endgültigen Definition des semantischen Netzes gelangen zu können. Dazu wird zunächst das Relationsmuster eingeführt, eine Struktur, die im weiteren Verlauf dieses Kapitels benötigt wird.

Definition 5 (Relationsmuster)
Ein Relationsmuster ist ein Paar (Λ,K), wobei Λ ein Name und K eine Menge von Konzeptklassen ist. Ein Relationsmuster drückt aus, dass es eine grundsätzliche Verbindung zwischen den in K enthaltenen Konzeptklassen gibt, die sich mit Λ benennen lässt.

Aus dieser Definition, die sich von ihrem Abstrahierungsgrad mit einem Konzept vergleichen lässt, kann direkt der Begriff der Relation sowie der Instanz eines Relationsmusters abgeleitet werden.

Definition 6 (Relation)
Eine Relation ist ein Paar (λ,I), wobei λ der Name der Relation und I die Menge der an der Relation beteiligten Instanzen ist.

Definition 7 (Instanz eines Relationsmusters)
Gibt es für eine Relation $\rho(\lambda,I)$ ein Relationsmuster $\Phi(\Lambda,K)$, so dass $\lambda \approx \Lambda$ und $\forall i \in I : \exists k \in K : i \in k$ gelten, dann ist ρ eine Instanz von Φ.

Schließlich gibt es auch bei Relationen den Begriff der Klassen.

Definition 8 (Relationsklasse)
Eine Relationsklasse $\mathcal{R}$ ist ein Paar $(\pi, \mathcal{I})$, wobei π ein Relationsmuster und $\mathcal{I}$ eine Menge ist, die ausschließlich Instanzen von π oder aber Subklassen von $\mathcal{R}$ enthält. Eine Relationsklasse $\mathcal{R}$' ist dann eine Subklasse von $\mathcal{R}$, wenn gilt: $\pi' \lesssim \pi$ und $\mathcal{I}' \subset \mathcal{I}$. Im Umkehrschluss ist dann $\mathcal{R}$ die Superklasse von $\mathcal{R}$'.

Damit sind die Vorbereitungen für eine Spezialisierung der Definition des semantischen Netzes getroffen. Sowohl für Knoten als auch für die Kanten eines solchen Netzes sind die Möglichkeiten zur Unterscheidung verschiedener Arten gegeben.

Definition 9 (Semantisches Netz)
Ein semantisches Netz ist ein Tupel (K, $\mathcal{K}|_K$, R, $\mathcal{R}|_R$). Dabei ist K die Menge der beteiligten Konzepte, $\mathcal{K}|_K$ die Menge der aus K folgenden Konzeptklassen, R die Menge der Relationsmuster, die zu $\mathcal{K}|_K$ gebildet werden können und $\mathcal{R}|_R$ die Menge der Relationsklassen zugehörig zu R. Zusammen definieren diese vier Mengen ein semantisches Netz mit typisierten Knoten und typisierten Relationen, die zwischen diesen Knoten bestehen.

Um so ein semantisches Netz im *Semantic Web* nutzbar zu machen, ist es notwendig, es nach OWL/RDF zu transformieren, der Notation für Ontologien im Semantic Web. OWL/RDF ist hinsichtlich der Form der Relationsmuster gegenüber semantischen Netzen eingeschränkt, weswegen deren Definition im RDF-Kontext angepasst werden muss:

Definition 10 (Relationsmuster in RDF)
Ein Relationsmuster in RDF ist ein Tripel $(\Lambda, \mathcal{S}, \mathcal{O})$, wobei Λ ein Name ist und $\mathcal{S}, \mathcal{O}$ jeweils Konzeptklassen sind, die nicht verschieden sein müssen. Das Relationsmuster definiert die Existenz einer gerichteten Verbindung von $\mathcal{S}$ nach $\mathcal{O}$, die sich mit Λ benennen lässt.

In dieser Definition wird insbesondere ausgedrückt, dass Relationen in RDF immer binär sind und eine eindeutige Richtung aufweisen. Im Gegensatz zu allgemeinen semantischen Netzen, die ungerichtete Graphen sind, handelt es sich bei RDF-Daten um gerichtete Graphen. In diesen beiden Aspekten unterscheidet sich RDF deutlich vom konkurrierenden Ansatz der Topic Maps (siehe Abschnitt 3.1.1).

Problemdefinition

Nachdem die grundlegenden Definitionen erfolgt sind, lässt sich nun das zu lösende Problem besser spezifizieren. Will man aus einem digital vorliegenden Korpus ein semantisches Netz erstellen, so sind, gemäß Def. 9 die Bestandteile des Tupels (K, $\mathcal{K}|_K$, R, $\mathcal{R}|_R$) der Reihe nach zu bestimmen.

Die Bestimmung von K ist eine intellektuelle Aufgabe, die nicht automatisiert werden kann, da K direkt davon abhängt, zu welchem Zweck das Netz erstellt werden soll. Die

Bestimmung von $\mathcal{K}|_K$ hingegen lässt sich in großem Maße automatisieren und bisherige Ontologielernsysteme setzen insbesondere an diesem Punkt an (siehe Abschnitt 4.1).

Die Festlegung der Relationsmuster für das semantische Netz ist schließlich der Punkt, ab dem die manuelle Ontologieerstellung spätestens beginnt, zeit- und kostenintensiv zu werden. An diesem Punkt setzt die vorliegende Arbeit an, indem die Automatisierung auch auf die Findung von R und die nachfolgende Bestimmung von $\mathcal{R}|_R$ ausgedehnt wird. Zu berücksichtigen ist hierbei, dass die Suche nach R nur semiautomatisch gestaltet werden kann, da eine Begutachtung durch menschliche Experten notwendig ist, um zu entscheiden, welche der möglichen Relationsmuster tatsächlich für die aktuelle Aufgabe von Belang und deshalb aufzunehmen sind. Darunter fällt auch die Bestimmung der verschiedenen Λ, für deren Benennung höchstens Vorschläge generiert werden können.

Die weitgehende Automatisierung des Prozesses auf R und $\mathcal{R}|_R$ stellt allerdings nur einen Teil der Arbeiten dar; zusätzlich ist zu untersuchen, inwieweit sich das resultierende semantische Netz in eines übertragen lässt, das den Anforderungen von RDF genügt. Dazu muss eruiert werden, ob sich die Relationsmuster des Netzes verlustfrei in solche transformieren lassen, die Def. 10 genügen; bzw. welche Informationsverluste bei der Transformation zu erwarten und hinzunehmen sind.

5.2.2 Workflow

Der sich aus der formalen Beschreibung ergebende Arbeitsablauf lässt sich auf die Reihenfolge abbilden, in der die Anforderungen aus Abschnitt 2.3.1 stehen. Zunächst muss eine Festlegung der Konzepte erfolgen, die grundlegend für das semantische Netz sind, gefolgt von der automatischen Suche nach Instanzen dieser Konzepte, also der Erstellung der Konzeptklassen. Daran schließt sich die Suche nach Relationen an, die das Netz aufspannen.

Während die formale Betrachtungsweise das Ziel definiert, nämlich die Entwicklung von Algorithmen zum semiautomatischen Erstellen semantischer Netze, erweitert die Anforderungsliste dieses Ziel um den dafür benötigten Prozess, der sich in ein Software-System abbilden lässt. Abbildung 5.1 zeigt den Arbeitsablauf schematisch und verortet die Anforderungen darin.

Auf der linken Seite der Abbildung sind verschiedene Datenquellen aufgetragen, z.B. Daten in tabellarischer Form, Textdokumente oder relationale Datenbanken. Diese dienen, zusammen mit den vorher festgelegten Konzepten, als Eingaben für das Modul zur Extraktion von Konzeptinstanzen, in der Graphik mit **Automatische Eigennamenerkennung** überschrieben. Darin sind die Anforderungen A3 und A4 zusammengefasst, da die Annotierung von Beispielen der einzelnen Konzepte für die Eigennamenerkennung benötigt wird. Das Resultat dieses Verarbeitungsschrittes ist eine Menge von Konzeptklassen, jede Instanz versehen mit ihrem Fundort in den Eingabedaten. Diese Menge stellt die Basis für die Arbeit des wichtigsten Moduls des Systems dar: Die **Semi-automatische Netzerstel-**

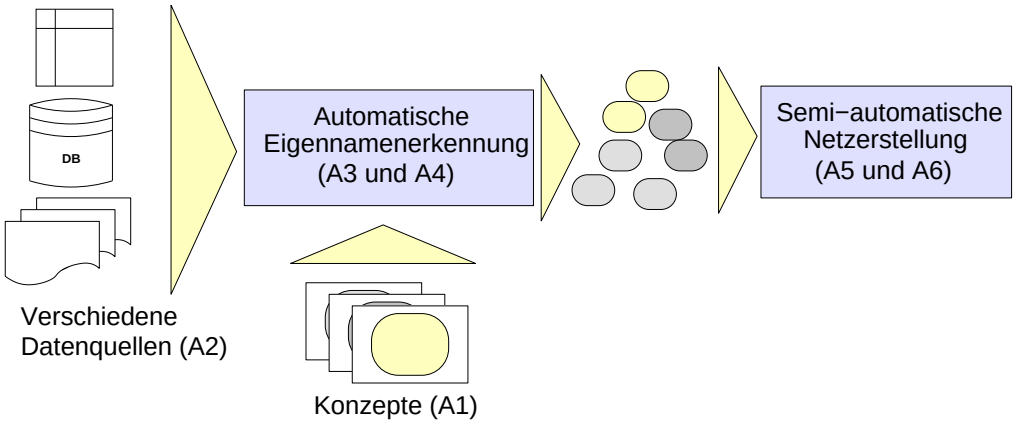


Abbildung 5.1: Workflow des Bereichs Netzerstellung

lung. Hierin sind die beiden Anforderungen A5 und A6 zusammengefasst. Die Ausgabe dieses Moduls schließlich ist das semantische Netz, das den Datenbestand beschreibt.

Die nachfolgenden Abschnitte behandeln die einzelnen Schritte dieses Arbeitsablaufs.

5.2.3 Import und Verarbeitung von Konzepten

In der Einleitung zu diesem Abschnitt ist bereits erwähnt worden, dass die Definition von Konzepten die Arbeit ist, die sich bei der Ontologieerstellung noch sehr gut manuell durchführen lässt. Hierfür gibt es mehrere Gründe. Zuvorderst der, dass die Definition der grundlegenden Konzepte rein inhaltlich motiviert ist und daher ein maschineller Prozess nicht zielführend wäre. Weiterhin ist die Anzahl der benötigten Konzepte typischerweise nicht besonders groß und wenn doch, gibt es üblicherweise Referenzliteratur, die als Hilfsmittel bei der Definition herangezogen werden kann.

Für den Arbeitsablauf ist die Existenz vordefinierter Konzepte eine harte Vorbedingung – ohne sie kann nicht einmal die Eigennamenerkennung durchgeführt werden. Daher muss dem Ablauf ein Schritt vorangestellt werden, in dem diese Konzepte erzeugt werden. Zur Erleichterung dieser Arbeit kann eines der Hilfsmittel zum Einsatz kommen, die in Abschnitt 4.2 besprochen worden sind. Dabei sollte darauf geachtet werden, dass der Export in RDF, RDFS oder OWL statt findet und nicht in einem evtl. verwendeten proprietären Format des jeweiligen Programms. Ferner sollte beim Design darauf geachtet werden, dass die Konzepte nicht zu feingliedrig geraten. Gerade graphische Modellierungswerkzeuge verleiten dazu, starken Gebrauch der Klassen-Subklassen-Beziehung zu machen. Wenn sich aber die einzelnen Unterklassen anhand textueller Beispiele nicht gut voneinander unterscheiden lassen, dann kann deren Erkennung auch nicht gut automatisch gelernt werden, was zu Fehlern und damit zu manueller Mehrarbeit führt.

Bei der Modellierung sollte außerdem zwischen Konzepten unterschieden werden, die

einen eigenen Begriff beschreiben und solchen, die einen Zustand anderer Konzepte beschreiben.

Hierzu ein Beispiel: Angenommen, es soll ein Netz erstellt werden, in dem das Organigramm eines Unternehmens dargestellt ist. Von entscheidender Bedeutung in diesem Netz sind also Personen und ihre Position im Unternehmen. Der naive Ansatz bestünde darin, Person und Funktion im gleichen Konzept zu modellieren, eine Instanz davon wäre dann z.B. " CEO Alfons Mustermann". Solange sich an dem Zustand des Organigramms nichts ändert, ist das kein Problem. Sobald allerdings eine Person ihre Funktion ändert, muss die zugehörige Instanz geändert werden.

Dieses Problem lässt sich vermeiden, indem von vornherein die Konzepte sauber getrennt werden. Es gibt *Personen*, die innerhalb des Unternehmens verschiedene *Rollen* bekleiden können. Solange eine Person eine Rolle ausfüllt, erhält die Personeninstanz eine Verbindung zu der entsprechenden Rolle; legt sie die Rolle ab, wird die Verbindung entfernt. Es werden also nicht Instanzen geändert, sondern lediglich ihre Verknüpfungen.

Dieses Vorgehen ähnelt der Komposition in der objektorientierten Modellierung, auch dort werden komplexere Klassen aus spezialisierten Klassen zusammengesetzt. Das hat den Vorteil, dass sich einzelne dieser spezialisierten Klassen bei Bedarf zur Laufzeit gegen andere austauschen lassen - das Verhalten der Klassen lässt sich also dynamisch über Anlage oder Löschung einer Relation (in diesem Fall die Verwendung einer bestimmten, verhaltensbestimmenden Klasse) anpassen.

Die Instanzen solcher verhaltens- oder auch zustandsbeschreibenden Konzepte lassen sich in der Regel gut aufzählen, so dass sie schon im Vorfeld des Prozesses unter Zuhilfenahme vorliegender Informationen modelliert werden können. Dazu zählen z. B. Hierarchiesysteme wie Rangfolgen und Organigramme, aber auch geopolitische Daten (Kontinente, Staaten, Bundesländer, Kommunen...). Die für das bessere Verständnis dieser Daten notwendigen Relationen (übergeordnet-untergeordnet, Lagebeziehungen) lassen sich vergleichsweise einfach modellieren und anwenden. Solcherart ausgezeichnete Konzeptklassen können in den folgenden Schritten zur Verbesserung der automatischen Ergebnisse verwendet werden.

Import

Sind die Konzepte erst einmal von den Fachexperten ausgewählt worden, so gestaltet sich deren Import in das geplante System verhältnismäßig einfach. Wenn die Daten in einem Format wie RDF(S) oder OWL vorliegen, das letztlich auf XML beruht, können sie einfach in andere Formen gebracht werden, die von den internen Modulen für eine weitere Verarbeitung benötigt werden. So benötigt die automatische Eigennamenerkennung üblicherweise nur die Namen der zu lernenden Konzepte. Je nach eingesetztem Verfahren benötigt man weiterhin Beispieldaten für typische Vorkommen der Konzepte, um den maschinellen Lernprozess anzuleiten. Sollten sich einzelne Konzepte gut aufzählen lassen, so können deren

Instanzen natürlich auch direkt in Listenform importiert werden. Diese Klassen müssen dann nicht mehr gelernt werden, sondern können direkt zum Markieren der Vorkommen im Korpus eingesetzt werden.

Verarbeitung

Für die Verwendung im semantischen Netz gibt es zwei Optionen. Die erste besteht darin, die vordefinierten Konzepte mit den automatisch gefundenen Instanzdaten zu vereinen, so dass ein großes semantisches Netz dabei herauskommt, das nicht nur logisch, sondern auch in der Datenhaltung eine Einheit bildet, also quasi in einer Datei notiert wird. Die zweite Option besteht darin, die automatisch erzeugten Daten über Querverweise mit den Informationen über die Konzepte zu verbinden, sie aber ansonsten getrennt voneinander zu halten. Das entspricht einer Trennung in Strukturdaten und Inhalten. Das erleichtert die Weiterverwendung einzelner Teile der Ontologie in anderen Anwendungen deutlich, weshalb es auch durch das W3C empfohlen wird (z. B. durch Trennung der Daten in Strukturbeschreibung als RDFS oder OWL und der eigentlichen Instanzdaten in RDF, die sich auf diese Strukturbeschreibungen beziehen).

5.2.4 Import unterschiedlicher Datenformate

Für die Organisation digitaler Daten gibt es eine schier unüberschaubare Anzahl verschiedener möglicher Formate. Wie bereits in der Erläuterung zu Anforderung A2 in Abschnitt 2.3.1 erwähnt, ist es illusorisch, jedes beliebige Format unterstützen zu wollen. Dennoch ist es sinnvoll, eine gewisse Auswahl populärer Formate importieren zu können und sich nicht nur auf plain text zu beschränken, da so die Schranken zur Bereitstellung interessanter Datensammlungen gesenkt werden können. Bei der Realisierung einer Importschnittstelle ist die Berücksichtigung einer anderen Anforderung hilfreich: Anforderung A7 in Abschnitt 2.3.2 fordert eine Systemarchitektur, die sich um neue Schnittstellen und Module erweitern lässt. Das gilt im besonderen Maße für die Importschnittstelle.

Ansätze für Importschnittstellen

Für das Design einer Importschnittstelle unter diesen Bedingungen gibt es grundlegend zwei Möglichkeiten: Entweder werden die Daten so importiert wie sie sind und die weiteren Verarbeitungsschritte werden um Funktionalitäten erweitert, die ein Umgehen mit den verschiedenen Formaten erlauben. Oder die Datenformate werden beim Import in ein Format transformiert, mit dem intern gearbeitet werden kann. Jede dieser Vorgehensweisen hat ihre eigenen Vor- und Nachteile:

Die Vorteile des reinen Imports sind Zweierlei: So werden die Daten nicht verändert, sondern in der gleichen Form verarbeitet, in der sie vorliegen. Dadurch werden Trans-

formationsfehler ausgeschlossen. Weiterhin erlaubt dieses Vorgehen, Algorithmen auf die Formate maßzuschneidern, so dass deren Fähigkeiten optimal ausgeschöpft werden können.

Nachteilig wirkt sich hingegen der beträchtliche Aufwand aus, der für die Integration eines neuen Formats getrieben werden muss. So muss jedes Modul, das dieses Format verarbeiten soll, um Methoden ergänzt werden, die speziell für dieses Format angepasst sind, im Endeffekt muss also eine komplett neue Fassung jedes Moduls geschrieben werden – und da man die Fremdformate nicht unter Kontrolle hat, unter Umständen bei jeder Revision wieder. Durch diesen Aufwand muss der Leidensdruck für die Integration eines neuen Formats schon sehr hoch sein, ehe diese Integration in Angriff genommen wird.

Das Transformationsverfahren weist ebenfalls zwei Vorteile auf: Durch die Wandlung der Importe in interne, kontrollierte Formate ist es möglich, die Spezifikation diese Formate als Schnittstelle nach außen bekannt zu geben. Damit werden andere Entwickler in die Lage versetzt, einen Transformator für ein Format zu schreiben, das sie unterstützt sehen möchten, ohne dass sie sich in die Spezifika aller Module des Systems einarbeiten, bzw. diese überhaupt kennen müssen. Intern ergeben sich ebenfalls Vorteile, da sich der Aufwand für die Realisierung der Module dadurch in Grenzen hält, dass nur für bekannte Format entwickelt werden muss, die man voll unter Kontrolle hat.

Die Nachteile dieses Ansatzes liegen darin, dass, sobald die Schnittstelle einmal bekannt ist und benutzt wird, Änderungen daran alle Transformatoren zu brechen drohen, die man zum Teil nicht einmal selbst entwickelt hat. Man muss also mit den Schnittstellendefinitionen sehr vorsichtig umgehen. Außerdem werden die externen Formate letztlich auf einige wenige gemeinsame Nenner reduziert, wodurch besondere Features neuerer Formate u.U. gar keine Berücksichtigung finden können, da zur Zeit der Definition niemand auch nur daran gedacht hat, dass diese interessant sein könnten.

Welches dieser beiden Verfahren man letztlich wählt, ist ein Stück weit eine Glaubensfrage, ähnlich wie die Diskussionen im Bereich der verteilten Informationssysteme, ob nun föderierte Suchsysteme oder Harvesting-Ansätze besser geeignet sind, um verteilte, heterogene Datenbanken zentral durchsuchbar zu machen. Hierbei ist das Importverfahren analog zu den föderierten Suchsystemen zu sehen, in denen untergeordnete Systeme mit evtl. anderen Datenmodellen über sog. Wrapper angesteuert werden, die Arbeit also beim Hauptserver liegt. Das Transformationsverfahren hingegen ähnelt eher dem Harvesting-Ansatz, der z.B. von der Open Archives Initiative[2] (OAI) mit dem Open Archives Initiative Protocol for Metadata Harvesting [67] verfolgt wird. Hierbei stellt jeder angeschlossene Server seine Daten in einem vorher definierten Format zur Verfügung, das vom Zentral-Server, dem *Harvester*, in regelmäßigen Intervallen per pull-Verfahren "geerntet" wird.

Für die Bedürfnisse des geplanten Systems ist die Verwendung des Transformationsansatzes die Wahl, die zu einem robusteren System führt. Dadurch werden die für die Unterstützung neuer Formate notwendigen Arbeiten auf eine Stelle im System konzentriert

[2]siehe www.openarchives.org

und zusätzlich ihr Umfang deutlich reduziert. Dieser Ansatz passt besser zur Forderung nach einer erweiterbaren Architektur, da die Erweiterungen nur von einigen internen Formaten abhängen. Sollte sich für eine Erweiterung die Notwendigkeit ergeben, ein Format ändern zu müssen, so lässt sich genau abschätzen, welchen Aufwand das intern verursacht.

Um eine später evtl. notwendige Veränderung des Formats zu vereinfachen, bietet es sich an, die internen Formate in XML zu notieren. Dadurch lassen sich einfache Änderungen in der Form optionaler Felder vornehmen, so dass unveränderte Transformatoren immer noch gültige Daten liefern, auch wenn sie die für die neuen Funktionalitäten benötigten Daten nicht enthalten. Auf diesen Punkt wird in Abschnitt 5.3 noch einmal vertieft eingegangen werden.

Auswahl der Formate

Durch die Festlegung auf ein System, das nach außen eine Reihe von Formaten bekannt gibt, in die Importdaten gewandelt werden müssen, ergeben sich mehrere Fragen: Wieviele solcher Formate gibt es? Nach welchen Kriterien werden sie ausgewählt?

Wieviele Formate werden bekannt gegeben? Das gewählte Vorgehen erlaubt ein bedarfsgetriebenes Veröffentlichen der Schnittstellen. Im einfachsten Fall wird nur ein Textformat benötigt, gestaltet nach den Bedürfnissen der Eigennamenerkennung. So ein System könnte dann allerdings nur Textdaten verarbeiten können. Möchte man auch tabellarische Daten für die Ausgestaltung des Netzes hinzuziehen, so wird ein weiteres Format benötigt, in dem die Konkordanz der einzelnen Tabellenspalten mit den Attributen der vordefinierten Entitätsklassen hergestellt wird. In dieses Format ließen sich dann alle tabellarischen Daten konvertieren, egal, ob es sich dabei um Daten aus Spreadsheet-Anwendungen oder Tabellen aus relationalen Datenbanken handelt.

Für weitere Nutzungswege werden entsprechend andere Formate benötigt. Soll das System auch Bilder, Videos oder Tonspuren auswerten können, so müssen die dafür benötigten Daten in einem entsprechenden Format zur Verfügung gestellt werden.

Nach welchen Kriterien werden sie ausgewählt? Hier gilt das gleiche Argument wie bei der ersten Frage: Die Wahl des Systems erlaubt eine bedarfsgetriebene Auswahl der Formate, die als Importe entgegengenommen werden. Wenn für die Auswertung von Textdokumenten eine Unterteilung in Paragraphen benötigt wird, so wird man ein Format wählen oder selbst definieren, das diese Unterteilungen anbietet. Gleiches gilt für die anderen Medienarten Bild, Ton und Video. Allerdings wird man in diesen Fällen die Daten höchstwahrscheinlich nicht in XML, sondern in einem Binärformat entgegen nehmen. Aber auch hier ist die Auswahl rein bedarfsgetrieben. Wenn lediglich MP3 benötigt wird, wird auch nur MP3 als Importformat entgegengenommen.

Das Transformationsverfahren erlaubt die Konzentration auf die Bedürfnisse des Systems, das man selbst aufbaut und sorgt dafür, dass die kleinstmögliche Anzahl von Formaten als Schnittstelle zum Import angeboten werden. Unter Umständen erhöht das den Aufwand, der vor der Verwendung des eigentlichen Systems getrieben werden muss, sorgt aber auf der anderen Seite dafür, dass das System selbst so wenig von der jeweiligen Problemdomäne abhängt wie möglich. Das erweitert das Einsatzspektrum deutlich, da beim Einsatz in einem neuen Gebiet auch nur die Komponenten neu zu schreiben sind, die direkt von der Domäne abhängen.

Zusammenfassung

Für diesen Abschnitt bleibt zusammenfassend festzuhalten, dass für den Import von Daten verschiedener Formate systemintern eigene Formate definiert werden, in die externe Daten beim Importvorgang transformiert werden. Die dadurch zu erzielenden Vorteile, also eine klare Schnittstelle nach außen und verringerte Koppelung der Module an Importspezifika, überwiegen die Nachteile, nämlich die suboptimale Ausnutzung der einzelnen Formate, sowie erwartete Probleme bei Änderung der Schnittstellen nach außen, deutlich. Zusätzlich wird die Verbindung verschiedener Softwaresysteme vereinfacht, da diese über klar definierte Schnittstellen miteinander verbunden werden.

5.2.5 Automatische Eigennamenerkennung

Das Modul zur automatischen Eigennamenerkennung ist das erste, in dem automatische Verfahren zum Einsatz kommen, mit denen die manuelle Arbeit bei der Ontologieerstellung verringert werden soll. Es deckt die Anforderungen A3 und A4 ab. Eigennamenerkennung, auf englisch *Named Entity Recognition* (NER), bezeichnet eine Disziplin aus dem Gebiet der Informationsextraktion, die sich mit der automatischen Extraktion von Instanzen verschiedener Entitätsklassen beschäftigt. Klassisch sind das Personen-, Orts- und Organisationsnamen, mittlerweile ist die Bandbreite aber deutlich ausgeweitet worden. Das Hauptanwendungsgebiet für NER ist die Unterstützung bei der Durchsuchung großer Textmengen nach interessanten Bestandteilen, etwa nach Vorkommen bestimmter Firmen- oder Personennamen. Ein typisches Beispiel hierfür sind Börsentickermeldungen. Aufgrund des industriellen Interesses an solchen Lösungen ist die NER ein ausführlich beforschter Bereich, für den eine große Zahl verschiedener Algorithmen entwickelt worden ist[3]. Wurde früher noch mit regelbasierten Verfahren gearbeitet, so ist heutzutage der Einsatz maschineller Lernverfahren nicht mehr wegzudenken. In [64], Seite 509, findet sich hierzu folgendes Zitat: *Was die Entwicklung von Teilaufgaben [der Informationsextraktion] betrifft, werden im Bereich der Erkennung von Eigennamen praktisch nur noch maschinelle Lernverfahren eingesetzt.*

[3] In [93] gibt Kapitel 4 einen Überblick über die verschiedenen Ansätze

Verfahren zur Eigennamenextraktion finden sich auch in generischen Architekturen für Text Engineering wieder, so z.B. in GATE (Generic Architecture for Text Engineering), einem weit verbreiteten System der University of Sheffield [34], das unter einer Open-Source-Lizenz zur Verfügung steht. Die darin enthaltenen Eigennamenerkenner sind trainierbar, d.h. mit passenden Beispielen können sie für die Erkennung praktisch jeder Entitätsklasse angepasst werden. GATE wird in der wissenschaftlichen Welt häufig eingesetzt und seit Jahren engagiert gepflegt und weiter entwickelt. Da die Eigennamenerkennung als solche nicht im Fokus dieser Dissertation steht und es bereits Softwarekomponenten gibt, die man in einen Arbeitsablauf einpassen kann, wird davon Abstand genommen, selbst eine Eigennamenerkennung zu entwickeln.

Zum Training der Erkenner für die Entitätsklassen ist eine Schnittstelle nötig, mit der Beispiele annotiert werden können. Dazu wird eine Teilmenge der Dokumentensammlung verwendet. Bei deren Auswahl ist darauf zu achten, dass sie möglichst repräsentativ für die Sammlung sind, da ansonsten die Qualität der Ergebnisse der automatischen Erkenner leidet. Zum Sammeln von Beispielen gibt es bereits verschiedene Tools, etwa als Teil von GATE, allerdings wenden sich diese zumeist an Experten auf dem Gebiet des Text Engineering, so dass sie nur bedingt für den unterstützungsfreien Einsatz durch Domänenexperten geeignet sind. Das Annotationstool WALU (für Wikinger Annotations- und Lernumgebung) von der Computerlinguistik der Universität Duisburg-Essen ist ein Beispiel für eine Annotationsumgebung, die für die Nutzung durch Domänenexperten ausgelegt ist[4].

5.2.6 Semiautomatische Relationserkennung

In den vorangegangenen drei Abschnitten wurden die Grundlagen für die Arbeiten gelegt, die zentral für den Mehrwert des vorgeschlagenen Systems stehen und gleichzeitig den Fokus dieser Arbeit bilden. Die Mechanismen für den Datenimport sind genauso behandelt worden wie die Definition und der Import der zu verwendenden Konzepte. Auf deren Basis wird eine Eigennamenerkennung durchgeführt, mit der die Konzeptklassen erzeugt werden. Die enthaltenen Instanzen zu einem Netz zu verknüpfen, ist die Aufgabe des hier beschriebenen Moduls. Es deckt die Anforderungen A5 und A6 ab.

Im Kern geht es darum, ein Verfahren zu entwickeln, das für Mengen von Instanzen verschiedener Konzeptklassen ein semantisches Netz konstruiert, das diese in einen sinnvollen, nicht-trivialen Zusammenhang stellt. Die einzelnen Instanzen bilden dabei die Knoten des Netzes, die Relationen zwischen den Instanzen sind seine Kanten. Das impliziert, dass es unterschiedliche Kantenarten geben muss, zumindest aber beschriftete Kanten, um nicht alle Relationen auf eine triviale Relation “X hat zu tun mit Y“ reduzieren zu müssen.

Um diese Aufgabe zu lösen, empfiehlt es sich, sie in zwei Schritten anzugehen. Im ersten Schritt werden die automatisch erkennbaren Relationen aus einem repräsentativen Teil des Korpus extrahiert. Diese Relationen werden dann den Domänenexperten zur Begutachtung

[4]WALU wird in Abschnitt 6.1 näher erläutert

vorgelegt. Die Aufgabe der Auswahl der für das geplante Netz relevanten Relationen obliegt den Experten, da nur diese die inhaltliche Qualität der Relationen beurteilen können. In einem zweiten Schritt werden die ausgewählten Relationen benutzt, um die in den übrigen Dokumenten vorkommenden Relationen zu klassifizieren.

Wissenschaftlich gesehen fällt die beschriebene Aufgabe in die Disziplin des Relation Learning. Relevante wissenschaftliche Arbeiten hierzu sind in Abschnitt 4.3 beschrieben worden. Die dort behandelten Ansätze ließen sich in zwei Gruppen unterteilen: Die Ansätze der ersten Gruppe dienten zum Lernen vorher festgelegter Relationen, um anschließend Klassifikationsaufgaben auf bisher unbekannten Textstellen durchführen zu können. Die zweite Gruppe enthielt Arbeiten, in denen eine mit Entitäten annotierte Textmenge auf Regelmäßigkeiten hin untersucht wurde, um daraus das Vorhandensein verschiedener semantischer Relationen zwischen den Entitäten abzuleiten. In der ersten Gruppe sind die Relationen also apriori bekannt, wohingegen sie in der zweiten erst gefunden werden sollen. Die unterschiedlichen Zielsetzungen schlagen sich auch in den Bezeichnungen nieder: Die Ansätze aus der ersten Gruppe verwenden den Begriff *Relation Extraction*, während die der zweiten Gruppe üblicherweise den Begriff *Relation Discovery* verwenden. Dieser zur Zeit noch informellen Unterteilung wird diese Arbeit im weiteren Verlauf ebenfalls folgen.

Die Aufgabe im ersten Schritt ist dem Relation Discovery zuzurechnen, da zwar anzunehmen ist, dass eine gewisse Kenntnis über die zu erwartenden Relationen besteht, das Material als solches aber nicht vollständig erschlossen ist. Deshalb ist es sinnvoll, ein Verfahren zu wählen, das alle Relationen aufzeigt, die automatisch erkannt werden können. Aus diesen kann dann eine Untermenge ausgewählt werden, die im Netz modelliert werden soll. Dabei ist zu berücksichtigen, dass Relationen, die Menschen trivial erscheinen mögen, u. U. für das automatische Schließen (engl. *Reasoning*) von großer Wichtigkeit sind. Gerade vermeintlich triviale Fakten sind dies nur von der menschlichen Warte aus, maschinell lassen sie sich hingegen nicht voraussetzen. Daher sollten auch Relationen berücksichtigt werden, deren Informationsgewinn auf den ersten Blick nur stark begrenzt zu sein scheint.

Diese Gründe sprechen für ein Verfolgen eines Ansatzes, der nicht auf der Basis einer vorher bestimmten Menge von Relationen arbeitet, sondern diese durch die Auswertung statistisch relevanter Häufungen bestimmt. Dabei wird implizit davon ausgegangen, dass Relationen eine gewisse Regelmäßigkeit aufweisen, d.h. mehr als einmal auftreten und verschiedene Vorkommen einer bestimmten Relation auch auswertbare Ähnlichkeiten aufweisen. Zusätzlich wird vorausgesetzt, dass Entitäten, die miteinander in Relation stehen, auch in einer gewissen räumlichen Nähe im Text zu finden sind. Diese Voraussetzungen sind weniger fordernd, als sie vielleicht zu sein scheinen. Tatsächlich gehen sie jedoch nicht weiter als Anforderungen an natürlichsprachliche Texte im Allgemeinen. Viele Relationen werden in Texten durch Verben ausgedrückt; die mit ihrer Verwendung verbundenen grammatikalischen Regeln stellen prinzipiell die gleichen Anforderungen.

Gleichwohl hat das automatische Verfahren auch seine Grenzen. So wird es immer Relationen geben, die nicht erkannt werden, entweder, weil sie zu selten vorkommen, oder

weil sie durch andere, ähnlich aussehende Relationen subsummiert werden. Auf der anderen Seite stellt allerdings jede automatisch gefundene Relation eine Arbeitserleichterung beim Erstellen des semantischen Netzes dar.

Der zweite Schritt im geplanten Ablauf hingegen ist eine klassische Klassifikationsaufgabe, denn zu diesem Zeitpunkt ist das Set der gewünschten Relationen bereits bekannt.

Relationserkennung unter Verwendung von Assoziationsregeln und Clustering

Zu Beginn der Entwicklung dieses Algorithmus stand der Wunsch, die Häufigkeiten der Klassenkombinationen in den Entdeckungsprozess für Relationen einbeziehen zu können. Das hätte den Vorteil, dass man sich bei der weiteren Verarbeitung auf die Kombinationen konzentrieren könnte, die den größten Abdeckungsgrad innerhalb des Korpus versprechen. Darüber hinaus ließen sich so für eine tiefere Analyse Kombinationen herausgreifen, die zwar vielleicht sehr kleine Anteile an der Gesamtmenge haben, aber thematisch interessant erscheinen.

Die Relationserkennung sollte auf einem Subset der Dokumentenmenge durchgeführt werden, idealerweise auf den gleichen Dokumenten, die von den Domänenexperten auch zur Annotation der Beispiele verwendet worden sind. So kann davon ausgegangen werden, dass die zur Ermittlung der Relationen verwendeten Annotationen eine hohe Qualität haben, also nicht von vornherein auf verrauschten Daten gearbeitet werden muss.

Zunächst müssen die annotierten Dokumente einer Vorverarbeitung unterzogen werden. Dazu werden die Texte als Mengen voneinander unabhängiger Sätze verstanden, aus denen alle solche Sätze im Vorhinein entfernt werden können, die höchstens eine Entität enthalten. Für jeden verbliebenen Satz werden nun die darin vorkommenden Entitätsklassen notiert. So gelangt man zu einer Repräsentation, die den Einsatz des *apriori-Algorithmus*(siehe Abschnitt 3.3.1) erlaubt. Dieser Algorithmus ermöglicht die Erstellung von Assoziationsregeln, die Zusammenhänge zwischen verschiedenen Entitätsklassen ausdrücken können.

Die Güte von Assoziationsregeln richtet sich nach zwei Maßen: *Support* und *Confidence*. Im Algorithmus können Schwellwerte für die beiden Maße angegeben werden, die beide überschritten sein müssen, ehe eine Regel aufgestellt wird. Die Regeln identifizieren die Kombinationen, die einerseits einen gewissen Mindestanteil des Materials abdecken (Support) und andererseits einen statistisch signifikanten Zusammenhang der an der Kombination beteiligten Klassen erkennen lassen (Confidence).

Dadurch kann man im weiteren Verlauf diejenigen Kombinationen prioritär bearbeiten, die die besten Ergebnisse versprechen, sei das in Hinblick auf die Abdeckung oder den Zusammenhang der Regeln.

Die Identifikation interessanter Klassenkombinationen ist allerdings nicht hinreichend, da sich hinter einer Assoziationsregel nicht nur *eine* Relation zwischen den beteiligten Klassen verbergen muss. In der in Abschnitt 4.3.1 erwähnten Arbeit von Hasegawa et al. fanden

sich zum Beispiel für die Kombination PERSON - GPE[5] im Korpus 36 verschiedene Relationen – und das bei nur 177 Vorkommen der Kombination[6]. Ergänzend ist zu erwähnen, dass die Autoren in ihrer Arbeit nur binäre Relationen berücksichtigt haben, wohingegen durch den Apriori-Algorithmus auch Assoziationsregeln erstellt werden können, die mehr als zwei beteiligte Klassen haben.

Daher gilt es, in weiteren Arbeitsschritten die in den Assoziationsregeln enthaltenen Relationen voneinander zu trennen. Dafür wird ein Clusteringschritt eingeschoben, der die verschiedenartigen Relationsmuster voneinander trennt.

Clustering Zu jeder Assoziationsregel gehört eine Liste mit den Sätzen, in denen die zur Regel gehörenden Konzeptinstanzen vorkommen. Aus den Sätzen werden die längstmöglichen Wortsequenzen extrahiert, die zwischen den markierten Instanzen liegen. Im Fall einer Assoziationsregel mit mehr als zwei Elementen wird die Sequenz gewählt, die zwischen der ersten und der letzten im Satz markierten Instanz steht.

Diese Sequenzen bilden die Grundlage für das Clustering. Um eine möglichst große Generalisierung der Sequenzen zu erreichen, werden die Instanzen gegen die Konzepte getauscht, zu denen sie gehören. Dazu werden sie in Wortvektoren übertragen, deren Gewichte mit dem tf*idf-Algorithmus bestimmt werden. Dadurch wird u.a. sichergestellt, dass die Konzepte selbst keinen Anteil an der späteren Bestimmung von Vektorähnlichkeiten haben, da deren Vorkommen in jeder Sequenz sie für die Bestimmung der Ähnlichkeit nichtig werden lässt[7]. Ließe man die Instanzen im Satz stehen, so würden sie die Ergebnisse in großem Maße beeinflussen. In diesem Schritt sind wir aber an den für die Relation ausschlaggebenden Mustern in den Sätzen interessiert, nicht an ihren Subjekten bzw. Objekten.

Das Clustering wird auf der Basis der Wortvektoren durchgeführt. Diese werden einem hierarchischen Clusteringverfahren, dem agglomerativen Clustering, unterzogen. Dabei ist jeder Wortvektor zu Beginn ein eigener Cluster. Im weiteren Verlauf des Algorithmus werden iterativ Cluster zusammengefasst, bis ein Punkt erreicht wird, an dem die Cluster entweder zu unterschiedlich sind, als dass sie weiter zusammengefasst werden könnten oder aber nur noch ein Cluster übrig geblieben ist.

Zur Definition des Abbruchkriteriums gibt es verschiedene Strategien; gängig sind *single linkage*, *complete linkage* und *average linkage*, die jeweils gegen eine Maximaldistanz berechnet werden. Die Unterschiede werden anhand der Formeln deutlich, in denen $\mathcal{A}$ und $\mathcal{B}$ jeweils Cluster bezeichnen:

single linkage clustering: $\min\{\mathrm{d}(\alpha, \beta)\colon \alpha \in \mathcal{A}, \beta \in \mathcal{B}\}$

complete linkage clustering: $\max\{\mathrm{d}(\alpha, \beta)\colon \alpha \in \mathcal{A}, \beta \in \mathcal{B}\}$

[5]GPE= Geo-Political Entity, definiert als ein klar abgegrenzter geographischer Bereich mit einer Regierung.

[6]Siehe [103], Seite 4

[7]Das liegt in der Konstruktion des tf*idf-Algorithmus begründet, siehe Abschnitt 3.3.3.

average linkage clustering: $\frac{1}{|\mathcal{A}||\mathcal{B}|}\sum_{\alpha\in\mathcal{A}}\sum_{\beta\in\mathcal{B}}\mathrm{d}(\alpha,\beta)$

Die Funktion d beschreibt in allen drei Formeln eine Distanzfunktion, die zum Messen der Entfernung zweier Vektoren herangezogen werden kann. Üblicherweise wird die Kosinus-Ähnlichkeit verwendet, also $\frac{\alpha\bullet\beta}{|\alpha||\beta|}$.

Die drei Cluster-Strategien unterscheiden sich in der Härte der Abbruchkriterien. So ist beim single linkage clustering lediglich erforderlich, dass das Minimum der Entfernungen kleiner als die Maximaldistanz ist, es muss also lediglich einen Vektor geben, der die Bedingung erfüllt. Dagegen muss beim complete linkage clustering das Maximum der Entfernungen kleiner sein als die Maximaldistanz, d.h. alle Vektoren des Clusters müssen das Kriterium erfüllen. Average linkage clustering liegt zwischen den beiden Extremen, dort muss das Mittel der Entfernungen kleiner als die Maximaldistanz sein, es darf also Ausreißer geben, solange im Mittel das Entfernungskriterium eingehalten bleibt.

Die Wahl des passenden Kriteriums hängt unter anderem von der Art des zur Verfügung stehenden Materials ab. Grundsätzlich lässt sich festhalten, dass das single linkage clustering in einer geringeren Anzahl von Clustern resultiert, die damit potenziell ein breiteres Spektrum an Bedeutungen abdecken. Das kann sowohl dazu führen, dass unterschiedliche Bedeutungen miteinander in einem Cluster vermischt werden, als auch dazu, dass übergeordnete Strukturen entstehen, die Bedeutungen von Mustern generalisieren.

Die Verwendung von complete linkage clustering hingegen ist dann angebracht, wenn die vorliegenden Muster eine kleinere Varianz untereinander aufweisen, da sich so, gekoppelt mit einem passenden Schwellwert, eine hinreichende Trennschärfe der Relationen doch noch erreichen lässt. Grundsätzlich führt allerdings der Einsatz dieses Kriteriums zu einer größeren Anzahl von Clustern.

Auswertung des Clustervorgangs

Der Clustervorgang erzeugt für die Muster jeder Assoziationsregel eine Menge von Clustern. Jeder dieser Cluster entspricht einer Relation. Um allerdings von weiterführendem Nutzen zu sein, bedarf es weiterer Arbeitsschritte. So ist es erforderlich, die Cluster mit einem Namen zu versehen, der die Art der beinhalteten Relation bezeichnet. Diesen Vorgang nennt man auch *labeling*. Je nach Beschaffenheit des Ausgangsmaterials lässt sich dieser Vorgang in unterschiedlichem Maße automatisieren.

Verfahren zur Namensfindung unterscheiden sich primär darin, welche Wortarten sie verwenden, um an Namen für die Relation zu kommen. Einerseits sind dies Verben, denn diese drücken am direktesten die Art der Relation aus, die zwischen verschiedenen Konzepten bestehen. Andererseits lassen sich Namen für die Cluster auch über das Vorkommen von Nomina erzeugen, die für den Cluster charakteristisch sind. Beiden Vorgehensweisen ist gemein, dass ihre Ergebnisse abschließend den Domänenexperten vorgelegt werden sollten, damit diese eine endgültige Auswahl der in das semantische Netz zu integrierenden

Relationen treffen können.

Nachfolgend werden automatische Verfahren beschrieben, mit denen sich Namen für die Cluster auf die verschiedenen Arten erzeugen lassen. Darüber hinaus ist es natürlich auch möglich, die Labels der Relationen von Experten manuell bestimmen zu lassen.

Labeling über Verben Dieses Verfahren beruht auf der Ansicht, dass die Art der Beziehung zwischen unterschiedlichen Konzepten am direktesten durch das Verb ausgedrückt wird, das sie verbindet. Neben einem Namen für die Relation lässt sich so über die Position der Konzepte im Satz auch Näheres über die Relation selbst in Erfahrung bringen: Subjekt und Objekte lassen sich auf grammatikalischer Basis unterscheiden. So kann also auch die Richtung der Relation bestimmt werden.

Allerdings erfordert dieses Vorgehen eine grammatikalische Erschließung der Sätze, aus denen die Vorkommen der Assoziationsregeln stammen. Gerade das Deutsche ist ein Beispiel dafür, dass das Verb im Satz auch hinten stehen kann - hat man also für das Clustering nur einen Ausschnitt der Assoziationsregeln betrachtet, muss man für das Labeling der Cluster wieder den ganzen Satz betrachten. Auf den Sätzen wird ein sog. Parts-of-Speech tagging durchgeführt, wodurch jedes Wort mit Informationen über seine Wortart und den Kasus versehen wird, in dem es im Satz steht. So lassen sich das Verb des Satzes sowie das Subjekt und die Objekte identifizieren.

Mit dieser Information kann dann für jedes Vorkommen in einem Cluster das Verb bestimmt werden, das die Relation ausdrückt. Normalerweise nimmt man dafür die Grundform des Verbs. Auf diese Weise hat jedes Vorkommen schon einmal ein eigenes Label. Wir sind allerdings an *einem* Namen für einen ganzen Cluster interessiert. Hier helfen externe Informationsquellen weiter. Mit der Liste der für den Cluster gefundenen Verben lässt sich GermaNet[8] anfragen, um die Liste um die Verben zu reduzieren, die sich synonym verwenden lassen. Sollten am Ende dieses Vorgangs immer noch mehrere Verben übrig bleiben, so sollte der Cluster auf jeden Fall den Domänenexperten zur endgültigen Entscheidung vorgelegt werden, um zu einem einzigen Bezeichner für die Relation zu gelangen.

Labeling über Schlüsselwörter Ein anderer Weg zur Bestimmung der Clusternamen verwendet Schlüsselwörter, die im Cluster vorkommen. Dazu gilt es, die wichtigsten Wörter aus dem Cluster herauszufinden, also die, die den Cluster am besten beschreiben. Dazu werden üblicherweise Nomen verwendet, dieses Verfahren setzt also nicht auf die Aktion zur Beschreibung der Relation, sondern auf die Akteure.

Die Suche nach den Schlüsselwörtern gestaltet sich recht einfach: Zu den Relationsmu-

[8]WordNet ist ursprünglich ein Projekt zur Erstellung einer lexikalischen Datenbank für das Englische, durchgeführt von der Princeton University. Mittlerweile gibt es diese Datenbanken für viele andere Sprachen, unter anderem auch für Deutsch. Die Datenbank heißt GermaNet, verantwortlich zeichnet die Universität Tübingen, siehe `http://www.sfs.uni-tuebingen.de/lsd/`

stern sind durch den Clusteringschritt bereits die Wortvektoren mit ihren tf*idf-Gewichten bekannt, es müssen also lediglich die n Nomen mit den höchsten Gewichten ausgewählt werden. Deren Sequenz beschreibt dann den Cluster über das thematische Umfeld, das durch sie aufgespannt wird.

Kontrollschnittstelle für Domänenexperten

Den im vorigen Abschnitt vorgestellten Verfahren zur Auswertung der Ergebnisse der Relationserkennung ist gemein, dass eine abschließende Sichtung durch Domänenexperten zumindest wünschenswert ist. Dazu wird eine Schnittstelle benötigt, die den Experten die Steuerung des kompletten Arbeitsablaufs ermöglicht, angefangen von der Auswahl der Dokumente, über die Bestimmung von Assoziationsregeln bis hin zum Clustering. Da die einzelnen Schritte parametrisiert sind, muss das Experimentieren mit verschiedenen Parametersätzen unterstützt werden.

Bei der Entwicklung einer solchen Schnittstelle muss den folgenden Punkten besonderes Augenmerk gegeben werden:

Nutzerfreundlichkeit Die avisierten Nutzer des Systems sind Domänenexperten, d.h. üblicherweise keine Informatiker oder Computerlinguisten. Das bedeutet, dass die Steuerung der einzelnen Schritte möglichst einfach von der Hand gehen sollte. Konkret sollte auf die Voreinstellung sinnvoller Standardwerte geachtet werden sowie darauf, dass für Nutzer leicht erkennbar sein muss, in welchem Stadium der Prozesskette sie sich gerade befinden und welche Parameter für den nächsten Schritt notwendig sind.

Teilmengenverarbeitung Bei Dokumentensammlungen mit großen Relationsmengen ist es nicht wahrscheinlich, dass alle Relationen in einer Sitzung bestimmt werden können. Daher muss es möglich sein, Zwischenergebnisse zu sichern, ehe abschließend Relationsdaten für die weitere Verarbeitung im System erzeugt werden. Diese Art der Arbeitsorganisation wird durch die initiale Ermittlung von Assoziationsregeln gefördert, da diese die Gesamtarbeit in kleinere Pakete aufteilen.

Korrekturmöglichkeiten Die Schnittstelle dient sowohl zur Überprüfung der gefundenen Cluster als auch der Bearbeitung der evtl. automatisch erzeugten Bezeichnungen für diese. Daher sind Korrekturmöglichkeiten auf verschiedenen Ebenen nötig: Korrektur der Zusammensetzung von Clustern durch Löschen, Zusammenfügen oder Trennen und die Korrektur der Benennung von Clustern.

Lokal Arbeiten Die Durchführung bzw. Überprüfung der Relationserkennung ist ein Prozess, der sowohl in direkter Verbindung mit dem Server, als auch losgelöst auf einem lokalen Rechner durchgeführt werden kann. Dies ist gegenüber einem rein online arbeitenden Verfahren sogar zu bevorzugen, da so die Experten daran arbeiten können,

ohne konstant eine Internetverbindung aufrecht erhalten zu müssen. Der anfängliche Datenabruf sowie die Speicherung von Ergebnissen sind die einzigen Gelegenheiten, zu denen eine Verbindung mit dem Server wirklich unabdingbar ist.

Durch die Berücksichtigung dieser Punkte wird eine Schnittstelle geschaffen, die es Domänenexperten ermöglicht, selbständig eine Relationserkennung für ihre Dokumentensammlung durchzuführen.

Klassifikation der Relationsmenge

Die vorherigen Schritte im Arbeitsablauf dienten zur Bestimmung der Menge der Relationen, die im semantischen Netz des Datenbestands verwendet werden sollen. damit ist der erste Schritt innerhalb der Relationserkennung abgeschlossen. Was bleibt, ist die Klassifikation der Inhalte der übrigen Dokumente im Korpus mittels Relationsmenge. Diese Dokumente sind durch die Eigennamenerkennung bereits automatisch annotiert worden, d.h. die Informationen über in den Dokumenten enthaltene Entitäten liegen vor.

Konzeptuell geht es bei der Klassifikation darum, die Dokumente in Sinnabschnitte zu zerlegen, die dann einzeln auf das Vorhandensein von Relationen überprüft werden müssen, die in der Relationsmenge vorkommen. Dabei muss potenziell eine große Menge von Sätzen verarbeitet werden, die nicht zum Profil der Relationsmenge passen.

Der Arbeitsablauf zur Relationserkennung offenbart jedoch Möglichkeiten, mit denen eine Filterung der Daten vorgenommen werden kann: Jede Relation aus der Relationsmenge gehört zu genau einer Assoziationsregel. Wenn man nun für die Menge der zu klassifizierenden Sätze ebenfalls eine Assoziationsregelbestimmung durchführt, kann man sich für die weitere Verarbeitung auf die Sätze beschränken, die zu einer Assoziationsregel gehören, zu der es auch Relationen in der Vergleichsmenge gibt. Dadurch lässt sich die Menge der zu untersuchenden Sätze effizient einschränken, gleichzeitig wird der Prozess der Klassifikation vereinfacht, da nun nicht mehr jeder Satz mit jeder Relation verglichen werden muss, sondern nur noch mit denen, die zu der aktuell bearbeiteten Assoziationsregel gehören.

Das Vorgehen bei der eigentlichen Klassifikation ist aufgrund dieser Vorarbeiten recht einfach: Es wird nach Assoziationsregeln vorgegangen. Die Vorkommen für die jeweilige Regel werden geclustert. Die dabei entstehenden Cluster werden nacheinander mit den Relationen verglichen, die zu der Regel gehören. Die Cluster werden der Relation zugeschlagen, zu der sie die geringste Distanz aufweisen, sofern der Abstand einem gewissen Schwellwert genügt. So wird verhindert, dass jeder Cluster unabhängig von der Distanz irgendeiner Relation zugeordnet wird.

5.2.7 Netzerstellung

In den vorangegangenen Abschnitten wurden zuerst die Entitätsklassen bestimmt und nach Training der automatischen Erkenner auch im Korpus annotiert. Anschließend wurden die Relationen identifiziert, die in das semantische Netz aufgenommen werden sollen und sodann die Inhalte des Korpus nach diesen Relationen durchklassifiziert. Der jetzt folgende Schritt dient dazu, ein semantisches Netz aufzuspannen, das diese Entitäten und ihre Relationen enthält. Dazu muss das Wissen um Entitätsklassen, ihre Instanzen und die Relationen in eine Ontologie übertragen werden. Ontologien werden im Semantic Web mit zwei Sprachen definiert: OWL und RDF/RDFS. OWL, für Web Ontology Language[9], bietet weitreichende Möglichkeiten, eine Ontologie genau zu spezifizieren. Aus eben diesem Grund ist sie für einen Einsatz in einem weitestgehend automatisierten System nicht so gut geeignet. Daher werden die Ontologien in diesem System direkt in RDF/RDFS formuliert. Damit sind zwar weniger Einschränkungen auf den Daten möglich, dafür lässt sie sich aber auch besser automatisiert erzeugen. Zu RDF/RDFS gibt es drei wichtige Fakten, die man sich an dieser Stelle noch einmal vor Augen führen sollte[10]:

1. RDFS erlaubt die Definition von Entitätsklassen
2. RDFS erlaubt die Festlegung von Relationstypen mit festgelegten Definitions- und Wertemengen
3. RDF (und damit auch RDFS) kennt nur binäre Relationen

Diese Fakten sind für die automatische Übersetzung des Wissens wesentlich. Der erste Fakt erlaubt es, die Entitätsklassen direkt als Klassen innerhalb von RDF zu definieren, so dass die Instanzen direkt über die in die Sprache eingebauten Mechanismen ihren Klassen zugeordnet werden können. Der zweite Fakt erlaubt die klare Definition von Relationen als Abbildung zwischen zwei festgelegten Mengen. Relationen, in RDF *Properties* genannt, sind als gerichtete Verbindungen definiert, d.h. symmetrische Abbildungen müssen in beide Richtungen definiert werden. Da allerdings RDF nur binäre Relationen kennt, sind für den Fall der Relationen mit mehr als zwei Beteiligten besondere Vorkehrungen zu treffen[11].

Beim Design von RDF wurde davon ausgegangen, dass binäre Relationen zum Ausdrücken von Fakten hinreichend sein würden. Damals war das Anwendungsszenario allerdings auch noch anders, denn es ging darum, eine Sprache zur Beschreibung digitaler Ressourcen zu entwickeln. Durch das Semantic Web hat sich der Anwendungsfokus von RDF allerdings deutlich erweitert, wodurch die Beschränkung auf binäre Relationen tatsächlich eine Einschränkung geworden ist, da sich viele real vorkommende Beziehungen nicht ohne

[9] Mehr Informationen zu OWL bietet Abschnitt 3.1.4

[10] Für eine eingehendere Behandlung der Sprache siehe Abschnitt 3.1.2

[11] Der ISO-Standard Topic Maps hätte hiermit übrigens kein Problem, dort sind 2+-stellige Relationen von vornherein vorgesehen. Er hat allerdings nie große Anwendung gefunden und spielt keine wesentliche Rolle mehr.

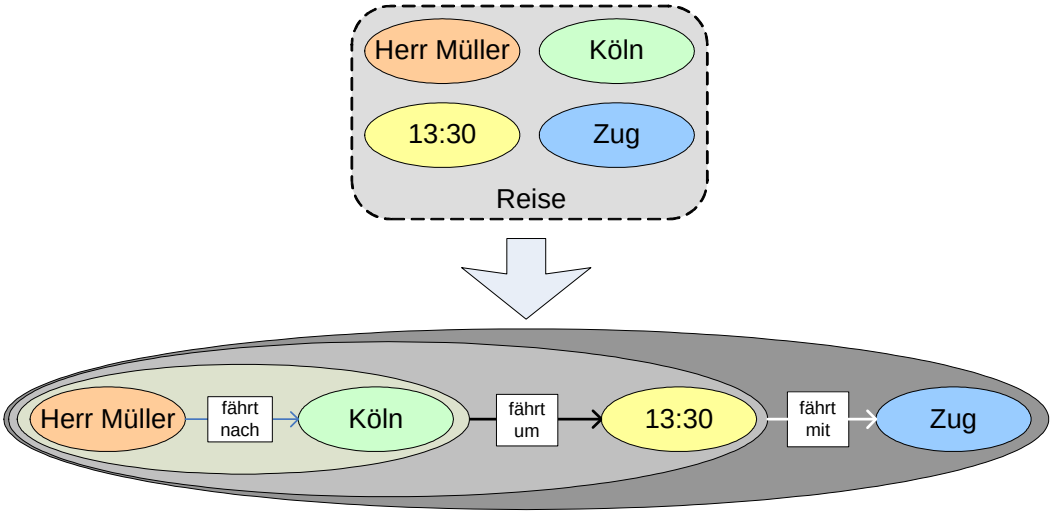


Abbildung 5.2: Auflösung des Beispielsatzes mittels Reifikation

Weiteres in binäre Beziehungen zerlegen lassen, ohne das Sinneinbußen zu verzeichnen sind. Schon die einfache Aussage " Herr Meier fährt mit dem Zug um 13:30 nach Köln" lässt sich zwar in die Tripel "Herr Meier fährt mit dem Zug" "Herr Meier fährt um 13:30" und "Herr Meier fährt nach Köln" zerlegen, allerdings geht so die Gesamtaussage verloren, da ihr Zusammenhang nicht mehr klar ist. Dem kann in RDF nur über Umwege abgeholfen werden: Da ist zunächst die Möglichkeit der Reifikation, d.h. man kann Aussagen über Aussagen treffen, indem man sie als Objekt in einem Tripel verwendet. Damit könnte man die Ausgangsaussage z.B. so kapseln wie in Abbildung 5.2 gezeigt. Was ist aber mit den anderen fünf möglichen Permutationen? Diese sind zwar unter Umständen ebenso gültige Rekonstruktionen der Aussage, die Äquivalenz wäre allerdings im Einzelfall zu überprüfen, da sie nicht zwingend sofort offensichtlich ist.

Die zweite Möglichkeit besteht darin, künstliche Knoten zu schaffen, die als Anker für binäre Relationen dienen können. In RDF gibt es das Konstrukt der so genannten anonymen Knoten, die sich für diesen Zweck einsetzen lassen. Abbildung 5.3 zeigt die RDF-Struktur, die hierbei zum Einsatz kommen könnte. Die Relation *Reise* ist hierbei zu einer Ressource geworden, die den Typ des anonymen Knotens bestimmt (der gestrichelte Pfeil steht für die Type-Beziehung). Der anonyme Knoten ist das Objekt einer *verreist*-Beziehung und ist gleichzeitig das Subjekt einer Reihe binärer Relationen, die zur näheren Beschreibung der Reise verwendet werden.

Diese zweite Möglichkeit ist bei manueller Konstruktion der ersten auf jeden Fall vorzuziehen, da sie die Komplexität des Graphen verringert und sich einfacher um weitere Fakten ergänzen lässt. Für die automatische Erstellung eines semantischen Netzes ist allerdings entscheidend, welche der beiden Möglichkeiten sich überhaupt (bzw. besser in Sachen Laufzeit und Korrektheit) für die Übersetzung einer Relation mit mehr als zwei beteiligten Entitäten verwenden lässt.

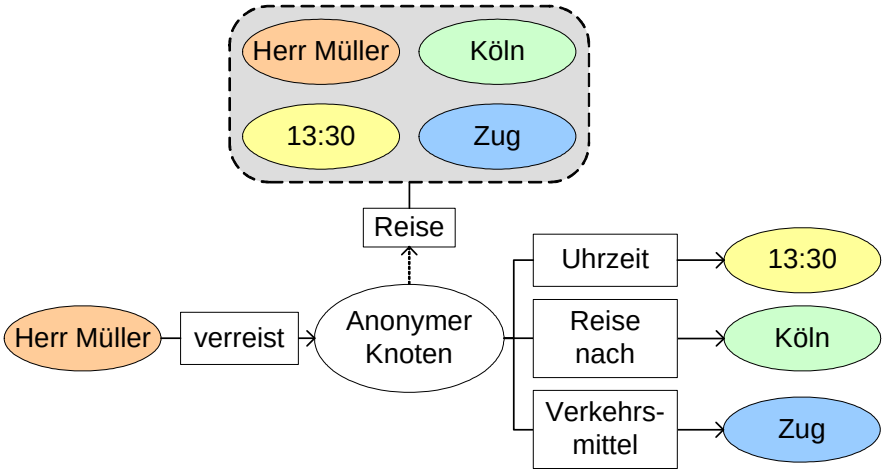


Abbildung 5.3: Auflösung des Beispielsatzes mit einem anonymen Knoten

Reifikation Die Konstruktion durch Reifikation funktioniert nur unter der Annahme gut, dass alle Permutationen von der Aussage her gleichwertig zu betrachten sind. Das vereinfachte außerdem die Übersetzung, da die Reihenfolge der Schachtelung nebensächlich wäre. Muss allerdings davon ausgegangen werden, dass sich die Permutationen in ihrer Aussage zumindest in Nuancen unterscheiden, bietet sich diese Lösung nicht an, da in diesem Fall zusätzlich zur eigentlichen Arbeit noch gelernt werden müsste, welche der Permutationen am ehesten den Wünschen der Betrachter entspricht. Das ist zwar vielleicht mit Relevance Feedback-Schleifen möglich, allerdings so subjektiv, dass es generalisiert keinen Bestand haben dürfte.

Darüber hinaus ist zu berücksichtigen, dass Reifikation in diversen Ontologien und darauf basierenden Netzen dafür genutzt wird, Aussagen nicht gesicherter Wahrheit zu kennzeichnen. Wenn also eine Quelle einen Fakt behauptet, dessen Wahrheitsgehalt unbekannt ist, so wird die Quelle des Fakts oft über Reifikation angegeben. Ist der Fakt zum Beispiel *X ist Y, behauptet von Quelle Z*, so wird dies als *Z behauptet (X ist Y)* notiert. Wählt man diesen Weg zur Auflösung n-ärer Relationen sind Missverständnisse mit denjenigen vorprogrammiert, die Reifikation zur Darstellung zweifelhafter Provenienz nutzen.

Hilfskonstrukte Die Modellierung von Relationen mit mehr als zwei Beteiligten über Hilfskonstrukte erscheint menschlichen Betrachtern spontan als der sinnvollere Weg, die Relation auszudrücken. Hier wird die Aussage genommen und auf ihre zentrale Aussage reduziert. In unserem Beispiel ist das: Herr Meier macht eine Reise. Über diese spezielle Reise kann man sprechen, sie kann an binären Relationen teilnehmen und über diese Relationen werden weitere Fakten zur Reise des Herrn Meier bekannt. Sollte darüber hinaus noch Herr Müller im gleichen Zug gewesen sein, dann kann das einfach durch das Einfügen einer einfachen Kante zwischen Herrn Müller und der speziellen, bereits ausgezeichneten Reise ausgedrückt werden. Soll zusätzlich ausgedrückt werden, dass Herr Müller und Herr

Meier *gemeinsam* diese Reise gemacht haben, sie also nicht nur zufällig im gleichen Zug gesessen haben, nun, dann wird als Hilfskonstrukt eine *gemeinsame Reise* eingeführt, die als binäre Verbindungen die Reisenden und die Reiseinformationen aufnimmt. Bei der maschinellen Verarbeitung ist jedoch die Frage zu klären, inwieweit es möglich ist, die passenden Hilfskonstrukte zu finden, die eine einfache Dekomposition der n-ären ($n > 2$) Relation in binäre Relationen ermöglicht.

Die Vorteile der zweiten Methode betreffend der Gestalt des resultierenden Netzes und der Erweiterungsmöglichkeiten wiegen deutlich schwerer als der Vorteil der ersten Methode, die sich, unter der Annahme der Gleichwertigkeit aller Permutationen, einfacher anwenden lässt. Daher wird in dieser Arbeit der zweite Weg verfolgt.

Auflösung n-ärer Relationen

Das Auflösung der n-ären Relationen für die Übertragung in die Ontologie in eine Sammlung binärer Relationen erfordert die Erfassung einiger zusätzlicher Angaben. Insbesondere muss für jede Relation geklärt werden, welche der beteiligten Entitätsklassen den Kopf der Relation darstellt. Diese Daten können über die Kontrollschnittstelle mit erfasst werden, die von den Experten zur Begutachtung der Relationen genutzt wird. Mit dieser Information kann dann jede Relation zerlegt werden.

Dabei wird analog zur Abbildung 5.3 vorgegangen. Der Kopf der Relation ist ein Knoten im semantischen Netz, der über einen URI identifiziert werden kann. Er erhält eine Relation mit einem anonymen Knoten (auf englisch „blank node“), der als Hilfskonstrukt dient. Die Relation wird die mit dem Label der Relation versehen. An das Hilfskonstrukt werden die übrigen an der Relation beteiligten Entitäten mit binären Relationen angeschlossen. Die Namen dieser Relationen leiten sich aus den Namen der Entitätsklassen ab, z.B. zu_Datum oder als_Rolle. Diese Bezeichner können in der Kontrollschnittstelle global festgelegt werden.

Anonyme Knoten sind nicht über einen URI zugreifbar, sie erhalten eine ID, deren Eindeutigkeit nur für das betreffende Netz garantiert wird. Sie eignen sich damit nicht für die Verwendung über die Grenzen einer Ontologie hinaus. Trotzdem stellen sie die einzige sinnvolle Möglichkeit dar, komplexe Relationen in RDF auszudrücken. Wenn sich abzeichnet, dass verschiedene Ontologien auf die gleichen Konzepte zugreifen wollen, die in einer der Ontologien über anonyme Knoten modelliert worden sind, so ist eine Remodellierung der betreffenden Ontologie deutlich angeraten. Die Erforschung von Mechanismen und Algorithmen für diese Aufgabe liegt jedoch außerhalb dieser Arbeit und wird deshalb nur notiert, nicht aber weiter verfolgt.

Nr	Anforderung
A7	Erweiterbare Architektur
A8	Performanz des Zugriffs auf das Netz

Tabelle 5.2: Die Anforderungen aus dem Bereich Infrastruktur

5.3 Infrastruktur

In Kapitel 2.3.2 sind die Anforderungen an die Infrastruktur eines Systems definiert worden, das die gesetzten Ziele erfüllen soll. Zur besseren Erinnerung sind diese Anforderungen in Tabelle 5.2 noch einmal aufgeführt. In diesem Unterkapitel werden die Ansätze beschrieben, die zur Erfüllung dieser Anforderungen erforderlich sind.

5.3.1 Erweiterbare Architektur

In Kapitel 2.2.2 sind bereits verschiedene Möglichkeiten für die Architektur eines Systems wie des geplanten gegenüber gestellt worden: Klassische Client/Server-Architekturen auf der einen, Service-orientierte Architekturen auf der anderen Seite. Ihre Hauptunterschiede liegen in der Strukturierung des Teils des Systems, das die Anfragen der Nutzer verarbeitet.

Im Client/Server-Modell besteht der Server aus einem zusammenhängenden Software-System, die Komponenten kommunizieren direkt miteinander, der Austausch von komplexen Objektstrukturen ist kein Problem. Dafür ist es nicht ohne große Anstrengungen möglich, einzelne Teile des Systems zu verändern, da durch die enge Koppelung Auswirkungen auf andere Teile des Servers schwer vorauszusagen sind.

Bei Service-orientierte Architekturen hat man es im eigentlichen Sinne nicht mit einem Server zu tun, sondern mit einer Vielzahl von Servern, von denen jeder eine spezielle Aufgabe erfüllt, also einen Dienst oder Service leistet. Die einzelnen Services kommunizieren miteinander über ein festgelegtes Protokoll, üblicherweise über den Austausch von SOAP-Nachrichten[12]. Mit diesen Nachrichten können Methoden der Dienste aufgerufen und mit den benötigten Parametern bestückt werden. Die dazu notwendigen Informationen sind für jeden Service in einem WSDL-Dokument (Web Service Description Language [31]) abgelegt, einer XML-Sprache, in der die Schnittstellen beschrieben werden, mittels derer mit einem Web-Service kommuniziert werden kann. Die Probleme mit dem Austausch komplexer Datentypen sind in Kapitel 2.2.2 bereits angesprochen worden, für jeglichen Datenaustausch, der über Strings oder primitive Datentypen hinaus geht, ist entsprechender Zusatzaufwand notwendig. Für die Definition der dafür benötigten Transformationen gibt

[12]Ursprünglich Simple Object Access Protocol. SOAP ist eine Empfehlung des W3C. Seit Version 1.2 wird SOAP jedoch als Eigenname und nicht mehr als Akronym verwendet, bösen Zungen zufolge, da es weder *Simple*, noch zum Zugriff auf *Objects* zu verwenden sei.

es allerdings spezielle Software[13], so dass sich der Aufwand hierfür stark begrenzen lässt. Auch mit solchen Tools können weiterhin nur Strings und primitive Datentypen übertragen werden, sie erleichtern lediglich die Erstellung der benötigten Dokumente und der Schnittstellen für die Web Services.

Die Vorteile Service-orientierter Architekturen gegenüber klassischen, monolithischen Client/Server-Applikationen liegen jedoch auf der Hand: Die Komponenten in so einer Architektur sind Services, deren Schnittstellen in einer externen Dokumentation klar geregelt sind. Da es außerhalb dieser Schnittstellen keine Kommunikationsmöglichkeiten zwischen den Komponenten gibt, ist die Kapselung der internen Logik der Komponenten sichergestellt. Änderungen an der Programmlogik einer Komponente können sich höchstens über Änderungen an den Schnittstellen auf andere Bestandteile des Systems auswirken. Das passiert nicht aus Versehen oder unbeabsichtigterweise; Seiteneffekte haben also höchstens lokale Auswirkungen.

Durch die lose Koppelung der Dienste untereinander wird es außerdem wesentlich vereinfacht, neue Dienste in das System zu integrieren, bzw. Dienste gegen andere auszutauschen, die ihre Aufgabe wesentlich besser erfüllen können. Beispiele hierfür sind das Hinzufügen neuer Verarbeitungsmechanismen oder Anzeigeoptionen, bzw. der Austausch eines Analyseverfahrens gegen ein anderes. Eine interessante Anwendung der losen Koppelung ist die Integration bestehender Software, indem um diese eine Kapselungsschicht gelegt wird, die für die Anbindung der Software als Web Service verwendet werden kann. So lassen sich Bibliotheken auch in einem Web Service-Umfeld weiternutzen, die dafür ursprünglich gar nicht ausgelegt gewesen sind.

Da die Kommunikation von Web Services über internet-gestützte Protokolle abgewickelt wird, ist die räumliche Verteilung der einzelnen Services über die ganze Welt für das System transparent möglich, mit Hilfe dieser Technik ließen sich also Dienste von überall auf der Welt zu einem virtuellen Gesamtsystem zusammenbringen.

Gleichzeitig ist es allerdings nicht so, dass sich die Nutzer mit einer Vielzahl von Diensten konfrontiert sehen, wenn sie das System benutzen wollen. Ihre Schnittstelle unterscheidet sich nicht von derjenigen, die ein Client/Server-System anbieten würde.

Daher wird in dieser Arbeit der service-orientierte Ansatz verfolgt. Die dabei entstehende Architektur lässt sich wesentlich besser auf die Anforderungen verschiedener Nutzergruppen und Applikationen anpassen, als dies mit einer Client/Server-Architektur möglich wäre.

Im weiteren Verlauf dieses Abschnittes werden Dienste skizziert, die für die Bereitstellung der Funktionalitäten des geplanten Systems benötigt werden. Das betrifft insbesondere eine Implementierung der bereits in diesem Kapitel besprochenen Systeme in der Form von Web Services, allerdings auch die Bereitstellung von Funktionalitäten, die unterstützenden

[13]Zum Beispiel Apache Axis [2], das Funktionalitäten zur Erstellung einer WSDL-Datei für komplexere Datentypen enthält.

Charakter aufweisen. Die hier dargestellte Liste ist sicher nicht erschöpfend, eine Vielzahl zusätzlicher Dienste ist denkbar. Allerdings bilden die folgenden Dienste den Kern des Systems, um den herum sich andere Dienste anschließen lassen, so es die Anwendungsfälle erforderlich machen sollten.

Datenverwaltung

Dieser Dienst regelt die Datenhaltung im System. Dies betrifft sowohl die Speicherung der Originaldokumente, als auch die Speicherung der zu diesen Dokumenten angelegten Metadaten. Idealerweise werden die Daten versioniert zur Verfügung gestellt, d.h. verschiedene Änderungsstände eines Dokuments werden parallel gehalten, damit eine Nachverfolgung der Änderungshistorie ermöglicht werden kann. In diesen Dienst integriert ist die Bereitstellung der verschiedenen akzeptierten Dokumentenformate (siehe Abschnitt 5.2.4).

Dieser Dienst kapselt alle mit der Datenhaltung verbundenen Implementierungsfragen, z.B. Auswahl der passenden Datenbank, Auswahl der Serialisierungs-Software, Zugriff auf Speichermedien, vor dem Rest des Systems. Andere Dienste müssen sich nicht mehr damit befassen, wo die Daten liegen und wie darauf zugegriffen werden kann, sie verwenden einfach die Schnittstelle, die für den Datenzugriff zur Verfügung gestellt werden.

Verwaltung des semantischen Netzes

Dieser Dienst kapselt die Datenhaltung für das semantische Netz. Die verwendete Modellierungssprache, die Art der Serialisierung der semantischen Daten, Mechanismen zum Zugriff, all das sind Details, die andere Dienste gar nicht wissen müssen. Dieser Dienst stellt eine einheitliche Schnittstelle zur Verfügung, mit der die anderen Dienste auf das semantische Netz zugreifen, bzw. Daten für das Netz zuliefern können.

Eigennamenerkennung

Dieser Dienst stellt die Funktionalitäten zur automatischen Eigennamenerkennung zur Verfügung. Seine Eingaben sind die von den Experten festgelegten Konzeptklassen mit den sie beschreibenden Beispielen. Der Dienst greift auf den Dienst zur Datenspeicherung zu, um Dokumente zur Annotation zu laden und die daraus generierten Metadaten wieder abzulegen. Für die Integration bestehender Ansätze für die Eigennamenerkennung, etwa aus GATE oder UIMA[14], lässt sich der Weg verfolgen, diese Software-Pakete über einen sog. Wrapper als Web Services nutzbar zu machen.

[14]UIMA steht für Unstructured Information Management Architecture. Es handelt sich dabei um ein IBM-internes Projekt zum Entwickeln einer generischen Architektur für Information Extraction, das seit Oktober 2006 als Open Source zur Verfügung steht und bei der Apache Foundation gehostet wird [106].

Relationserkennung

Dieser Dienst stellt die Funktionalitäten rund um die Relationserkennung zur Verfügung. Dazu gehört die Bestimmung der Assoziationsregeln genauso wie die Durchführung der Clusteringoperationen. Die Eingaben sind Schlüssel, mit denen sich die annotierten Metadatendokumente aus dem Datenspeicher beziehen lassen, sowie Parameter für die verschiedenen Algorithmen, die zum Einsatz kommen. Die Ausgaben des Algorithmus lassen sich im System als zusätzliche Metadaten ablegen, damit Experten diese nachbearbeiten und evaluieren können.

Relationsklassifikation

Dieser Dienst verwendet die Daten der Eigennamenerkennung und die abgelegten Relationen, um damit den restlichen Datenbestand zu klassifizieren. Die Ausgaben dieses Dienstes werden ebenfalls als Metadaten im System abgelegt.

Netzerstellung

Dieser Dienst wird zur Erstellung des semantischen Netzes aus den Daten der Eigennamenerkennung und der Relationsklassifikation verwendet. Die zu wandelnden Daten und Parameter, die für die gewählte Modellierungssprache benötigt werden, dienen als Eingabe in diesen Dienst. Die Ausgabe ist eine Ontologie, die alle eingegebenen Daten enthält. Dieses wird an den Verwaltungsdienst für das semantische Netz weitergegeben. Insofern kommt diesem Dienst eine Sonderstellung zu, da er Daten in der Modellierungssprache selbst erzeugt. Allerdings ist zur Wahrung der Flexibilität dieser Schritt notwendig. Das Update der Systemontologie geschieht erst innerhalb des Verwaltungsdienstes, das vom Netzerstellungsdienst erzeugte Netz ist eine lokale Kopie.

Semantische Suche

Dieser Dienst wird verwendet, um Suchanfragen an das semantische Netz zu stellen. Das könnte zwar auch über den Verwaltungsdienst geschehen, unter Berücksichtigung der zweiten Anforderung aus dem Bereich Infrastruktur empfiehlt es sich jedoch, die Bearbeitung von Suchanfragen von einem eigenen Dienst aus zu bedienen, da so auch speziellere Vorverarbeitungen auf dem semantischen Netz durchgeführt werden können.

Datenimport

Dieser Dienst ist ein Beispiel für einen Hilfsdienst. Er wird eingesetzt, um Dokumente in das System zu importieren. Er dient als Anknüpfungspunkt für lokale Applikationen, die es

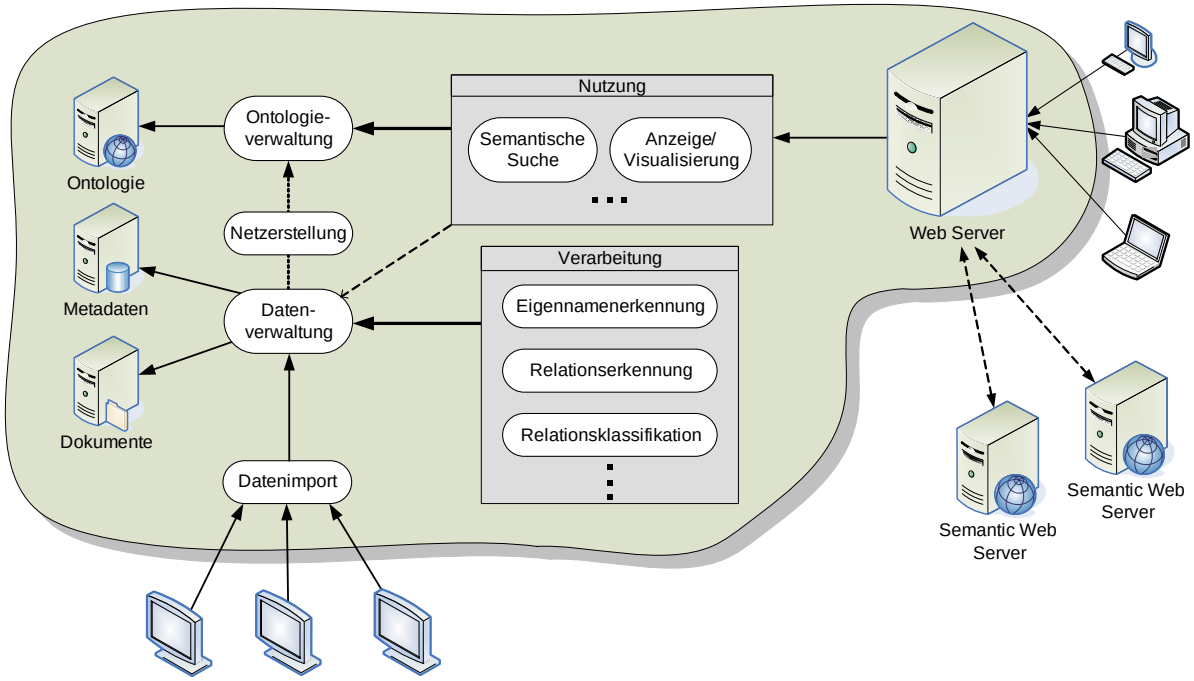


Abbildung 5.4: Zusammenspiel der Dienste im geplanten System

erlauben, Dokumente aus einem Verzeichnis eines Rechners oder auch von einer Webseite in das System zu übertragen.

Authentifizierung

Dieser Dienst ist ebenfalls ein Beispiel für einen Hilfsdienst, der für ein nichttriviales System notwendig ist. Er regelt den Zugriff auf einzelne Teilbereiche des Systems, insbesondere der Daten in den verschiedenen Repositorien, kann aber auch dazu genutzt werden, um die Anmeldung von Nutzern an der jeweiligen Nutzerschnittstelle zu überprüfen.

Zusammenspiel der Dienste

Abbildung 5.4 zeigt das Zusammenspiel der einzelnen Dienste im geplanten System. Integriert ist ein Web Server, über den die Zugriffe auf das System gesteuert werden können. Daten werden über den Datenimport-Dienst in das System geladen, wo sie von den verschiedenen Verarbeitungsdiensten aufbereitet werden, ehe daraus ein semantisches Netz gewoben wird.

Innerhalb der Dienste lassen sich zwei Gruppen identifizieren, die in der Abbildung in den Kästen *Verarbeitung* und *Nutzung* zusammengefasst sind. Zu beiden Gruppen ist eine Vielzahl zusätzlicher Dienste vorstellbar, die ebenfalls Teil eines solchen Systems werden

könnten. Dies ist in der Abbildung durch Fortsetzungspunkte angedeutet. Besonders die Analyse zusätzlicher Datenformate und die Einbindung von Mehrwertdiensten zum Angebot innerhalb des Web Servers sind denkbare Erweiterungen.

Der Web Server bildet in dem dargestellten Szenario die Schnittstelle nach außen. Er greift zur Anzeige der Daten auf die Nutzungs-Dienste zu, die außerdem Möglichkeiten zur semantischen Suche bereitstellen. Andere Systeme sind denkbar, in denen zum Desktop-Applikationen direkt mit dem Server Kontakt aufnehmen, etwa wenn die Verwaltung eines verteilten Dokumentenservers für eine Einrichtung über ein solches System geregelt wird.

Außer den normalen Zugriffen über Web Browser ist in der Abbildung auch die Möglichkeit gezeigt, das Angebot mit anderen Semantic Web-Anwendungen im Netz zu verknüpfen, wenn thematische Berührungspunkte dies angeraten erscheinen lassen. Dazu greifen die anderen Server über definierte URL über den Web Server auf die Datenbasis des Systems zu.

5.3.2 Performanz der semantischen Suche

Volltextsuchmaschinen haben, trotz aller ihrer in dieser Arbeit beschriebenen Nachteile, einen entscheidenden Vorteil: Sie sind schnell. In die Erforschung effizienter Textindexierungsmechanismen ist in den letzten Jahrzehnten viel Forschungsarbeit investiert worden, ebenso in die Skalierbarkeit der Suche auf Datenmengen, wie sie bei der Suche auf großen Teilen des Internets anfallen. Volltextsuche lässt sich sehr gut parallelisieren, weswegen Suchmaschinen wie die von Google oder Yahoo typischerweise Antwortzeiten im höchstens einstelligen Sekundenbereich, meist sogar im Subsekundenbereich, aufweisen. Dafür sorgen ganze Serverfarmen, weshalb einzelne Server nur noch kleine Teile des Datenbestands zu durchsuchen haben. Das größte Problem ist die Sicherstellung der Qualität der Antworten, die Gewichte der Rankingalgorithmen gehören zu den am besten gehüteten Geheimnissen der großen Anbieter.

Damit erhöhen sich die Barrieren für neue Ansätze beträchtlich, denn neben der Qualität ist die Geschwindigkeit der Anfragenbearbeitung ebenfalls ein sehr wichtiger Faktor. Dieser Herausforderung muss sich auch ein System stellen, das für sich in Anspruch nimmt, deutlich bessere Ergebnisse für eine Domäne zu liefern, als das mit aktuellen Suchmaschinen möglich wäre. Dauert diese Suche erheblich länger, so hat die neue Technik einen sehr schweren Stand, denn Internetnutzer hassen kaum etwas mehr, als auf Ergebnisse einer Suchanfrage zu warten.

Semantische Suche ist von der Grundidee deutlich komplexer als eine Volltextsuche, da die Suche auf Ontologie eine Suche in einem gerichteten Graphen ist, wobei sowohl verschiedene Knoten- als auch Kantentypen vorliegen können. Für diese Aufgabe hat das W3C die Anfragesprache SPARQL entwickelt[15] [89]. Das W3C hat allerdings nur die Sprache spezifiziert, die Implementierung durch Dritte wird über die frei verfügbare Spezifikation

[15]Siehe Abschnitt 3.1.5 für einen Überblick über SPARQL.

ermöglicht. Dazu stellt das W3C eine Testsuite zur Verfügung, mit der Implementierungen auf ihre Konformität zur Spezifikation getestet werden können [110]. Testergebnisse für 15 verschiedene Implementierungen von SPARQL liegen vor, davon bestehen allerdings nur drei alle Testfälle komplett[16].

Der nachfolgende Abschnitt geht näher auf eine dieser drei Implementierungen ein.

SPARQL, ARQ und Jena

ARQ ist der Name der SPARQL-Umsetzung von den HP Labs. Dieses Software-Paket ist in Java geschrieben und erweitert die ebenfalls von HP erstellte Bibliothek Jena, eine RDF/OWL-Engine, um eine SPARQL-Suche. ARQ ist ebenfalls in Joseki enthalten; dabei handelt es sich um einen dedizierten SPARQL-Server, der ebenfalls zum Jena-Projekt gehört. Die Semantic Web Aktivitäten der HP Labs stehen als Open Source zur Verfügung, weswegen sie weite Verbreitung innerhalb der Semantic Web Community gefunden haben.

Jena ermöglicht die Serialisierung von Ontologien in relationale Datenbanken, die so nicht komplett im Speicher gehalten werden müssen, um damit arbeiten zu können. Das dafür benötigte Datenbankschema wird von Jena selbst erzeugt, standardmäßig wird das Schema namens RDB verwendet. Dessen Antwortzeiten sind mit einem Benchmark für die Messung der Anfrageperformanz getestet worden, dem Lehigh University Benchmark[17] (LUBM) [55, 56]. Dieser Benchmark erzeugt auf der Basis einer Universitäts-Ontologie synthetische Daten zu Professoren, Mitarbeitern und Studenten an verschiedenen Hochschul-Lehrstühlen, sowie Daten über Rollen, Kurse und Publikationen, die mit den unterschiedlichen Personen verknüpft sind. Die Anzahl der zu erzeugenden Hochschulen kann als Parameter gesetzt werden.

Die Tests für ARQ wurden mit 15 virtuellen Hochschulen durchgeführt, die Ontologie enthielt (nach Inferenz) 19,5 Millionen Tripel und war damit deutlich zu groß für ein komplettes Einlesen in den Hauptspeicher. Zwei Fragen aus dem Benchmark wurden zur Evaluierung ausgewählt: Zunächst eine Suche nach Studierenden, die einen bestimmten Kurs an einer bestimmten Universität belegt haben und danach eine Auflistung aller Diplomanden mit den Lehrstühlen und Universitäten, an denen sie ihr Vordiplom gemacht haben. Die erste Anfrage benötigt auf einem typischen Server gut 24 Sekunden, die zweite 232 Sekunden, also knapp vier Minuten! Selbst Tools, die sich einer weiten Verbreitung erfreuen, sind also für einen interaktiven Suchprozess nicht wirklich gut geeignet.

Die Lösung liegt in der Optimierung der Datenbankstruktur. Die Jena-Suite bietet seit kurzem genau das in der Form des SDB-Datenbankschemas für Jena an. Dieses ist speziell auf die Nutzung in SPARQL ausgerichtet. Wird dieses Schema verwendet, so reduzieren sich die Antwortzeiten drastisch: Für die erste Frage wird nicht mal eine Zehntel Sekunde

[16] Stand Juni 2008

[17] Siehe `http://swat.cse.lehigh.edu/projects/lubm/`

benötigt, die zweite Anfrage lässt sich in 3,4 Sekunden beantworten. In beiden Fällen ist SDB also zwei Größenordnungen schneller als die Standardimplementierung mit RDB. Mit diesen Antwortzeiten werden semantische Suchen zumindest konkurrenzfähig gegenüber Volltextsuchen.

Schlussfolgerung für das geplante System

Der vorige Abschnitt hat aufgezeigt, wie wichtig die Optimierung der Suchprozesse für semantische Suchsysteme ist. Die Antwortgeschwindigkeit der Standard-Implementierung eines der am weitesten verbreiteten Systeme in der Community lässt sich durch eine gezielte Optimierung um Zehnerpotenzen beschleunigen. Daher ist es angeraten, die semantische Suche innerhalb des Systems in einem eigenen Suchserver anzusiedeln, der über diese optimierte Variante verfügt. Bei der Konzeption einer größeren Installation des Systems ist es wünschenswert, den Such-Server auf einer eigenen Maschine betreiben zu können. In diesem Fall ist dafür Sorge zu tragen, dass die Daten aus dem systeminternen Speicher mit dieser gesonderten Maschine regelmäßig synchronisiert werden.

5.3.3 Performanz der internen Algorithmen

Die Geschwindigkeit der intern ablaufenden Algorithmen zur Erstellung des semantischen Netzes spielt lange keine so große Rolle für den täglichen Einsatz, wie dies für die Geschwindigkeit der interaktiv ablaufenden Prozesse gilt, allerdings muss auch bei diesen, im Backend des Systems angesiedelten, Prozessen darauf geachtet werden, dass sie effizient und schnell ablaufen.

Dies betrifft insbesondere die kontinuierliche Aufgabe des Analysierens der Änderungen in dynamischen Datenbeständen, ob sich Änderungen ergeben haben, die Auswirkungen auf die Struktur der Ontologie haben könnten. Je nach Nutzerkreis eines solchen Angebots kann es sich um eine große Anzahl von Änderungen handeln, die derart analysiert werden müssen. Hier sind sowohl Fragen der dafür benötigten Zeit als auch des für die Analyse benötigten Speicherplatzes von Belang.

Zu betrachten sind die verarbeitungsintensiven Blöcke des Arbeitsablaufs: Die Eigennamenerkennung, die Relationserkennung, die Relationsklassifikation und die anschließende Transformation der Ergebnisse in das semantische Netz.

Eigennamenerkennung

Da dieser Block des Workflows mit Softwaremodulen bestritten wird, die extern entwickelt worden sind, können etwaige Performanzengpässe nur über externe Maßnahmen abgefedert werden. Das ist im Wesentlichen die Verteilung der Arbeiten auf mehr Prozessoren, die je-

weils einen Teil der Arbeit abwickeln. Sofern das Modell für die Erkennung der einzelnen Konzeptklassen bereits erstellt ist, lässt sich eine derartige Verteilung der Arbeiten gut realisieren. Insbesondere für Angebote, die auf großen, dynamischen Dokumentensammlungen beruhen, ist dieser Umstand interessant.

Relationserkennung

Dieser Block des Workflows stellt wahrscheinlich die härtesten Anforderungen an die Performanz der verwendeten Algorithmen; schließlich sind diese dafür vorgesehen, auf Arbeitsplatzrechnern in Desktop-Applikationen abzulaufen. Dies erfordert Algorithmen, deren Laufzeit auf einigermaßen aktuellen Arbeitsplatzrechnern mit der zu erwartenden kleinen Menge an Trainingsdokumenten höchstens im einstelligen Minutenbereich - möglichst im unteren! - liegen sollten.

Die Assoziationsregelbestimmung ist der einzige Prozess innerhalb des Workflows, bei dem tatsächlich alle Trainingsdaten verarbeitet werden müssen. Angesichts der überschaubaren Menge an Dokumenten und Entitätsklassen sollte die Laufzeit jedoch solide in dem geforderten Bereich liegen. Sind die Regeln erst einmal bestimmt, so führt die Auswahl einzelner Regeln dazu, dass die anschließende Clusterung der Vorkommen auf einem drastisch kleineren Datenbestand vorgenommen werden muss, insofern ist auch hier die Laufzeit nicht als problematisch einzuordnen. Ebenso verhält es sich mit dem Speicherbedarf der Applikation: Aufgrund der kleinen Trainingsmenge von Dokumenten sollten sich hier für einen durchschnittlichen PC keine Probleme ergeben.

Relationsklassifikation

Im Verlauf der Relationsklassifikation sind alle Dokumente zu verarbeiten, deren Metadaten sich durch die vorhergehenden Schritte des Workflows geändert haben. Der Extremfall hier ist die Erstverarbeitung aller zur Verfügung stehender Dokumente. Das größte Performanzhindernis dieses Verarbeitungsprozesses ist der benötigte Speicherplatz. Im ersten Schritt, der Assoziationsregelbestimmung, sind alle Sätze zu berücksichtigen, die mindestens zwei Entitäten enthalten. Durch die für die weitere Verarbeitung benötigten Klassenstrukturen kann es hier leicht zu Speicherproblemen kommen. Im darauf folgenden Clusteringschritt fallen ebenfalls zusätzliche Daten an, nämlich für die Bestimmung der Wortvektoren.

Die Laufzeit der Algorithmen ist bei diesen Arbeitsschritten praktisch zu vernachlässigen. Die Laufzeit der Assoziationsregelbestimmung ist stark abhängig von der Anzahl der betrachteten Gegenstände. Bei einer großen Anzahl **n** von Gegenständen ist die Bestimmung der jeweiligen Itemsets bis hinauf zum (n-1)-elementigen Itemset sehr aufwändig. Im vorliegenden Szenario ist die zu erwartende Anzahl an Konzeptklassen allerdings klein, typischerweise ist mit nicht mehr als 20 verschiedenen Klassen zu rechnen. Das Zusammenstellen der vorliegenden Sätze ist der Vorgang, der hier am meisten Zeit erfordert. Sobald

die Daten in eine geeignete Nachweis-Struktur gebracht worden sind, ist die Bestimmung der einzelnen Sets keine rechenintensive Angelegenheit.

Ähnliches gilt für die eigentliche Klassifikation. Da die Vektorenmenge im Vorfeld durch die sequentielle Abarbeitung der relevanten Assoziationsregeln eingeschränkt wird, ist auch die Anzahl der sich daraus ergebenden Cluster überschaubar, die gegen die Relationen klassifiziert werden müssen. Dadurch ist auch hier die Laufzeit stark beschränkt.

Die Speicheranforderungen sind jedoch beträchtlich, vor allem, wenn man es mit umfangreichen Eingangsdokumenten zu tun hat. In solchen Fällen muss damit gerechnet werden, dass nicht alle Dokumente in einem Durchgang verarbeitet werden können; die Arbeit muss also aufgeteilt werden. Die einzelnen Schritte des Klassifikationsprozesses lassen sich allerdings auch gut mit Teilmengen der Dokumente durchführen.

Bei der Assoziationsregelerstellung lassen sich Abweichungen in der Zusammenstellung der Assoziationsregeln sind zwar nicht auszuschließen, allerdings sind bei einer genügend großen Mindestanzahl von Dokumenten in den einzelnen Teilmengen daraus resultierende Verfälschungen der erzeugten Relationen auszuschließen. Der Support-Parameter wird üblicherweise sowieso recht niedrig eingestellt, da man gerade an den weniger häufigen Relationen interessiert ist. Wichtig für die Entscheidung über die Bearbeitung einer Regel ist daher im Wesentlichen die Konfidenz, denn sie zeigt die Wahrscheinlichkeit an, dass tatsächlich ein relevanter Zusammenhang zwischen den beteiligten Konzeptklassen besteht.

Das Clustering wiederum ist abhängig von den Ähnlichkeiten der frisch erzeugten Cluster zu den gesicherten Relationen, insofern hat die Menge der betrachteten Vektoren hier keine Auswirkung.

Das Ziel muss also sein, solche Pakete zu schnüren, dass der zur Verfügung stehend Speicher effizient ausgelastet wird, die einzelnen Aktionen des Verarbeitungsprozesses also ohne Rückgriff auf Auslagerungsdateien durchgeführt werden können. Dies wirkt sich deutlich auf die Performanz des gesamten Prozesses aus, da das Laden von Speicherseiten von sekundären Medien (in der Regel von Festplatte) viel Zeit in Anspruch nimmt.

Gute Werte für diese Pakete lassen sich nur empirisch zu bestimmen, da sie sowohl von der Größe der Dokumente, der Häufigkeit darin erkannter Entitäten, dem verwendeten Betriebssystem und nicht zuletzt dem zur Verfügung stehenden echten Hauptspeicher abhängt. Es ist also darauf zu achten, dass diese Größe parametrisiert wird.

Transformation

Bei der Transformation der gesammelten Entitäts- und Relationsdaten in das semantische Netz handelt es sich streng genommen lediglich um ein Umkopieren von Programmstrukturen in eine andere Repräsentation. Dieser Vorgang wird unterstützt durch Software-Tools

für die Verwaltung semantischer Netze, zum Beispiel durch Jena oder Sesame[18]. Technisch gesehen wird eine Vielzahl neuer Objekte im Speicher erzeugt, zusätzlich erzeugt die Verwaltung des semantischen Netzes ebenfalls Overhead im Speicher.

Der vorliegende Prozess ist also ebenfalls eher speicher- denn zeitkritisch. Auch hier lohnt es sich also, über eine Reduktion der Speicheranforderungen nachzudenken. Die Erzeugung des Netzes kann genauso in Arbeitspakete aufgeteilt werden wie die Klassifikation der Relationen, mit dem Vorteil, dass hierbei keine Qualitätseinbußen zu befürchten sind. Um die Menge des verfügbaren Speichers weiter zu erhöhen, empfiehlt es sich, den RDF-Graphen nicht komplett im Speicher zu halten und ihn erst am Ende des Prozesses zu serialisieren, sondern direkt auf einem Modell aufzubauen, das den Graphen in einer relationalen Datenbank hält.

Durch die Notwendigkeit von Zugriffen auf die Datenbank verlängert sich allerdings die Laufzeit des Prozesses, da die De- bzw. Serialisierung der Daten aus/in die Datenbank naturgemäß länger braucht, als wenn die Daten direkt im Speicher gehalten würden. Je nach Datenaufkommen kann es sich daher lohnen, die Daten in ein Zwischenformat zu übertragen, dass dann gesondert mit speziellen Tools, so genannten Bulk Loadern, in die Datenbank übertragen werden können. Diese verwenden einen speziellen Treiber, der erst am Ende des Imports prüfen, ob die Integritätsbedingungen innerhalb der Tabellen eingehalten werden.

Schlussfolgerungen für das geplante System

Die in diesem Abschnitt aufgeführten Algorithmen sind im Wesentlichen als Backend-Prozesse einzuordnen. Die Relationserkennung fällt aus dem Rahmen, da die Algorithmen in eine interaktive Desktopanwendung eingebettet sind, allerdings gelten für sie die gleichen Argumente, die auch für die Relationsklassifikation angeführt worden sind: Bei den verwendeten Verfahren ist eher der Speicherplatz ein Problem als die Laufzeit der Algorithmen. Daher ist es ein großer Vorteil, dass sich die Daten in Pakete aufteilen lassen, so dass Verarbeitungen nicht an der Speicherkapazität scheitern müssen.

Angesichts der späteren Nutzungsszenarien empfiehlt es sich, das Modell direkt in einer relationalen Datenbank vorzuhalten, idealerweise in einer, die für die Suche optimiert ist, auch wenn deren Performanz beim Einstellen der Daten schlechter sein sollte.

5.4 Nutzung

In diesem Abschnitt werden die Ansätze besprochen, die sich mit der tatsächlichen Nutzung des semantischen Netzes beschäftigen (siehe Abschnitt 2.2.3). Die in Kapitel 2 aufgestellten

[18] Siehe `http://www.openrdf.org`

Nr	Anforderung
A9	Ermöglichen unterschiedlicher Sichten auf die Ontologie
A10	Unterstützung dynamischer Datenbestände

Tabelle 5.3: Die Anforderungen aus dem Bereich Nutzung

Anforderungen sind zur besseren Übersicht in Tabelle 5.3 aufgeführt.

Der letzte Abschnitt endete bereits mit einem Hinweis auf die Nutzungsszenarien. Auf diese wird in den Abschnitten dieses Unterkapitels noch näher eingegangen werden. Generell wird das semantische Netz dazu verwendet, die den Dokumenten inhärenten Strukturen und Inhalte zu- und begreifbar zu machen. Dies kann auf verschiedene Arten geschehen:

- Indem Teile der Struktur in der Form inhaltlich geführter Zugänge manifestiert werden, etwa als hierarchischen Einstieg in das Datenmaterial.
- Durch das Angebot semantischer Suchmöglichkeiten auf der Ontologie, mit der Möglichkeit, spezielle Suchanfragen zusammen zu stellen.
- Mit speziellen Visualisierungen, die sich die über die Inhalte bekannten Informationen und ihre Relationen untereinander zu Nutze machen können (etwa für Zeitstrahl-Ansichten, Clusteranalysen auf Landkarten für Personen Innovationen, Verlaufsanalysen von Lebensläufen verschiedener Personen mit ihren Begegnungspunkten usw.
- Durch die Übernahme aktualisierter Daten, die sich aus Änderungen in der Dokumentenbasis ergeben
- Durch das Anbieten von Schnittstellen, über die externe Web Services auf die Ontologie zugreifen können, bzw. mit denen Informationen aus Ontologien anderer Organisationen integriert und referenziert werden können.

Die hier aufgeführten Punkte zeigen insbesondere die Vorteile, in deren Genuss *menschliche* Nutzer bei der Verwendung des geplanten Systems kommen sollen. Dies ist der Tatsache geschuldet, dass die Idee der autonom agierenden Software-Agenten, die anhand einer semantischen Beschreibung der Aufgabe eigenständig für ihre Auftraggeber z.B. Flüge oder Hotels buchen, Medikationen vorschlagen und Medikamente erwerben oder Millionengeschäfte an der Börse tätigen, im Moment noch nicht real existent sind, auch wenn genau diese Anwendung einer der ausschlaggebenden Gründe für die Entwicklung des Semantic Web gewesen sind.

Bei aller (für einen Informatiker sehr nachvollziehbaren) Begeisterung für die möglichen Anwendungen von Software-Agenten im Semantic Web bleibt anzumerken, dass darüber die unmittelbaren, heute schon umsetzbaren, Möglichkeiten zur Verbesserung der menschlichen Arbeit mit großen, u. U. dynamischen, Datenmengen oft in den Hintergrund gedrängt werden. Daher liegt der Fokus in den vorliegenden Ansätzen auf der Vereinfachung der Arbeit für Nutzer.

5.4.1 Definition von Sichten auf das Netz

Diese in A9 formulierte Anforderung bildet die Basis für alle Ansätze, die in der Applikationsschicht des Systems anzusiedeln sind. Egal, ob es sich um einen hierarchisch geführten Einstieg in die Inhalte des Systems handelt, der rein auf einer textuellen Ansicht basiert, ob verfügbare Informationen in einer Baumansicht angezeigt werden sollen oder ob Daten in einer eigens entwickelten, futuristischen Visualisierung erlebbar gemacht werden sollen, die Grundvoraussetzung für all diese Darstellungsformen ist ein Zugang zu der Datenbasis, der eine adäquate Aufbereitung der Daten erlaubt.

Benötigt wird also eine Schnittstelle, die Daten aus der Ontologie für die weitere Verarbeitung durch Anwendungen in der Applikationsschicht des Systems zur Verfügung stellt. Diese Schnittstelle muss unter zwei Gesichtspunkten anwendungsneutral ausgelegt werden: Sie darf keine tieferen Informationen über die im System enthaltene Ontologie benötigen, da diese vom Szenario abhängt, unter dem das System installiert worden ist. Darüber hinaus darf sie aber auch keine Annahmen enthalten, was die Applikationen angeht, die auf diese Schnittstelle zugreifen, da sich die Schnittstelle so zu eng an die Applikationsschicht binden würde. Das würde das Einbinden neuer Komponenten unnötig erschweren, was einen Verstoß gegen Anforderung A7 bedeutete.

Die Schnittstelle sollte daher einen einheitlichen Zugang bieten, egal welcher Art die Ontologie ist, die im jeweiligen System angelegt worden ist. Zur Realisierung eines solchen Zugangs bietet es sich an, die designierte Abfragesprache für RDF/OWL zu verwenden: SPARQL. Diese nimmt für Ontologien die gleiche Position ein, wie sie SQL für relationale Datenbanken inne hat. Die Ergebnisse einer Anfrage mit SPARQL werden in XML zurückgegeben, die Daten lassen sich also verhältnismäßig einfach in eine Form bringen, die von den jeweiligen Anwendungen weiter verarbeitet oder angezeigt werden kann. Darüber hinaus ist SPARQL nicht an eine Programmiersprache gebunden, die Schnittstelle kann also von den verschiedensten Programmiersprachen aus bedient werden. Die Funktionalitäten zur Anfragebearbeitung werden in einem Modul gebündelt, dem SPARQL-Server.

Abbildung 5.5 zeigt die Einbettung des SPARQL-Servers in die Gesamtarchitektur. Der Server ist in die Ontologieverwaltung einzuordnen, da er Zugriff auf die Serialisierung der Ontologie benötigt, womit auch dieser Teil des Systems sich in mehrere Unterdienste aufsplittet (vgl. hierzu Abbildung 5.4). Alle system-externen Anfragen zu Informationen aus der Ontologie werden von diesem Server bearbeitet, d.h. alle Dienste, die zur Kategorie *Nutzung* zählen und Daten aus der Ontologie benötigen, z.B. für die semantische Suche, für thematische Einstiege oder spezielle Visualisierungen, beziehen ihre Daten über den SPARQL-Server. Diese Dienste beschreiben die Hauptinteraktionen eines Web Servers mit dem Backend-System.

Darüber hinaus ist es aber auch möglich, externen Applikationen Zugriff auf die Daten der Ontologie zu gewähren, seien es jetzt die eingangs erwähnten Software-Agenten oder andere Semantic Web Applikationen. Diese greifen ebenfalls über den Web Server zu,

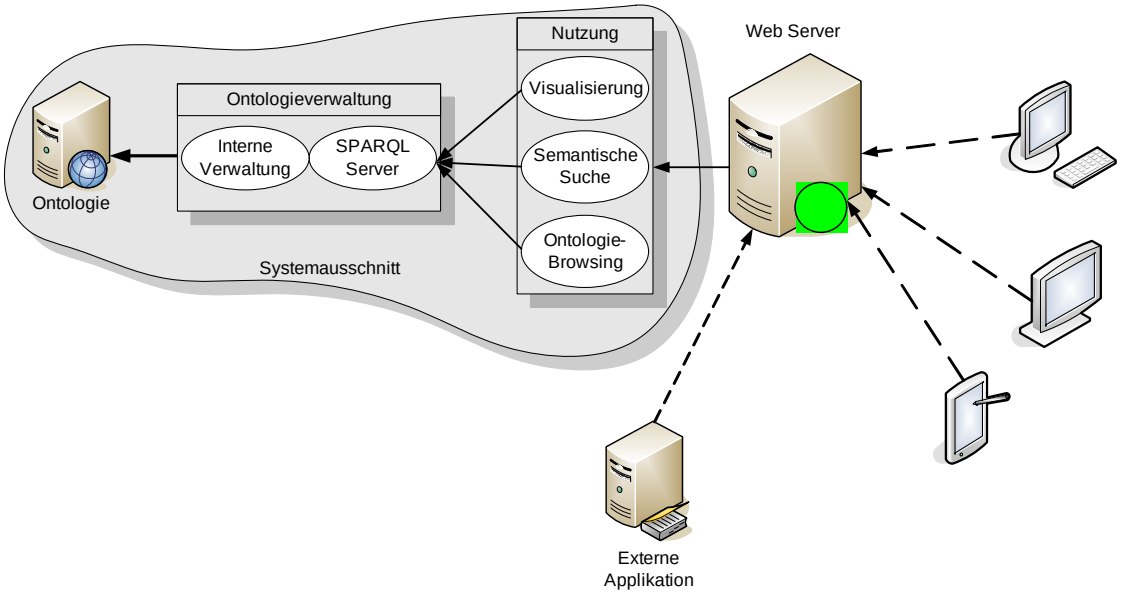


Abbildung 5.5: Einbettung des SPARQL-Servers ins System

allerdings wird eine spezielle URL verwendet, die Resultate nicht graphisch aufbereitet, sondern diese im SPARQL Response XML zurückgibt. Der Zugang über den Web Server wird gewählt, um leicht Zugangsbeschränkungen zum SPARQL-Server einrichten zu können. HTTP-Server enthalten die dafür notwendigen Module bereits.

Egal, ob zugangsbeschränkt, offen oder nur für den Gebrauch in einem Intranet, da alle Anfragen zu Daten aus der Ontologie über den SPARQL-Server gehen müssen, empfiehlt es sich, darüber nachzudenken, ihn auf einem dedizierten Server laufen zu lassen, der mindestens so gut ausgestattet ist wie der Web Server. Durch seine exponierte Stellung im Gesamtsystem wird er sonst leicht zu einem Flaschenhals für die Performanz der ganzen Applikationsschicht. Um eine hohe Geschwindigkeit der Anfragenbearbeitung sicherstellen zu können, sollte zusätzlich auf eine Serialisierung der Ontologie geachtet werden, die auf Performanz optimiert ist (siehe z.B. die Diskussion zur Auswirkung verschiedener Ontologie-Serialisierungen auf die Suchperformanz in Abschnitt 5.3.2).

5.4.2 Semantische Suche

Die semantische Suche ist sehr stark mit dem vorigen Abschnitt verbunden, denn die hierzu zu verwendende Sprache ist eben SPARQL. Gewissermaßen das SQL des Semantic Web, teilt es das Kernproblem mit dem Vorbild aus der Welt der relationalen Datenbanken: Normale Nutzer können üblicherweise ihr Informationsanliegen damit nicht formulieren. Das ist allerdings meist auch nicht erwünscht, erfordert die Formulierung doch tiefere Kenntnisse über die Modellierung, die der Datenbank (oder eben der Ontologie) zu Grunde

liegt.

Daher sind Wege zu finden, die es den Nutzern ermöglichen, Anfragen an das System zu stellen, ohne sich in SPARQL und außerdem auch noch in die Modellierung der Ontologie einarbeiten zu müssen. Die Rahmenbedingung hierbei ist, dass die Recherche-Möglichkeiten nicht zu sehr eingeschränkt werden dürfen - denn wenn diese „neue“ Art der Suche nicht besser ist als eine Volltextsuche, dann wird sie nicht genutzt werden.

Hilfsmittel zur Anfrageformulierung

Grundsätzlich lassen sich drei verschiedene Methoden zur Unterstützung von Nutzern bei der Anfrageformulierung unterscheiden:

Volltextsuchzeile Diese Suchform ist besonders aus Volltextsuchmaschinen bekannt. Die Suchbegriffe werden nacheinander eingegeben, boolesche Operatoren können verwendet werden, um die Art der Beziehung der Begriffe untereinander anzugeben. Standardmäßig wird eine Verknüpfung der Terme mit **AND** angenommen. Die mit der Anfrage ausgedrückte aussagenlogische Formel wird im Volltextindex gesucht, die Ergebnisse anhand der Anfrage und interner Metriken nach Relevanz sortiert und die Liste der passenden Dokumente zurückgegeben.

Natürlichsprachliche Formulierung Die Anfrage wird in ganzen Sätzen formuliert und das System verwendet computerlinguistische Verfahren, um die Anfrage zu verstehen und das passende Ergebnis herauszufinden.

Formularbasierte Anfragen Die Anfrage wird über ein vorgegebenes Formular zusammengestellt, indem die teilnehmenden Terme ausgewählt und optional noch parametrisiert werden. Im Kern handelt es sich hierbei um ein Vorgehen nach der query-by-example-Methode, da ein Muster zur Verfügung gestellt wird, zu dem passende Vorkommen gesucht werden.

Volltextsuchzeile Für die in dieser Arbeit schon mehrmals geäußerte Überzeugung, dass semantische Suchen größeren Mehrwert bieten können als Volltextsuchmaschinen, ist es vielleicht verwunderlich, dass ausgerechnet die Ikone der Volltextsuche als erste Möglichkeit zur Nutzerunterstützung in Betracht gezogen wird. Sie eignet sich zwar nicht, um komplexe inhaltliche Suchen anzustoßen - dafür müsste das Operatoreninventar deutlich vergrößert werden - aber als einfacher Einstieg in die Arbeit mit inhaltlich erschlossenen Datensammlungen eignet sie sich gut.

So kann sie im einfachsten Fall dazu verwendet werden, um Nutzer nach Eingabe eines Schlagworts zu einer Instanz oder einem Konzept der Ontologie führen zu können. Dazu muss die Ontologie nach Vorkommen des Suchbegriffs als Bezeichner eines Konzepts oder

einer Instanz durchsucht werden. Gibt es den Begriff nur einmal in der Ontologie, so können direkt die zugehörigen Informationen angezeigt werden.

Gibt es den Begriff aber an mehreren Stellen, so kann man eine Disambiguierung durchführen, indem die verschiedenen gefundenen Bedeutungen nebeneinander gestellt werden. Ist der Begriff hingegen nicht in der Ontologie enthalten, dann sollte die Suche an eine konventionelle Suchmaschine weitergeleitet werden.

Die Eingabe mehrerer Worte hingegen lässt sich als Frage nach den Gemeinsamkeiten der Eingaben interpretieren. Die Suche läuft dann beispielsweise so ab:

- Bedeutungen und Typen der einzelnen Wörter in der Ontologie suchen.
- Ist keins der Suchwörter in der Ontologie enthalten, an die Volltextsuche weitergeben.
- Ist mindestens eins der Wörter nicht in der Ontologie enthalten, dann die Bedeutungen der übrigen anzeigen und Suche im Volltext für die ursprüngliche Kombination anbieten.
- Sind alle Wörter in der Ontologie enthalten, bei Bedarf disambiguieren lassen, anschließend eine SPARQL-Anfrage starten.
- SPARQL Response XML für die Anzeige aufbereiten

So lassen sich zwar noch keine wirklich komplexen Anfragen an das System stellen, allerdings können die vorhandenen semantischen Verknüpfungen der Inhalte bereits genutzt werden.

Natürlichsprachliche Formulierung Die Verwendung von NLP (natural language processing) für das Verstehen von Nutzeranfragen ist ein aktuell beforschtes Gebiet. Besonders interessant ist der Einsatz dieser Methode zusammen mit Speech-to-Text-Verfahren, d.h. bei der Auswertung von Anfragen, die z.B. per Telefon gestellt werden. Der Einsatz von NLP-Verfahren zur Anfragenbearbeitung ist selbst Stoff für Dissertationen, deshalb wird in dieser Arbeit davon Abstand genommen, diesen Weg weiter zu verfolgen.

Formularbasierte Anfragen Suchformulare finden sich besonders in datenbankgestützten Anwendungen. Mit ihnen können Nutzer ihre Anfragen umschreiben, so dass auf der Basis der angegebenen Daten die Suchen im System durchgeführt werden können.

Die Erstellung eines solchen Formulars ist eine Angelegenheit, die sich nur schlecht generalisieren lässt, da die Entscheidung über die zur Verfügung zu stellenden Felder Kenntnis über die Inhalte der Ontologie erfordert, wenn das Formular eine sinnvolle Unterstützung

leisten soll. Dennoch erscheint es sinnvoll, die Nutzer bei der Formulierung ihrer Suchanfragen durch ein Formular zu unterstützen, das ihnen den Rahmen der verfügbaren Möglichkeiten auf einfache Art zeigt.

Dazu gehören Dropdown-Felder, aus denen eine der verfügbaren Konzeptklassen ausgewählt werden kann, um den nebenstehenden Textparameter zu klassifizieren, ebenso Auswahllisten oder Dropdown-Felder für die Auswahl von Relationen, auf die eine Anfrage beschränkt werden kann, bzw. soll.

Die Inklusion von Relationen in die Parameter von Anfragen, ist einer der größten Vorteile, den Formulare gegenüber der Suchzeile bieten. Damit werden Anfragen möglich, die Daten in einem bestimmten Kontext abfragen: „Ich suche nach allen Personen, die in Köln geboren wurden und ihr Studium in Heidelberg absolviert haben“. Diese Anfrage ist mit einer Suchzeile nur schwerlich zu stellen, selbst mit dem weiter oben beschriebenen Verfahren. Wenn es die passenden Relationen *geboren_in* und *studiert_in* im Netz gibt, ist es hingegen kein Problem, diese Anfrage in einem Formular graphisch übersichtlich zusammenzustellen. Dazu kommt, dass die Definitions- und Wertebereiche von Relationen in Ontologien festgelegt werden können, d.h. man kann schon durch die verfügbaren Optionen im Formular Suchfehler ausschließen. Wenn klar ist, dass nur Personen der Ausgangspunkt einer Relation *geboren_in* sein können und nur Orte die Objekte einer solchen Relation sein können, dann kann man die Anzeige der verfügbaren Relationen für bestimmte Klassen von vornherein ausdünnen, so dass Irrläufer gar nicht erst entstehen.

Wenn man es mit statischen Datenbeständen zu tun hat, lassen sich einige zusätzliche Unterstützungen in ein Formular einbauen. So kann eine Liste mit den zu einer Klasse verfügbaren Instanzen angezeigt werden, sobald die Klasse ausgewählt ist. Dann muss der Name der Instanz nicht komplett eingetragen werden, denn Listenfilter, die auf dem Präfix der Namen arbeiten, lassen sich einfach integrieren.

Die exemplarische Realisierung einer formularbasierten SPARQL-Suche für einen gegebenen Bestand ist Thema einer jüngst erfolgreich abgeschlossenen Bachelor-Arbeit an der Fachhochschule Bonn-Rhein-Sieg gewesen. Die darin durchgeführte Evaluierung bestand zum Teil aus Nutzertests durch Fachexperten, die mit der Oberfläche in die Lage versetzt wurden, auch komplexere Anfragen an das System zu stellen [73]. Der andere Teil der Evaluierung beschäftigte sich unter anderem mit der Performanz der semantischen Suche. Hier wurden grundsätzlich die Ergebnisse der in Kapitel 5.3.2 beschriebenen Evaluierung von RDB gegen SDB anhand einer real genutzten Ontologie bestätigt.

Schlussfolgerung für das geplante System Da auf absehbare Zeit mit eher überschaubaren Nutzeranzahlen gerechnet werden kann, die in der Lage sind, SPARQL-Anfragen direkt zu formulieren, stellt die formularbasierte Eingabe von Anfragen einen guten Kompromiss zwischen der Erhaltung der Mächtigkeit von Anfragen und der Ermöglichung der Suche für einen möglichst weiten Nutzerkreis dar.

Allerdings lassen sich Suchformulare nur schwer generisch realisieren, da eine Anpassung auf den verfügbaren Datenbestand sehr wünschenswert ist. Hier ist Arbeit beim Aufsetzen konkreter Anwendungsprojekte auf der Basis des vorliegenden Systems nötig.

5.4.3 Unterstützung dynamischer Datenbestände

Die bisher in diesem Kapitel beschriebenen Ansätze zur Erzeugung von Ontologien gingen implizit von einem statischen Datenbestand aus. Die Dokumente werden mit NER verarbeitet, Relationen zwischen den einzelnen Klassen werden extrahiert und die gesammelten Informationen nach Durchsicht der Experten in eine Ontologie übertragen. So erstellte Ontologien eignen sich gut für Anwendungen, in denen es darum geht, die Recherche auf abgeschlossenen Sammlungen zu erleichtern, neue Ansichtsmöglichkeiten zu schaffen und die Sammlung fit für die Einbeziehung in das Semantic Web zu machen.

Wie sieht es jedoch aus, wenn man nicht von einer statischen Basis ausgeht, sondern zulässt, dass diese sich ändern kann? Daten können korrigiert werden, neue Dokumente kommen hinzu, überholte werden gelöscht. Eine Ontologie, die sich nicht mit ihrer Datenbasis entwickeln kann, wird mit der Zeit deren Wissensstand immer weniger wiedergeben, also eine Sicht auf eine Welt präsentieren, die es so gar nicht mehr gibt. Damit wäre sie als Recherchetool und als Basis des Diskurses einer Gruppe von Personen über die Inhalte der betreffenden Sammlung nicht mehr zu gebrauchen.

Anwendungsszenarien mit dynamischen Dokumentensammlungen sind zum Beispiel der Einsatz als semantischer Dokumentenserver für eine Einrichtung, als semantisches Portal zu den Inhalten eines Archivs (hier beschränkten sich die Änderungen wahrscheinlich auf das Hinzufügen neuer Dokumente) oder aber die Einbeziehung der Daten aus einem Wiki-System, das sich mit dem Themenkreis der Ontologie beschäftigt.

In diesen Szenarien muss sicher gestellt werden, dass die Ontologie auf dem gleichen Stand wie die Dokumentenbasis ist. Je nach Häufigkeit der Änderungen und strategischer Bedeutung des Systems für die Einrichtung kann es ein gangbarer Weg sein, die Ontologie in festen Intervallen neu zu erzeugen und die alte Version zu ersetzen. Das kann nachts geschehen, aber auch so, dass im Hintergrund eine neue Ontologie erzeugt wird, die dann für die Nutzer transparent die aktuell verwendete ablöst. Dieses Vorgehen wird oft im verwandten Gebiet der Volltextindexierung von Dokumentensammlungen angewandt, so dass bei nächtlichen Updates der Index bis zu einem Tag der Realität hinterher hinkt.

In Szenarien mit großen Datenbeständen, bzw. solchen, in denen sich Teile der Dokumente häufig ändern können, empfiehlt sich dieses Vorgehen nicht. Der Aufwand für die anhaltende Regenerierung der Ontologie ist nicht zu unterschätzen, zumal ein großer Teil der Bestände sich nicht gegenüber der letzten Version der Ontologie geändert haben wird. In diesen Fällen ist es besser, eine Aktualisierungsstrategie zu verfolgen, in der nur die Teile der Ontologie bearbeitet werden, die durch die geänderten Dokumente tatsächlich

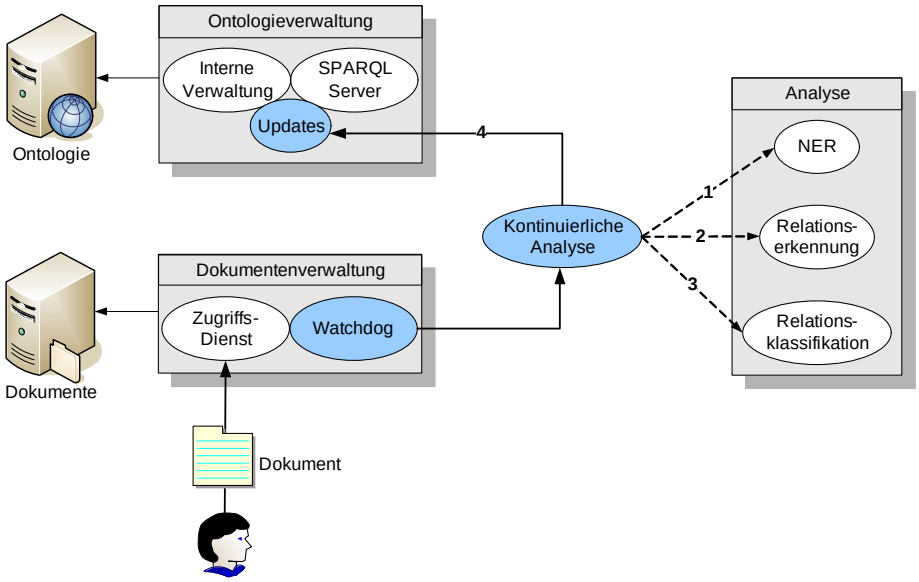


Abbildung 5.6: Einbettung der kontinuierlichen Analyse in das System

betroffen sind.

Dieses Vorgehen erfordert das Nachhalten der Herkunft der in der Ontologie enthaltenen Fakten, so dass sich mit Sicherheit sagen lässt, aus welchem Dokument die Informationen stammen, die in der Ontologie enthalten sind. Erforderlich hierfür ist die Einführung von Relationen, die Metadaten zu den einzelnen Aussagen der Ontologie enthalten, daher sollte die Entscheidung zur Unterstützung dynamischer Datenbestände zum Zeitpunkt des Systemaufbaus getroffen werden, um eine nahtlose Erfassung und Integration der benötigten Zusatzdaten zu ermöglichen.

Die nachfolgenden Abschnitte stellen verschiedene Wege vor, mit denen sich die Ontologie aktualisieren lässt, um ein Auseinanderdriften von Ontologie und Wirklichkeit zu verhindern.

Kontinuierliche Analyse

Dieser Abschnitt stellt ein Modul vor, das zur Überwachung von Änderungen an der Dokumentensammlung mit anschließender Aktualisierung der Ontologie verwendet werden kann. Abbildung 5.6 zeigt die Stellen, an denen Änderungen im System vorgenommen werden müssen, damit dieses Modul seine Aufgaben erfüllen kann. Diese betreffen zunächst den Dienst, der für die Verwaltung der Dokumentendatenbank verantwortlich ist. Er wird um einen Dienst ergänzt, der Änderungen an der Datenbasis erkennt und an andere Dienste weiterleitet, den so genannten *Watchdog*-Dienst. Die Abstände, in denen der Dienst anschlägt, sind dabei frei wählbar.

Wenn der Watchdog Änderungen in der Datenbasis meldet, wird vom Analyse-Dienst die Verarbeitungskette für die betreffenden Dokumente angestoßen, d.h. die Eigennamen- sowie die Relationserkennung werden durchgeführt, anschließend die Resultate gegen die Referenzrelationen geclustert. Der letzte Schritt in der Verarbeitungskette ist die Übertragung der Ergebnisse in die Ontologie. Dazu wird der Ontologieverwaltungsdienst verwendet, der dafür um eine Schnittstelle erweitert werden muss, die es erlaubt, Fakten aus spezifischen Dokumenten zu aktualisieren.

Bei der Verwendung dieses Moduls sind einige Punkte zu berücksichtigen:

1. Die Verarbeitung der Dokumente erfordert eine gewisse Zeit, Änderungen der Ontologie in Echtzeit wird es nicht geben.

2. Die tatsächliche Änderung der Ontologie muss sehr schnell gehen, um den Zeitraum kurz zu halten, in dem es zu eventuellen Inkonsistenzen kommen kann. Idealerweise wird die Ontologie für diesen Zeitraum kurz mit einem Read-Lock versehen.

   Wenn Hardware keine Rolle spielt, kann eine gespiegelte Kopie vorgehalten werden, so dass Änderungen immer abwechselnd auf der jeweiligen Offline-Kopie durchgeführt werden, die nach Beendigung des Updates online geht. Die aktuelle Änderung wird in der dann nicht verwendeten Kopie nachgezogen, diese steht dann für die darauf folgende Änderung zur Verfügung und so weiter. So lässt sich das Inkonsistenz-Problem eliminieren.

3. Im Fall von Web-Anwendungen sollten die Seiten mit einer refresh-Anweisung ausgeliefert werden, damit die Browser automatisch in bestimmten Abständen eine neue Kopie der Seite anfragen. So zeigen auch länger in einem Browser geöffnete Webseiten die aktuellen Daten an und keine alten Stände (dangling pointer Problem).

Ermöglichen manueller Korrekturen

Der zuvor gezeigte Ansatz arbeitet Änderungen in die Ontologie ein, die sich aus der Änderung der Zusammensetzung der Dokumentenbasis ergeben. Das ist ein automatischer Prozess auf der Basis der von den Experten festgelegten Klassen und Relationen.

Nun kann es aber sein, dass Fakten in der Ontologie enthalten sind, die schlicht falsch sind, sei es, dass Dokumente Fehler enthalten oder bei einer optischen Zeichenerkennung Fehler in das Transkript eines Digitalisats eingebracht worden sind. In solchen Fällen ist es wünschenswert, eine Möglichkeit anzubieten, die manuelle Korrekturen einzelner Fakten erlaubt. Die Realisierung ist nicht für alle Anwendungsszenarien in gleichem Umfang möglich, zudem jeweils von den gewählten Ausgabemedien abhängig sowie der Nutzergruppe abhängig. Hier spielen Fragen der Authentifizierung und nicht zuletzt des Vertrauens in die Kompetenz der eigenen Nutzer eine gewichtige Rolle.

Gerade in wissenschaftlichen Kontexten kann es jedoch sehr nützlich sein, eine einfache Möglichkeit vorzuhalten, die eine Vervollständigung bzw. Korrektur der präsentierten Fakten erlaubt. Dies kann, einen Web-Kontext vorausgesetzt, über ein Formular geschehen, das an die jeweiligen Konzeptklassen angepasst ist. In diesem Formular könnten Fakten zu einer Instanz präsentiert werden, so dass autorisierte Nutzer diese ergänzen oder korrigieren können. In anderen Umgebungen lassen sich solche Formulare auch realisieren, etwa im Rahmen eines Wartungsmoduls, das Teil der Applikation zum Zugriff auf die Ontologie ist.

Die Auswertung dieser Änderungen profitiert davon, dass keine Vorverarbeitung notwendig ist, da die Objekte der Relationen direkt im Kontext der Ontologie geändert werden.Deshalb können die Daten direkt in die Ontologie zurückgeschrieben werden. Natürlich nur unter Angabe der Herkunft der Daten (deshalb sollten nur dem System bekannte Nutzer Änderungen durchführen dürfen), falls Fragen oder Zweifel an der Herkunft oder Korrektheit auftauchen sollten. Die Qualität manuell nacherfasster Daten hat allerdings zunächst Vorrang gegenüber maschinell erschlossenen.

In diesem Kontext könnte die Frage auftauchen, warum solche Änderungen nicht einfach an den Dokumenten selbst vorgenommen werden, immerhin beschäftigt sich das weiter oben beschriebene Analyse-Modul mit der Auswertung von Dokumentenänderungen. In Fällen, in denen die Dokumente geändert werden können, ist das auch richtig. Allerdings können nicht alle Dokumente einfach geändert werden. Wenn ein Digitalisat eines Buches Teil der Dokumentensammlung ist, können darin nicht einfach Fakten geändert werden, ohne dass der Wert des Buches als Primärquelle gemindert wird. In diesen Fällen kann man sich nur darauf beschränken, die richtige Information einzupflegen und entweder die Aussage der Dokumentenquelle zu entfernen oder diese Fehlinformation durch eine spezielle Relation zu kommentieren (z.B. über Reifikation).

Schlussfolgerungen für das geplante System

Die beiden vorgestellten Wege adressieren das Problem wie sichergestellt werden kann, dass Ontologien im Einklang mit der sie umgebenden Welt gehalten werden können. Der erste Weg verfolgt Änderungen an der Dokumentenbasis, um gegebenenfalls Änderungen an der Ontologie vornehmen zu können, der zweite Weg erlaubt das Korrigieren von Fehlern, die sich maschinell nicht entdecken lassen, die aber ebenfalls dazu beitrügen, dass die Ontologie einen verzerrten Blick auf die Welt gewährte.

Diese beiden Wege können durchaus gemeinsam für eine Applikation implementiert werden, um die Möglichkeit zu bieten, sowohl Änderungen in den Dokumenten, als auch Erkenntnisse der Domänenexperten in der Ontologie angemessen berücksichtigen zu können.

5.5 Zusammenfassung

Dieses Kapitel hat die Ansätze vorgestellt, die zur Realisierung eines Systems führen, das die in Abschnitt 2.1 aufgestellten Ziele unter Berücksichtigung der Anforderungen aus Abschnitt 2.3 erfüllt. Zu jedem der in diesen beiden Abschnitten identifizierten Bereiche sind entsprechende Ansätze präsentiert worden, aus denen sich die Architektur des benötigten Systems ableiten lässt. Das nächste Kapitel stellt die Referenzimplementierung der hier vorgestellten Ansätze im Detail vor.

Kapitel 6

Systeme zur Erstellung von Ontologien

Dieses Kapitel beschreibt Systeme, die zur Erforschung der Möglichkeiten und Grenzen der (semi-)automatischen Ontologieerstellung im Rahmen der Dissertation entwickelt worden sind. Den größten Teil nimmt dabei die Beschreibung des WIKINGER-Systems ein, das im Rahmen eines gleichnamigen vom BMBF geförderten Forschungsprojekts umgesetzt worden ist. Kapitel 6.1 beschreibt die Rahmenbedingungen des Projekts, den verfolgten Ansatz, sowie die Systemarchitektur, während Kapitel 6.2 näher auf die Komponenten des Projekts eingeht, die im Kontext dieser Dissertation von Belang sind.

Daran anschließend wird in Kapitel 6.3 ein System beschrieben, das zur Erforschung von Einsatzmöglichkeiten für automatisch erstellte Ontologien entwickelt worden ist.

6.1 WIKINGER – Wiki Next Generation Enhanced Repositories

Das WIKINGER-Projekt[1] [19, 20] wird im Rahmen der zweiten Ausschreibung des e-Science-Programms des BMBF gefördert, die unter dem Motto "e-Science und vernetztes Wissensmanagement" steht. Im Projekt wird versucht, mit Hilfe neuer Verfahren und Techniken die Zusammenarbeit räumlich verteilter wissenschaftlicher Communities zu verbessern. WIKINGER ist ein Verbundvorhaben des Fraunhofer-Instituts für Intelligente Analyse- und Informationssysteme in Sankt Augustin zusammen mit der Universität Duisburg-Essen, Lehrstuhl für wissensbasierte und natürlichsprachliche Systeme[2] unter

[1] siehe `www.wikinger-escience.de`

[2] siehe www.uni-due.de/computerlinguistik/index.shtml

Professor Dr. Hoeppner, sowie der Kommission für Zeitgeschichte (KfZG) in Bonn[3].

Die Beschreibung beginnt mit einer Einordnung des Projekts in die Forschungslandschaft, gefolgt von einem Projektüberblick und der zu Grunde liegenden Software-Architektur, ehe in Kapitel 6.2 die im Kontext dieser Dissertation relevanten Teile des Projekts näher betrachtet werden.

6.1.1 Das e-Science-Programm des BMBF

Das Bundesministerium für Bildung und Forschung hat sich, beginnend im Jahr 2003, verstärkt der Förderung des Grid-Computing zugewandt. Dabei handelt es sich um Aktivitäten, die darauf abzielen, neben Informations- auch Hardware-Ressourcen im Internet in einer organisierten Art und Weise zur Verfügung zu stellen, etwa um besonders rechenintensive Arbeiten auf mehrere Rechner an verschiedenen Standorten und insbesondere in verschiedenen Organisationen verteilen zu können. Besonders interessant sind diese Möglichkeiten für die Wissenschaft, fallen doch bei vielen, gerade natur- und ingenieurswissenschaftlichen, Disziplinen heutzutage sehr große Datenmengen bei Experimenten an, die zeitnah verarbeitet werden müssen.

Viele einzelne Organisationen sind damit überfordert, erst durch die Verteilung auf mehrere Einrichtungen können solche Daten bewältigt werden. Hierzu ist die Entwicklung neuer Protokolle und Verfahren nötig, um Sicherheit, Dienstqualität, Authentifizierung und Abrechnung für solche Angebote abbilden zu können. Um eine Bündelung der verschiedenen Entwicklungen auf diesem Gebiet zu ermöglichen, hat sich in Deutschland im Jahr 2003 die D-GRID – Initiative gebildet, die mittlerweile ungefähr 100 Mitgliedsorganisationen umfasst. Von dieser Initiative wurde in Absprache mit dem BMBF ein Strategiepapier erarbeitet, das Mitte 2003 veröffentlicht wurde und Eckdaten für die anschließend vom BMBF geförderten Aktivitäten zur Verfügung stellte.

Das BMBF hat flankierend zu diesen Maßnahmen ein Forschungsprogramm mit Namen “e-Science“ aufgelegt, innerhalb dessen unter anderem die Aktivitäten im Grid-Computing gefördert werden. Eine Definition des Begriffs *e-Science* findet sich in einem Folgepapier zum D-Grid-Strategiepapier aus dem Jahr 2004:

> e-Science (digitally enhanced science) ist die Bezeichnung für eine Arbeitsweise in der Wissenschaft, die durch gemeinsame, kooperative Entwicklung, Öffnung und Nutzung ihrer Ressourcen und Projekte eine wesentliche Steigerung der Qualität und Leistungsfähigkeit erreicht. Ressourcen sind wissenschaftliche Verfahren einschließlich Expertise, Software, Datenbestände, Rechner, Kommunikationsnetze und andere wissenschaftliche Geräte. In zahlreichen Disziplinen ermöglicht e-Science erst bestimmte Formen der wissenschaftlichen Arbeit, die

[3] siehe www.kfzg.de

> die Bearbeitung neuer Zielstellungen ermöglichen und so zu völlig neuen Erkenntnissen führen können.[4]

Diese Definition steckt den Rahmen für die Förderaktivitäten des BMBF recht gut ab. Ein Kernanliegen des Programms ist die Förderung des Grid-Computing, dazu kommt die Unterstützung von Forschungsprojekten, die eine neue Herangehensweise an den wissenschaftlichen Prozess untersuchen und ermöglichen sollen. Forschungsvorhaben können im Programm nur innerhalb thematischer Ausschreibungen beantragt werden, wodurch sich die Übereinstimmung der geförderten Projekte mit der strategischen Ausrichtung des Programms besser steuern lässt.

Die erste Ausschreibung vom August 2004 konzentrierte sich auf Community-Grids und den Aufbau einer Grid-Middleware-Integrationsplattform [26]. Näheres zu den darin und aktuell geförderten Projekten findet sich auf der Webseite der D-Grid-Initiative[5].

Im November 2004 folgte eine Ausschreibung mit dem Titel "e-Science und vernetztes Wissensmanagement", deren Fokus auf der Förderung der Zusammenarbeit innerhalb einzelner Fachdisziplinen, bzw. verschiedener Fachdisziplinen miteineinander, lag [25]. Hierbei wurde insbesondere Wert gelegt auf die Entwicklung neuartiger Methoden zur Förderung dieses Ziels, die skalierbar und auch außerhalb des Projekts verwertbar sein sollten – unter Umständen sogar kommerziell.

Die Verbindung zu den Grid-Aktivitäten bildet die Konzentration auf web-gestützte Verfahren zur Förderung der Zusammenarbeit, d.h. zu der Verfügbarkeit von Ressourcen über das Netz sollen sich Mechanismen gesellen, die es Wissenschaftlern ermöglichen, auch tatsächlich mit Wissenschaftlern an anderen Standorten oder anderen Einrichtungen effizient zusammen zu arbeiten.

6.1.2 Projektüberblick

Trotz der technischen Möglichkeiten des Internets ist die Art und Weise, in der heutzutage die Kommunikation innerhalb wissenschaftlicher Communities abläuft, immer noch durch die klassischen Kanäle der Wissenschaft geprägt: Konferenzen, Veröffentlichungen in wissenschaftlichen Journalen sowie bilaterale Kommunikation.

Die durch das Internet hervorgebrachten Hilfsmittel wie E-Mail, Mailinglisten und im Netz verfügbare Dateiserver (sei es per FTP oder moderneren Varianten wie BSCW oder SharePoint) ergänzen diese Kanäle nachhaltig. Darüber hinaus befinden sich allerdings nur wenige Hilfsmittel im Einsatz.

Dabei bietet gerade das Semantic Web ein ungeheures Potenzial für e-Science-Anwendungen: Mit den zur Verfügung stehenden Techniken können über das Internet semantische

[4] Siehe [36], Seite 4

[5] `http://www.d-grid.de`

Beschreibungen beliebiger Ressourcen in standardisierten Formaten angelegt, verwaltet, ausgetauscht und ausgewertet werden – von Menschen wie von Maschinen. Gerade für die Schaffung neues Wissens bieten sich dadurch große Chancen: Die Ortsabhängigkeit wird aufgebrochen, Forscher können sich zu virtuellen Communities zusammenschließen und gemeinsam an Fachthemen arbeiten, deren Ergebnisse für alle einsehbar abgelegt werden können. Folgerichtig geht es in WIKINGER darum, domänenneutrale Methoden und Verfahren zu entwickeln, mit denen fachspezifische Wissensbasen angelegt und über das Internet für den Zugriff durch Fachcommunities zur Verfügung gestellt werden können. Die Wissenschaftler sollen dabei nicht nur lesenden Zugriff auf das gesammelte Wissen erhalten, sondern aktiv an dessen Erweiterung teilnehmen können. Dabei ist die Vernetzung neu geschaffener Beiträge mit dem bereits Vorhandenen besonders wichtig, weshalb hierfür im Projekt ebenfalls Methoden entwickelt werden sollen.

Durch diese Möglichkeiten wird es Wissenschaftlern vereinfacht, sich Hintergrundmaterial zu ihrer aktuellen Arbeit zu beschaffen und gleichzeitig wird den Rezipierenden der Zugang zu den neuen Erkenntnissen deutlich erleichtert, da Verbindungen zu bereits Bekanntem erzeugt werden – und diese Verbindungen sind nur den fast schon sprichwörtlichen Klick entfernt.

Ausgangslage, Pilotanwendung für die Plattform

Auslöser für den Projektantrag war eine Situation wie sie bereits in Kapitel 1 beschrieben und die deswegen als Pilotanwendung für die geplante Plattform ausgewählt worden ist: Die Kommission für Zeitgeschichte (KfZG) in Bonn beschäftigt sich mit der Erforschung der katholischen Zeitgeschichte in Deutschland vom 19. Jahrhundert bis heute. Ergebnisse dieser Forschungen, üblicherweise Dissertationen und Habilitationen, erscheinen in einer von der KfZG herausgegebenen Schriftenreihe, der so genannten “Blauen Reihe“.

Seit Gründung der KfZG im Jahr 1962 sind ungefähr 150 Bücher in dieser Reihe veröffentlicht worden. Die Büchern behandeln zwar Themen über einen Zeitraum von gut 250 Jahren, trotzdem finden verschiedene Personen, Orte, Organisationen und Ereignisse in mehr als einem der Bücher Erwähnung. Die Verknüpfung der unterschiedlichen Kontexte dieser Bücher über die erwähnten Entitäten macht eine Zusammenführung in einer gemeinsamen Sammlung interessant, würde doch so eine Gesamtsicht auf das Fach ermöglicht, die ein einzelner Autor so nicht herstellen kann.

Dem stand bisher der Umfang der zu betrachtenden Entitäten im Weg, die KfZG schätzt allein die Anzahl der mit Biogramm erwähnten Personen auf ca. 30.000, wobei hier noch Dubletten enthalten sind. Die Zahl der unterschiedlichen Personen dürfte bei ungefähr einem Drittel davon liegen.

Ziel der KfZG ist es, ein biographisch-bibliographisches Handbuch des deutschen Katholizismus des 19. und 20. Jahrhunderts zu erzeugen, das über das Internet der wissenschaftlichen Community zur Verfügung steht und von dieser durchsucht und bei Bedarf

ergänzt werden kann. Ferner ist die Anbindung von Datenquellen verwandter Einrichtungen angedacht, die sich mit verwandten Themengebieten beschäftigen, etwa Bilddatenbanken wichtiger Persönlichkeiten des Gebiets.

Das Semantic Web bietet die Möglichkeiten, die Verknüpfungen der Entitäten untereinander zu formulieren und somit ein thematisches Netz für die zeitgeschichtliche Erforschung des deutschen Katholizismus aufzuspannen. Dieses Netz lässt sich zur Veröffentlichung im Internet nutzen, wodurch auch andere Forscher des Gebiets Zugriff auf die erfassten Daten erhalten.Ein weiterer Wunsch ist es, die Ergänzung der Daten zu ermöglichen, also Wissenschaftlern die Möglichkeit einzuräumen, direkt das Netz zu ergänzen und so ihre eigenen Erkenntnisse in es einfließen zu lassen.

6.1.3 Ansatz

Die Ziele der Pilotanwendung verdienen eine weitere Betrachtung, da sich von ihnen abstrahiert Anforderungen an ein generalisierteres System ableiten lassen. Auch in anderen Fachgebieten gibt es umfangreiche Textkorpora, deren Themen über verschiedene Arten von Entitäten miteinander verbunden werden können, um eine Gesamtsicht auf das Themengebiet zu erzeugen, ebenso generisch ist die Anforderung, die so erzeugten Verbindungsnetze zu visualisieren und mit anderen Forschern zu teilen, bzw. ihnen die Ergänzung oder Änderung des Netzes zu erlauben. Insofern lassen sich generalisiert die folgenden Anforderungen an ein System ableiten, das die Erfüllung solcher Ziele ermöglicht:

1. Vokabularien zur Beschreibung der Metadaten benötigter Entitätenklassen müssen integriert werden können.
2. Das System muss den Import von Daten aus verschiedenartigen Quelltypen erlauben.
3. Es muss technische Unterstützung bei der Extraktion von Entitäten verschiedener Klassen leisten.
4. Die Verknüpfung der Entitäten muss technisch unterstützt werden, da die manuelle Erstellung des Netzes mit mindestens 10.000 Entitäten nicht von der Community geleistet werden kann.
5. Da das System auf die Kooperation der Domänenexperten angewiesen ist, müssen diese in die Lage versetzt werden, ohne lange Einarbeitungszeit die von ihnen erbetenen Aufgaben wahrnehmen zu können
6. Wissenschaftler müssen mit dem semantischen Netz interagieren können, neben einer reinen Ansicht der Daten ist unter Umständen auch eine Änderungsmöglichkeit vorzusehen.
7. Das System muss in der Lage sein, Ergänzungen des Netzes um neue Quellen durchzuführen.

Mit der WIKINGER-Plattform wird ein Software-System entwickelt, das die oben genannten Anforderungen erfüllt und somit die Etablierung einer semantisch vernetzten über das Internet zugreifbaren Wissensbasis für ein Fachgebiet ermöglicht. Die Ähnlichkeiten der oben aufgeführten Anforderungen zu denen, die in Kapitel 2.3 aufgestellt worden sind, liegen auf der Hand. In der Tat ist das Design der Plattform dergestalt angelegt, dass sie alle Anforderungen aus Kapitel 2.3 erfüllt und damit als Referenzimplementierung dienen kann.

Hier nicht explizit aufgeführt sind die Anforderungen A4 (Annotation von Beispielen), A5(Verknüpfungen auf Instanzebene), A7(erweiterbare Architektur) sowie A8 (Laufzeit). Diese werden zum Teil von den anderen oben aufgeführten Anforderungen subsummiert, zum Teil werden sie durch die Designentscheidungen der WIKINGER-Plattform vorweg genommen. Im weiteren Verlauf dieses Kapitels wird auf die Schritte zur Erfüllung der Anforderungen aus Kapitel 2.3 gesondert eingegangen.

Zur Einführung der Plattform beginnt die Erläuterung mit einem Überblick: Abbildung 6.1 zeigt den typischen Arbeitsablauf in einem WIKINGER-System[6].

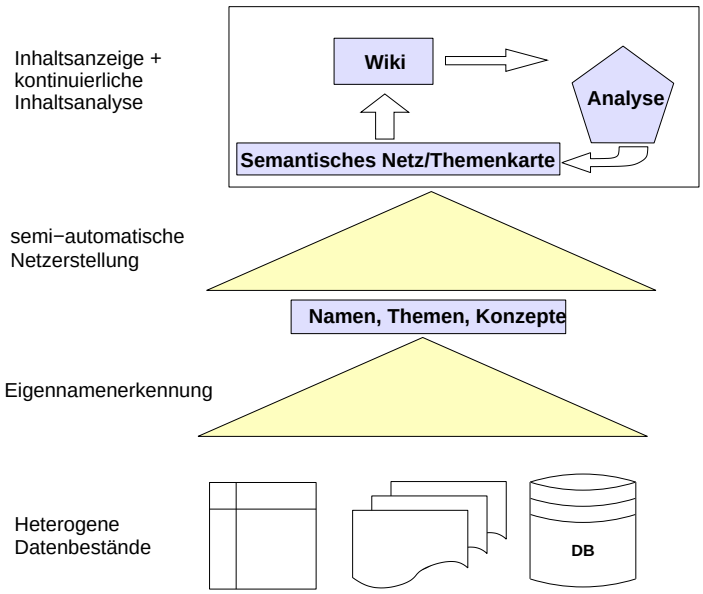


Abbildung 6.1: Schema des Arbeitsablaufs in WIKINGER

Der Prozess beginnt mit einer Sammlung von Dokumenten, die in das System integriert werden sollen. Dabei kann es sich um Dokumente unterschiedlicher Provenienz handeln, exemplarisch dargestellt sind tabellarische Daten, Schriftdokumente und relationale Datenbanken. Aus den Dokumenten werden in einem ersten Prozess, der Eigennamenerkennung, automatisch die enthaltenen Entitäten extrahiert. Die Entitätsklassen werden zur Konzeptionszeit des Projekts festgelegt und deren automatische Extraktion anhand von Beispielen

[6]Ein WIKINGER-System ist eine Installation der WIKINGER-Plattform.

trainiert.

Sobald die Entitäten extrahiert sind, kann aus ihnen ein semantisches Netz des Fachgebiets erstellt werden. Dazu werden die Entitäten hinsichtlich der zwischen ihnen bestehenden Verknüpfungen analysiert. Dies geschieht in einem automatischen Prozess, in dem alle automatisch extrahierbaren Relationen gefunden und den Experten zur Entscheidung über deren Integration in das semantische Netz vorgelegt werden. Die ausgewählten Relationen können außerdem benannt werden. Diese Aufgaben können am besten von den Domänenexperten ausgeführt werden, daher handelt es sich nur um ein semiautomatisches System. Die so gefundenen Relationen decken im Wesentlichen die Beziehungen zwischen Entitäten verschiedener Typen ab, Koreferenzrelationen werden in diesem Projekt nicht berücksichtigt, das wäre eine Aufgabe für eine mögliche Erweiterung des Systems. Sobald die Entitäten und ihre möglichen Relationen bestimmt sind, kann daraus ein semantisches Netz des Bestands erstellt werden.

Mit der Fertigstellung des semantischen Netzes ist die Vorbereitungsphase abgeschlossen. Für ein statisches System reicht es nun, eine Visualisierung des Netzes zur Verfügung zu stellen, die eine Durchsicht der Daten ermöglicht. Dazu wird eine Webanwendung benötigt, die in der Lage ist, die verschiedenen Verknüpfungen auszuwerten und ihre Bedeutungen bei der Verlinkung anzuzeigen. Für ein System, das ändernden Zugriff auf die Daten erlaubt, ist etwas mehr Aufwand zu treiben, da hierbei die Infrastruktur für die Verwaltung der Änderungen zur Verfügung gestellt werden muss.

Für den Zweck des kollaborativen Arbeitens an Texten im Internet gibt es bereits eine Klasse von Webanwendungen, die sich großer Beliebtheit erfreut[7]: Wiki-Systeme. Für eine einführende Betrachtung von Wiki-Systemen sei auf Kapitel 3.3.4 verwiesen. Für die Verwendung eines Wikis als Oberfläche für ein semantisches Netz, das Nutzer verändern können sollen, sprechen die Vorteile dieser Technologie:

Sie sind einfach zu bedienen. Zum Erfolg der Wiki-Systeme hat die einfache Oberfläche einen großen Teil beigetragen. Änderungen an Inhalten sind sehr schnell vorgenommen, Kenntnisse in HTML oder CSS sind für eine Beteiligung nicht von Nöten und die Ergebnisse der Änderungen können direkt angeschaut werden. Diese Faktoren senken die Eintrittshürden erheblich und sorgen für breite Akzeptanz des Systems auch bei technikfernen Anwendern.

Ihre Inhalte sind versionsverwaltet. Wiki-Systeme implementieren eine einfache Fassung der Versionsverwaltung, d.h. frühere Versionen von Artikeln sind weiterhin im System gespeichert und können auf Knopfdruck wieder hergestellt werden. Damit ist die Historie eines Artikels jederzeit einsehbar, je nach Einrichtung des Systems ist auch nachvollziehbar, wer welche Änderungen eingebracht hat. Wiki-Systeme enthal-

[7] Jedenfalls, wenn man sich die Wikipedia anschaut, die eine sehr große Menge aktiver, freiwilliger und unbezahlter Nutzer/Autoren in der ganzen Welt aufweist und fast im Alleingang für die große Verbreitung von Wikis außerhalb der Programmiererszene gesorgt hat.

ten damit einen Schutz vor fehlerhaften Änderungen, seien sie nun fahrlässig oder bewusst herbei geführt.

Wikis gleichen Ontologien in ihrem Aufbau. Wiki-Systeme bestehen aus einer Menge von Webseiten, die sich jeweils mit einem bestimmten Thema befassen und Verbindungen zu thematisch verwandten Seiten enthalten. Wenn man die einzelnen Seiten als Konzepte auffasst, hat man ein semantisches Netz vor sich. Insofern lässt sich ein semantisches Netz auf ein Wiki-System abbilden, Konzepte und ihre Klassen als Seiten, die Verknüpfungen als Links zwischen den Konzepten. Geeignete Implementierung vorausgesetzt bleibt auch die semantische Qualität der Verknüpfungen erhalten.

Zur Nutzung dieser Vorteile wird im Projekt ein Wiki-System als Benutzerschnittstelle eingesetzt. Zusätzlich zu den Vorteilen aus Nutzersicht ergibt sich für die Implementierung der weitere Vorteil, dass ein großer Teil der Interaktionsmöglichkeiten in dem System bereits implementiert sind und deshalb nicht aufwändig neu entwickelt werden müssen. Das im Vorverarbeitungsschritt erzeugte semantische Netz wird in das Wiki übertragen. Zusätzlich werden, soweit möglich, die Artikel mit passenden Texten aus den Buchbänden vorbelegt, damit ein Kern zur Verfügung steht, der von den Nutzern ausgearbeitet werden kann. Über die vom Wiki bereitgestellte Schnittstelle können die Nutzer mit dem System interagieren und ihre Ergänzungen vornehmen, also Texte ändern oder neue Artikel einstellen. Über ein Analyse-Modul werden die Änderungen im Wiki überwacht und auf Auswirkungen für das semantische Netz untersucht. Dabei werden verschiedene Arten von Änderungen unterschieden:

Textänderung in einem bestehenden Artikel: In diesem Fall wird der aktuelle Text des Artikels nach Vorkommen neuer oder bereits bekannter Konzepte aus dem Netz durchsucht, da sich aus solchen Änderungen neue Verknüpfungen oder sogar neue Konzepte für das semantische Netz gewinnen lassen. Analog gilt das für den Fall, dass bereits in vergangenen Iterationen erkannte Konzepte aus dem Artikel entfernt worden sind. Auch diese Korrekturen werden im Netz vermerkt.

Neue Artikel: Das Erstellen eines neuen Artikels ist ein Indiz dafür, dass ein Konzept in das Netz einzufügen ist. Zusätzlich werden der Text des Artikels sowie evtl. eingefügte Links analysiert.

Artikellöschung: Dieser Fall wird erfahrungsgemäß eher selten eintreten, jedoch ist auch die Löschung eines Artikels eine Änderung, die im semantischen Netz nachgehalten werden muss. Die meisten dieser Fälle werden allerdings eigentlich Veränderungen des Artikels sein, etwa wenn der Artikel umbenannt wird, um Rechtschreibfehler zu korrigieren.

Jede dieser Änderungen führt unter Umständen zu Änderungen im semantischen Netz, so dass es weiterhin auf dem aktuellen Stand bleibt und den Datenbestand adäquat beschreibt.

Ein zusätzlicher Nutzen der Analyse ist die Möglichkeit, gefundene Konzepte direkt im Fließtext mit den dazugehörigen Artikeln zu verlinken. Erste Überlegungen des Autors zu einer möglichen Architektur, die Wikis zum Navigieren und Verfeinern von Ontologien verwendet, finden sich in einem Beitrag für die *GI-Jahrestagung 2005* [18].

6.1.4 Architektur

Zur Umsetzung der WIKINGER-Plattform ist ein Web Service-orientierter Ansatz gewählt worden, d.h. es ist kein klassisches Client-Server-System mit einem monolithischen Server und einer Reihe von Clients, die auf den Server zugreift. Statt dessen besteht die Server-Seite der Plattform aus einer Reihe von Diensten, welche die benötigten Funktionalitäten zur Verfügung stellen – sowohl den anderen Diensten des Servers, als auch den potenziellen Nutzern des Systems. Die Verwendung dieser Art von Software-Architektur bringt für die Plattform verschiedene Vorteile mit sich:

- Die Dienste lassen sich auf mehrere Rechner verteilen, so dass z.B. rechenzeitintensive Dienste auf einen eigenen Rechner ausgelagert werden können. In der Tat eignet sich so ein System auch sehr gut für eine Verteilung über mehrere Standorte. In einem Grid-Kontext könnte so z.B. die Datenhaltung von einem Dienst mit angeschlossenem Massenspeicher in Institution A verwaltet werden, während verarbeitende Dienste von Institution B aus auf die Daten zugreifen.

- Das System lässt sich verhältnismäßig einfach um neue Dienste erweitern, da schon die Architektur einen modularen Aufbau unterstützt und fordert. So werden neue Funktionalitäten folgerichtig von neuen Diensten zur Verfügung gestellt, eine kosten- und fehlerintensive Erweiterung bestehender Dienste ist nicht vorgesehen.

- Die Architektur fördert den Blick auf funktionale Einheiten, da sich entlang dieser Einheiten die Abgrenzung der Dienste zueinander fast natürlich ergeben. Dadurch lässt sich schon die Entwicklung gut parallelisieren, da die Schnittstellen zwischen den Einheiten frühzeitig festgelegt werden können, bzw. müssen. Auf diese Weise lassen sich rasch Versionen des Systems zur Verfügung stellen, deren Funktionalitäten zwar beschränkt, aber in sich bereits voll funktionsfähig sind und iterativ weiter vervollständigt werden. Dies ermöglicht es, bereits in frühen Phasen des Projekts mit dem Testen einzelner Bestandteile, in diesem Fall einzelner Dienste, beginnen zu können.

Abbildung 6.2 zeigt eine schematische Darstellung der funktionalen Einheiten eines WIKINGER-Systems. Es handelt sich dabei in erster Näherung um ein System mit drei Schichten, bestehend aus einer Ressourcenschicht, einer Dienstschicht und darüber liegend der Anwendungsschicht.

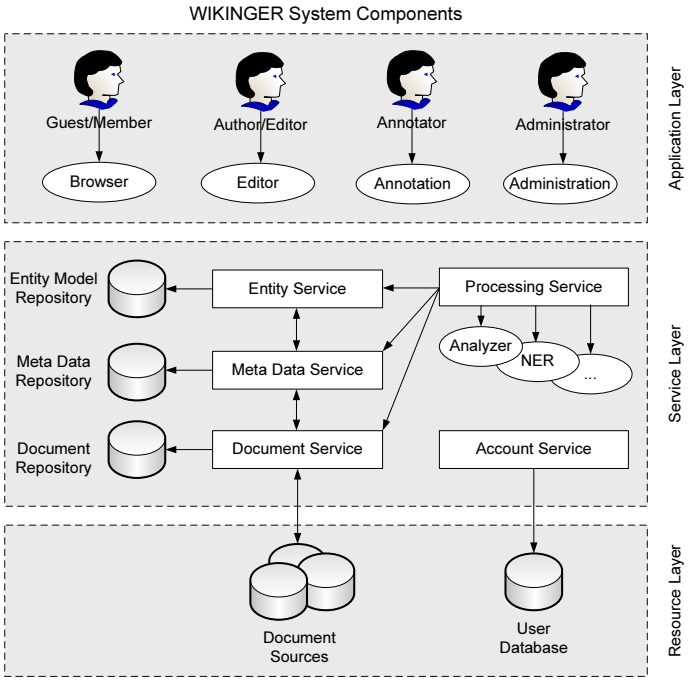


Abbildung 6.2: Komponenten eines WIKINGER-Systems

Die unterste Ebene, die Ressourcenschicht, enthält die verschiedenen Quellen eines WIKINGER-Systems. Einerseits sind da die Rohdaten, in der Abbildung als *Document Sources* bezeichnet. Dabei handelt es sich um digital vorliegende Dokumentensammlungen. Die Art des Zugriffs, ob von einem lokalen Filesystem oder über das Internet, ist nicht festgelegt, das System ist für beide Szenarien ausgelegt. Daneben enthält die Ressourcenschicht noch die Nutzerdatenbank, die Daten darüber enthält, welche Nutzer mit welchen Zugriffsrechten Zugang zu einem WIKINGER-System haben. Hierbei kann es sich z. B. um ein lokales LDAP handeln, das System beinhaltet allerdings auch eine eigene Nutzerverwaltung.

In der nächsten Ebene, der Dienstschicht, sind die eigentlichen Funktionalitäten des Systems angesiedelt. Die hier enthaltenen Dienste teilen sich in zwei Gruppen: Die erste Gruppe regelt den Zugriff auf die verschiedenen Datenbanken des Systems, während die zweite Gruppe Dienste enthält, die auf die Daten innerhalb dieser Datenbanken zugreifen und zum Teil neue Daten erzeugen. Die erste Gruppe enthält die folgenden Datenbanken und Dienste:

Da ist zunächst das *Document Repository*, das die in das System importierten Rohdaten enthält. Diese sind während des Imports in ein internes Format transformiert und mit grundlegenden Metadaten versehen worden, so dass sie von den Diensten des Systems weiterverarbeitet werden können. Das Repository unterstützt die Gruppierung von Dokumenten in sogenannten Pools, so dass logische Gruppen von Dokumenten angelegt werden können. Eine Verschachtelung von Pools wird nicht unterstützt.

Dokumente werden versioniert abgelegt, d.h. verschiedene Stände von Dokumenten lassen sich nachhalten und auf ihre jeweiligen Unterschiede hin untersuchen. Diese Funktionalität wird für statische Dokumentensammlungen üblicherweise nicht benötigt, sobald man es jedoch mit dynamischen Sammlungen zu tun hat, ist die Möglichkeit des Vergleichs verschiedener Versionen des gleichen Dokuments sehr hilfreich. Den Import von Daten sowie den Zugriff auf das Document Repository regelt der *Document Service*.

Darüber befindet sich das *Meta Data Repository*. Darin werden alle Metadaten abgespeichert, die im Verlauf der Weiterverarbeitung zu den im System enthaltenen Dokumenten erzeugt werden. Jeder Datensatz im Meta Data Repository ist einem Dokument im Document Repository zugeordnet. Dies ist insbesondere für Dokumente wichtig, die in mehreren Versionen im Document Repository vorliegen, da sich deren Metadaten zum Teil drastisch unterscheiden können. Den Zugriff auf dieses Repository regelt wiederum ein eigener Dienst, der *Meta Data Service*.

Ganz oben ist das *Entity Model Repository* dargestellt, das die Ontologie verwaltet. Die Fakten der Ontologie sind mit den Dokumenten annotiert, aus denen sie stammen. So lassen sich Fakten einfach entfernen, die auf einer veralteten Version eines Dokuments basieren. Den Zugriff auf diese Datenbank regelt der *Entity Service*. In der Abbildung ist sichtbar, welche der Datenbankdienste direkten Kontakt zueinander aufnehmen können. Die eingezeichneten Pfeile zwischen Entity Service und Meta Data Service, bzw. zwischen Meta Data Service und Document Service geben die bereits erklärten Rückbezüge der einzelnen Datensätze an.

Außerdem befindet sich in dieser Gruppe auch der *Account Service*, der für die Verwaltung der Nutzer und ihrer Berechtigungen benötigte Funktionalitäten zur Verfügung stellt und und als Schnittstelle des WIKINGER-Systems zum umliegenden Betriebssystem fungiert.

Die zweite Gruppe von Diensten ist in der Abbildung unter *Processing Services* bezeichnet. Unter diesem Oberbegriff sind verschiedene Dienste zusammengefasst. Im Bild enthalten sind der NER- und der Analyzer-Dienst. Der NER-Dienst extrahiert auf der Basis erlernter Konzeptklassen Entitäten aus den Dokumenten des Document Service. Diese werden als Metadaten zum Dokument in das Meta Data Repository geschrieben. Diese Metadaten können dann vom Analyzer verarbeitet werden, um die Relationen zu bestimmen, die zwischen den verschiedenen Entitäten bestehen.

Aus den Entitäts- und Relationsdaten erzeugt ein anderer Dienst schließlich eine Ontologie des Datenbestandes, die über den Entity Service in das Entity Model Repository gespeichert wird. Die Punkte im letzten Oval unterhalb des Processing Services deuten bereits an, dass an dieser Stelle noch weitere Dienste in das System eingeklinkt werden können.

Weitere Dienste existieren bereits als Teil des WIKINGER-Systems: Als Unterstützung für die Arbeit der übrigen Services wird zum Beispiel der Volltext der Dokumente im Docu-

ment Repository in einem Volltextindex vorgehalten, der eine Schlagwortsuche innerhalb des Volltextbestands ermöglicht. Die Suche lässt sich passgenau auf verschiedene Dokumentarten eingrenzen. Für Begriffe, die nicht Teil der Ontologie sind, lässt sich so dennoch eine passgenaue und effiziente Recherchemöglichkeit anbieten.

Die oberste Schicht des Diagramms enthält die verschiedenen Benutzergruppen mit ihren jeweiligen Aufgaben bzw. Berechtigungen. Ganz links sind die einfachen Benutzer bzw. Besucher aufgetragen. Diese haben lediglich lesenden Zugriff auf das System. Direkt daneben gibt es die Editoren oder Autoren, die zusätzlich auch noch schreibenden Zugriff haben. Weiterhin gibt es die Annotatoren, deren Aufgabe die manuelle Annotation von Inhalten des Systems ist, die von den maschinellen Lernverfahren als Trainingssets benutzt werden können. Als letzte Gruppe gibt es die Administratoren, die für die technische Betreuung einer WIKINGER-Instanz zuständig sind. Je nach Bedarf haben die verschiedenen Gruppen Zugriff auf spezialisierte Werkzeuge, mit denen sie ihre Aufgaben erfüllen können. Die verschiedenen Benutzergruppen erhalten jeweils zusätzlich die Rechte der links von ihnen aufgetragenen Gruppen, ein Annotator darf also auch als Autor tätig werden.

Für Annotatoren wurde innerhalb des Projekts von der Universität Duisburg-Essen ein Tool entwickelt, das eine manuelle Annotation von Volltexten erlaubt und die Ergebnisse in der Form eines XML-Dokuments in das Meta Data Repository abspeichern kann (siehe Kapitel 6.2.3).

Für die Administration sind zwei weitere Werkzeuge geschrieben worden, die eine genauere Überwachung der in die Repositorien eingestellten Dokumente erlauben. Zum Browsen und zum Einstellen neuer Inhalte wird innerhalb des Pilotprojekts ein angepasstes Wiki-System verwendet. Für andere Anwendungen sind aber auch andere Oberflächen zum Betrachten der Inhalte einer WIKINGER-Instanz denkbar. Die Architektur des WIKINGER-Systems ist in einem Beitrag für die *1st German E-Science Conference 2007* in Baden-Baden beschrieben worden [23].

Abbildung 6.3 zeigt die Systemarchitektur des WIKINGER-Systems. In hellgrün sind die verschiedenen Dienste eingetragen, die bereits in Abbildung 6.2 schematisch aufgetragen waren. Die verschiedenen angeschlossenen Repositories greifen jeweils auf eine MySQL-Datenbank zu. Die Apache-Bibliothek OJB[8] [4] übernimmt dabei das Mapping zwischen Objekten in Java und ihrer Serialisierung in der Datenbank, so dass dieser Schritt nicht von den Anwendungsentwicklern durchgeführt werden muss. Diese können durchgehend mit Java-Objekten arbeiten und müssen nicht selbst Code für den Zugriff auf die Speicherungsebene schreiben.

Der Entity Service greift nicht auf OJB zurück, statt dessen wird die Bibliothek Jena von HP verwendet [63, 76]. Dabei handelt es sich um ein Open-Source-Projekt, das eine Engine für die Erzeugung, Manipulation, Abfrage und Serialisierung semantischer Netze in RDF und OWL zur Verfügung stellt. Jena übernimmt die Serialisierung in die Datenbank,

[8] ObJect Relational Bridge

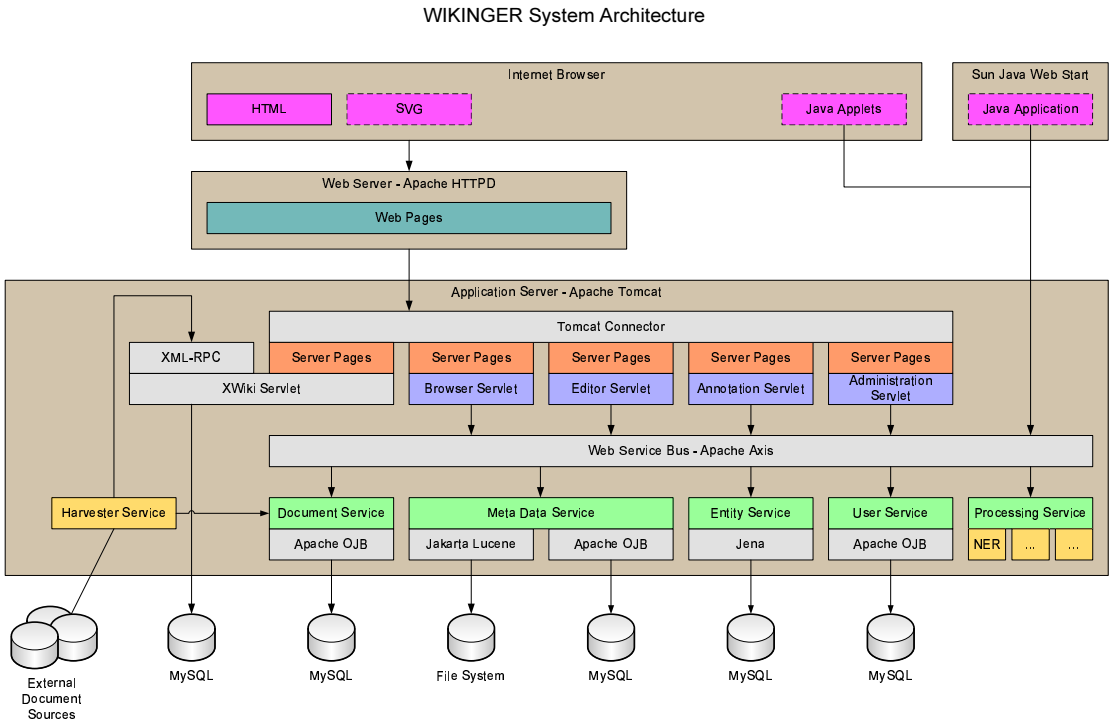


Abbildung 6.3: WIKINGER-Systemarchitektur

auch hier werden die Entwickler von der Speicherschicht gekapselt.

Der Metadata Service greift auf zwei verschiedene Speichermedien zurück. Da ist zum Einen das bereits beschriebene Repository, zugegriffen über OJB, zum Anderen ist ein Volltextindex der enthaltenen Texte integriert, der über die Apache-Bibliothek *Lucene* [3] zugegriffen wird. Der Index ist im Dateisystem des Servers abgelegt und ermöglicht die Suche nach Schlagworten im Korpus. Verbunden mit den Dokumentenformaten des Systems lassen sich so zielgenau die Absätze ermitteln, die zu einer Anfrage passende Inhalte aufweisen.

Die in orange eingezeichneten Dienste sind optionale Erweiterungen eines WIKINGER-Systems. Dazu gehören die in der letzten Abbildung bereits eingeführten Processing Services ebenso wie der in Abbildung 6.3 enthaltene *Harvester Service*. Dieser “Ernter“ kann über eine kleine GUI auf einem Client ausgeführt werden und importiert alle Dateien aus einem lokalen Verzeichnis in ein WIKINGER-System. Die Entscheidung über Notwendigkeit bzw. Implementierung der orange markierten Dienste ist stark anwendungsgetrieben, daher ist dieser Teil des Systems erweiterbar angelegt, d.h. solche Dienste können noch nachträglich in das System integriert werden und auf bereits vorhandene Dienste zugreifen. Beispielsweise greift der Harvester Service intern auf Funktionalitäten des Document Service zu, ist insofern nur eine nutzerfreundlichere Schnittstelle zum Import.

Alle Dienste eines WIKINGER-Systems greifen auf einen Web Service Bus zu, der mit Apache Axis realisiert ist. Axis [2] ist eine Implementierung von SOAP, einer Recommendation des W3C zur Beschreibung des Zugriffs auf Web Services [43, 53, 54]. Der Web Service Bus ist die Schnittstelle, über die alle Services miteinander kommunizieren und über die alle Anfragen von außen an die zuständigen Services verteilt werden.

Oberhalb davon sind die verschiedenen möglichen Applikationsarten mit dem Weg ihrer Zugriffe eingezeichnet; serverbasierte in blau und client-basierte in lila. Servlets zur Realisierung web-basierter dynamischer Applikationen fallen in die erste, über Java Web Start oder auf einem Client installierte Applikationen in die zweite Kategorie. In der Abbildung enthalten ist bereits das im WIKINGER-Pilotprojekt verwendete Wiki-System XWiki, das ebenso wie WIKINGER als Webapplikation in einem Tomcat Server abläuft. Tomcat kann wie abgebildet über einen Connector in einen Apache Webserver integriert werden, so dass der Webserver die statischen Seiten und Tomcat die dynamischen verwaltet und ausliefert. Schließlich ist im Browser-Kasten noch eine mögliche graphische Visualisierung semantischer Netze angezeigt, nämlich mittels SVG (Scalable Vector Graphics), einer weiteren Empfehlung des W3C [48].

6.2 WIKINGER-Komponenten mit Dissertationsbezug

Die ausführliche Beschreibung sämtlicher Komponenten, die zum Kern der WIKINGER-Plattform gehören, würde den Rahmen dieser Dissertation sprengen. Daher werden nachfolgend lediglich diejenigen Komponenten genau beschrieben, die konkret für die Erfüllung der Ziele dieser Dissertation benötigt werden. Bis auf die Komponenten in Abschnitt 6.2.3 und 6.2.4 sind sie unter der Ägide des Autors entstanden.

6.2.1 Harvester Service

Die Fähigkeit zur Verarbeitung verschiedener Dokumentformate ist eine der Anforderungen, die in Kapitel 2 definiert wurden. Diese Anforderung besteht auch im Rahmen des WIKINGER-Projekts. Gleichzeitig benötigen die Services des Systems verlässliche Dokumentstrukturen, auf denen sie arbeiten können.

Um beiden Anforderungen gerecht werden zu können, wird im Projekt wie folgt vorgegangen: Intern werden verschiedene Dokumentenklassen definiert, die für verschiedene Einsatzzwecke verwendet werden können. Für den Import von Bildern, Videos oder Audiobeiträgen, die momentan alle nicht weiter ausgewertet sondern lediglich angezeigt werden, gibt es ein Format, *MIMEDocument*, das die Speicherung der Binärdaten, des MIME-Typen der Datei und evtl. nützlicher Metadaten, letztere als frei wählbare Key-Value

Paare, erlaubt. Zusätzlich gibt es ein XML-Format speziell für Textdateien, das *PageDocument*. Darin werden Texte in ihre groben Bestandteile (Seiten, Seitenüberschriften, Absätze, Fußnoten, Tabellen) zerlegt abgespeichert. Diese sind können alle über eine ID separat zugegriffen werden, behalten aber die Informationen über ihren Ursprung (als doppelt verketteter Baum). Diese Bestandteile werden in eine interne Suchmaschine eingefüllt und somit von anderen Services aus gezielt suchbar.

Für den Import von Dokumenten ist der *Harvester-Service* zuständig, der bereits in 6.1.4 erwähnt worden ist. Soll eine Datei importiert werden, muss sie beim Import in eines der passenden Formate konvertiert werden. Für Binärformate ist dies einfach, ihre Binärdaten werden einfach im Containerformat gekapselt. Bei Textdaten ist der Vorgang komplizierter, denn diese müssen in das erwartete XML-Format transformiert werden. Minimal kann das erreicht werden, indem ein neues einseitiges PageDocument angelegt wird, in dessen einzigen Absatz der ganze Text kopiert wird. Das ist allerdings für die Verarbeitung hinderlich; insofern ist es angebracht, hier mehr Aufwand zu treiben. Dafür ist die Programmierung von Transformatoren nötig. Für drei Formate gibt es bereits solche Transformatoren:

Rohtext Dieses Format ist heutzutage in Dateiform nicht mehr so häufig anzutreffen, allerdings wird dieser Transformator auch zur Umwandlung von Wiki-Artikeln in das interne Format verwendet. Da sich eine Einteilung in Seiten hier nicht vornehmen lässt, wird der Text lediglich entlang von Doppelzeilenumbrüchen in Absätze unterteilt.

PDF Dieses Format hat sich in den letzten Jahren zum Industrie-Standard für den Austausch von Textdokumenten entwickelt. Sofern ein PDF-Dokument tatsächlich Text enthält und nicht verschlüsselt ist, kann der Harvester-Service es importieren und in ein PageDocument überführen. Aufgrund der Strukturinformationen innerhalb des PDF kann der Text in Seiten zerteilt werden, so dass Texte analog zur Seitennummerierung des Originals verarbeiten lassen.

FineReader-XML Hierbei handelt es sich um ein Ausgabeformat der OCR-Software FineReader der Firma Abbyy. FineReader ist das marktführende Produkt auf dem Gebiet der OCR in Textdokumenten und wurde in der Vergangenheit auch erfolgreich bei schlechten Vorlagen eingesetzt (z.B. bei der Retrodigitalisierung des Zeitungsarchivs der Neuen Zürcher Zeitung). Neben der Zeichenerkennung liefert FineReader auch Layout-Informationen über das eingelesene Dokument, zum Beispiel Kopf- und Fußzeilen, Fußnoten und Fußnotenindikatoren im Fließtext. Das Ausgabeformat lässt sich direkt in PageDocumente übertragen. Dabei bleiben Layout-Informationen erhalten.

Zur Verwendung des *Harvester Service* ist eine GUI geschrieben worden, in der Verzeichnisse oder einzelne Dateien ausgewählt werden können, die anschließend zum WIKINGER-

Server übertragen werden. Die Schnittstelle ist aber auch ohne GUI ansprechbar, etwa zur Batch-Verarbeitung größerer Verzeichnisbäume.

6.2.2 Weitere Importschnittstellen

Der in Kap. 6.2.1 beschriebene Harvester erlaubt den Import von Dokumenten in das System. Darüber hinaus gibt es aber noch andere Quellen, von deren Import das System profitieren kann. Konkret handelt es sich dabei um Datenbanken und bereits existierende Informationen in OWL bzw. RDF. Für diese Datenarten sind andere, spezialisierte Importmechanismen zuständig. Diese werden im Anschluss näher erläutert.

Relationale Datenbanken

Relationale Datenbanken eignen sich aufgrund ihres hohen Strukturierungsgrades besonders für den Import in eine Ontologie. Zeilen einer Tabelle definieren zusammengehörige Daten und die einzelnen Felder einer Zeile lassen sich sehr gut als Tripel ausdrücken, da sich der Schlüssel der Zeile, die Spalte und der Wert des betreffenden Feldes direkt in Subjekt, Prädikat und Objekt einer RDF-Aussage abbilden lassen.

Allerdings gibt es keine automatisierten Transformer für relationale Datenbanken. Dies, obwohl das ER-Schema einer Datenbank eigentlich eine Ontologie beschreibt. Das Problem liegt denn auch nicht in der Art der Daten, sondern im Matching zwischen der in der Datenbank vorliegenden Ontologie und der Zielontologie. Ontologien sind allgemeine Graphen, insofern ist das Problem der Komplexitätsklasse NP zuzuordnen - eine automatische Lösung ist daher nicht zu erwarten.

Darum wird ein Tool entwickelt, das den Prozess des Imports automatisiert, das Matching allerdings in menschliche Hände legt. Das Tool ist eine Java-Applikation, die entweder auf der Datenbank selbst, oder aber auf einem CSV-Dump der einzelnen Tabellen arbeitet. Den verschiedenen Spalten der zu importierenden Tabellen müssen lediglich Konzeptklassen und Relationen zugeordnet werden, in die die Werte zu übertragen sind. Es ist nicht geplant, dieses Tool jedem Anwender zur Verfügung zu stellen; das Einspielen großer Datenmengen aus einer Datenbank soll den Betreibern einer WIKINGER-Instanz vorbehalten bleiben, da von ihnen die notwendige Sorgfalt bei der Erstellung des Mappings zwischen Datenbank und semantischem Netz am ehesten erwartet werden kann.

OWL/RDF-Daten

Für verschiedene Domänen existieren bereits Daten in OWL oder RDF. Um diese innerhalb der Ontologie verwenden zu können, gibt es eine eigene Schnittstelle, über die ein Import in die interne Ontologie möglich ist. Für einen Import eignen sich besonders taxonomische

Daten, etwa Organigramme, Rangsysteme oder geographische Lagebeziehungen betreffend. Mit diesen Informationen kann die Strukturierung der automatisch generierten Ontologie unterstützt werden.

Um den größten Nutzen aus dem Import von Strukturdaten zu ziehen, empfiehlt es sich, die semantischen Informationen mit dem gleichen Basis-URI wie dem der internen Ontologie zu versehen. So ist sichergestellt, dass die Zusatzinformationen mit bestehenden Knoten verschmolzen werden können.

Neben strukturierenden Informationen, die eher auf der Klassenebene wirken, können natürlich auch neue oder zusätzliche Instanzdaten in die Ontologie eingefügt werden. Diesen Importweg können zusätzliche Datenpflege-Applikationen verwenden, um fehlende Angaben in der Ontologie zu ergänzen oder fehlerhafte Angaben zu korrigieren.

Es ist nicht vorgesehen, den Import von OWL/RDF-Daten externen Nutzern zu ermöglichen, diese Importwege stehen nur den Betreibern einer WIKINGER-Instanz offen. Das Einfügen weiterer strukturierender Informationen, die sich auf das ganze Netz auswirken, sollten an einer zentralen Stelle durchgeführt werden, schon um Doppelarbeiten innerhalb der Community zu vermeiden, aber insbesondere, um das Gesamtsystem sicherer gegen Manipulationen böswilliger Dritter zu machen.

6.2.3 WALU

WALU steht für WIKINGER Annotations- und Lernumgebung. WALU ist eine Entwicklung der Universität Duisburg-Essen[9], die eine wichtige Rolle im Projekt spielt, denn bei WALU handelt es sich um eine Java-Applikation, mit der Domänenexperten einfach Beispiele für Konzeptklassen in Textdokumenten annotieren können [24, 111].

Abbildung 6.4 zeigt die dazu verwendete Arbeitsfläche. Der Text wird in einem großen Editorfenster angezeigt, darunter sind Schaltflächen zu sehen, die mit den Namen der im aktuellen Projekte verwendeten Konzeptklassen beschriftet sind, jede in einer anderen Farbe. Eine Annotation wird dadurch in das Dokument eingefügt, dass der zugehörige Text im Editor markiert und anschließend die Schaltfläche der entsprechenden Konzeptklasse gedrückt wird. Dadurch wird der Text mit der passenden Farbe im Editor hervorgehoben, so dass auf Anhieb zu erkennen ist, welcher Klasse eine Annotation zugehörig ist. Diese einfache Textmarker-Metapher ist sehr einfach zu erklären, wodurch schnell in die Arbeit mit dem Tool eingestiegen werden kann.

Im linken Bereich der Abbildung ist eine Baumansicht zu sehen, in der die bereits markierten Entitäten zu sehen sind, die Farbe des Kreises davor gibt Auskunft über den Typ, die Art des Kreises zeigt an, ob es sich um eine gesicherte Annotation (ausgefüllt) oder um eine automatische Annotation handelt, die noch nicht begutachtet wurde (nicht

[9]Siehe www.uni-due.de/computerlinguistik/walu.shtml

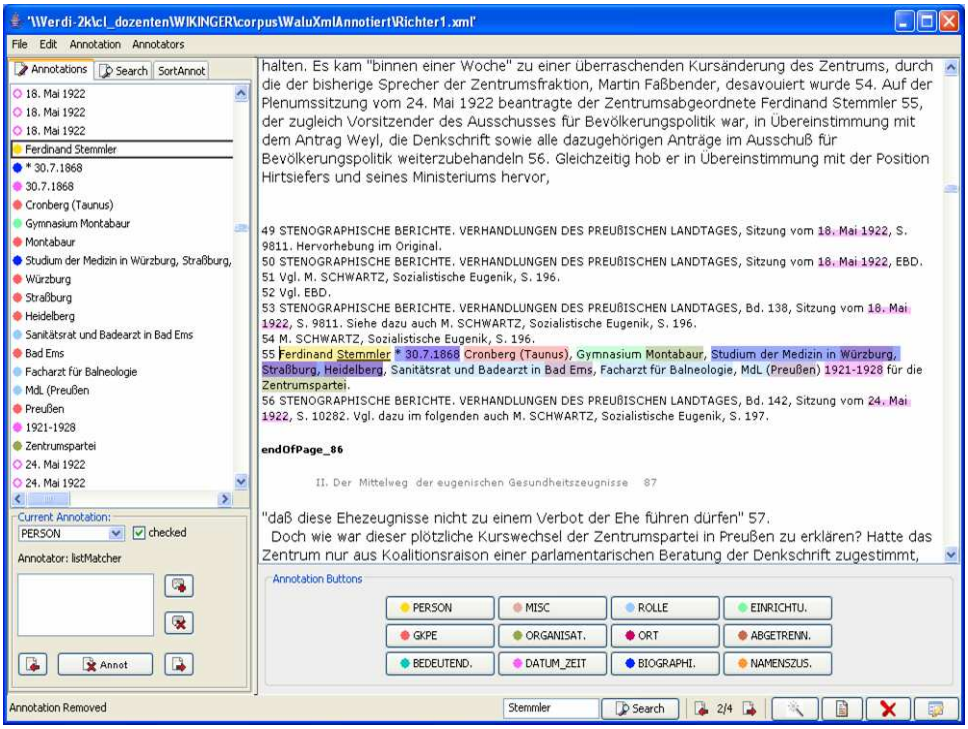


Abbildung 6.4: Screenshot von WALU

ausgefüllt). WALU ist in der Lage, auf der Basis bereits annotierter Entitäten weitere Vorkommen automatisch zu markieren. Desweiteren ist eine Datumsannotierung mit regulären Ausdrücken integriert.

6.2.4 Annotationsserver

Der Annotationsserver, ebenfalls ein Beitrag der Universität Duisburg-Essen, ist für die automatische Extraktion von Entitäten aus dem Korpus zuständig. Für die verschiedenen semantischen Klassen werden Erkenner auf den Beispielen trainiert, die im WALU-Client von den Experten annotiert worden sind. Die automatische Auswertung des Korpus kann anschließend über den Web Service angestoßen werden.

Seine Daten speichert der Annotationsserver in einer eigenen Datenbank und nicht im Metadata-Repository. Dies geschieht entgegen der generellen Architektur, allerdings lässt sich so die Performanz des Servers stark steigern, da die benötigten Daten in direktem Zugriff sind und nicht erst über den Metadata Repository Service angefordert werden müssen. Der Annotationsserver übernimmt daher auch die Aufgaben, die mit dem Zugriff auf die erstellten Annotationen zusammenhängen. Diese können über eine Web Service Schnittstelle vom Annotationsserver abgerufen werden.

6.2.5 Semiautomatische Relationserkennung

Der Ausgangspunkt für die Relationserkennung ist eine Menge größtenteils automatisch erzeugter Annotationen zu Fußnotenelementen, die Biogramme enthalten. Dadurch ist eine grundlegende Zuordnung der Entitäten der Fußnote zu der Person gegeben, die in der Fußnote beschrieben wird, d.h. diese stehen alle in einem (zunächst unbekannten) Verhältnis zu ihr. Die Struktur der Biogramme ist stark formalisiert, d.h. sie bestehen aus einer Abfolge biographischer Ereignisse, deren Entitäten jeweils direkt miteinander in Beziehung stehen, mit den Entitäten anderer biographischer Ereignisse jedoch nicht. Dadurch besteht das eigentliche Problem darin, zunächst die Grenzen der einzelnen biographischen Ereignisse und anschließend die Relationen zu erkennen, die in den jeweiligen Ereignissen ausgedrückt sind. Zur Veranschaulichung hier ein Beispiel für ein Biogramm:[10]

> [96] Johannes M. Cramer, geb. 25. April 1914, Priesterweihe 1939, 1940 Vikar in Wattenscheid, dann Wilndorf, 1942 Pfarrvikar in Atzendorf, 1946 Pfarrvikar in Wolmirstedt, 1948 Vikar in Stendal, 1953 Vikar in Genthin, 1955 Pfarrer in Burg, 1960 Dechant im Dekanat Burg, 1967 Pfarrer in Halle/St. Norbert, 1968 Pastoralreferent im Dekanat Halle, 1976 Dechant für das Dekanat Halle, 1979 Geistlicher Rat, 1981 Ruhestand.

Hier zeigt sich der starre Aufbau sehr deutlich. Jede Datumsangabe kann zum Abtrennen eines biographischen Ereignisses verwendet werden, das Problem der Identifikation der Segmente innerhalb des Biogramms lässt sich also mit einfachen Heuristiken zufriedenstellend lösen. Der nächste Schritt ist allerdings deutlich schwerer automatisch zu erledigen. Der Großteil der Ereignisse dieses Beispiels behandelt zwar die gleiche Relation, nämlich die Übernahme bestimmter Rollen an bestimmten Orten zu einer bestimmten Zeit, allerdings weisen die Biogramme, betrachtet über die Gesamtheit des Korpus zu viele stilistische Unterschiede auf, als dass sich alle Biogramme mit Heuristiken dieser Art verarbeiten ließen. Betrachtet man z.B. Biogramme anderer Autoren, zum Beispiel dieses hier:[11]

> [2] WALTER ARNOLD ABRAHAM DIRICHLET (1833-1887); Studium der Rechte in Berlin; Gutsbesitzer; 1877-1879, 1880-1886 Mitglied des preußischen Abgeordnetenhauses (Deutsche Fortschrittspartei, Deutsche Freisinnige Partei); 1881-1887 Reichstagsabgeordneter.

so stellt man schnell gravierende Unterschiede fest. Die Ansetzung der Namen, Daten und der Ereignisse ist anders, auch die Ausformulierung ist unterschiedlich. Befristete

[10] aus Reinhard Grütz: *Katholizismus in der DDR-Gesellschaft 1960-1990*, Blaue Reihe B - Forschungen, Band 99, Schöningh Verlag, 2004.

[11] aus Hans-Georg Aschoff: *Ludwig Windthorst, Briefe 1881-1891*, Blaue Reihe A - Quellen, Band 47, Schöningh-Verlag, 2002

Funktionen sind deutlich gekennzeichnet, fortlaufende Funktionen ("Gutsbesitzer") hingegen werden ohne Datum genannt. Zudem sind die Ereignisse deutlich komprimierter dargestellt.

Angesichts der stark formalisierten Domäne könnte man auf die Idee kommen, Regeln zur Bestimmung der Relationen aufzustellen. Dieser Ansatz erfordert jedoch eine intensive Beschäftigung mit dem Material, also letztlich eine Durchsicht aller Fußnoten, um sicherzustellen, dass die Regeln tatsächlich den Datenbestand hinreichend abbilden. Das ist, mit Blick auf die Menge an Material, nicht realistisch. Darüber hinaus ist zu berücksichtigen, dass die gleichen Regeln später auch auf Texte angewendet werden sollen, die von Nutzern im Wiki erzeugt werden. Spätestens dort muss ein Algorithmus versagen, der auf Aufzählen der erlaubten Relationen beruht. Daher ist ein Vorgehen, das keine weitergehenden Annahmen über die Inhalte der Fußnoten trifft, wesentlich tragfähiger mit Blick auf die Gesamtmenge der Texte.

Daher ist zur Relationserkennung der in Kapitel 5.2.6 vorgestellte Ansatz umgesetzt worden. In den nachfolgenden Abschnitten werden die einzelnen relevanten Schritte beschrieben und Abweichungen vom theoretischen Ansatz motiviert. Der gewählte Ansatz ist in einer Veröffentlichung für den ACM-Workshop *Semantic Authoring, Annotation and Knowledge Markup (SAAKM07)* in Whistler, Kanada beschrieben worden [24].

Gewinnung von Assoziationsregeln

Dazu wurde ein Algorithmus zum Erstellen von Assoziationsregeln nach dem apriori-Algorithmus von Agraval und Srikant [90] implementiert[12]. Als Eingangsdaten dienen automatisch aus dem Korpus extrahierte Entitäten, die durch den WIKINGER-Annotationsserver, eine Entwicklung der Universität Duisburg-Essen, zur Verfügung gestellt werden. Dabei werden eingangs nur die Annotationen verwertet, die in den in den Fußnoten anzutreffenden Biogrammen erzeugt worden sind. Das WIKINGER-System erlaubt über das PageDoc-Format den gezielten Zugriff sowohl auf die Fußnoten jedes Dokuments aus dem Korpus, als auch auf die dazu gehörenden Annotationen vom Annotationsservice. Zur Erkennung der Biogramme wird eine Heuristik eingesetzt, die aufgrund der Anzahl und der Zusammensetzung der Entitäten abschätzt, ob es sich um ein Biogramm handeln kann oder nicht.

Die Biogramme werden in Sinnabschnitte unterteilt, die jeweils einen Fakt aus dem Leben der betreffenden Person umfassen. Die Sinnabschnitte dienen als Eingabe für den apriori-Algorithmus, die in ihnen jeweils enthaltenen Mengen von Entitätstypen werden im Algorithmus ausgewertet. Die beiden Parameter *support* und *confidence* können dabei frei variiert werden. Die durch den Algorithmus gewonnenen Assoziationsregeln stehen als Java-Objekte mitsamt den zugehörigen Vorkommen zur weiteren Verarbeitung im System zur Verfügung.

[12]Eine Beschreibung des Algorithmus findet sich in Kapitel 3.3.2

Clustering

Auf der Basis der Assoziationsregeln werden anschließend die Relationscluster erstellt. Die Eingangsdaten für den Clustervorgang sind die Vorkommen der gerade aktiven Assoziationsregel. Im Ursprungstext dieser Abschnitte werden die Entitäten gegen ihre Entitätstypen ausgetauscht, so dass generalisierte Muster entstehen. Aufgrund der speziellen Form der Fußnoten wird auf eine Stoppwortentfernung verzichtet. Die so geänderten Texte werden mit Apache Lucene indexiert, um an die Wortvektoren zu gelangen, auf denen sich das tf*idf-Maß berechnen lässt.

Im eigentlichen Clusterschritt kommt agglomeratives Clustering zum Einsatz, unter Verwendung der Cosinus-Ähnlichkeit als Distanzfunktion und der Single oder der Complete Linkage Distance als Cluster-Kriterium. Eine Mindestgröße für die Cluster kann dem Verfahren als Parameter vorgegeben werden.

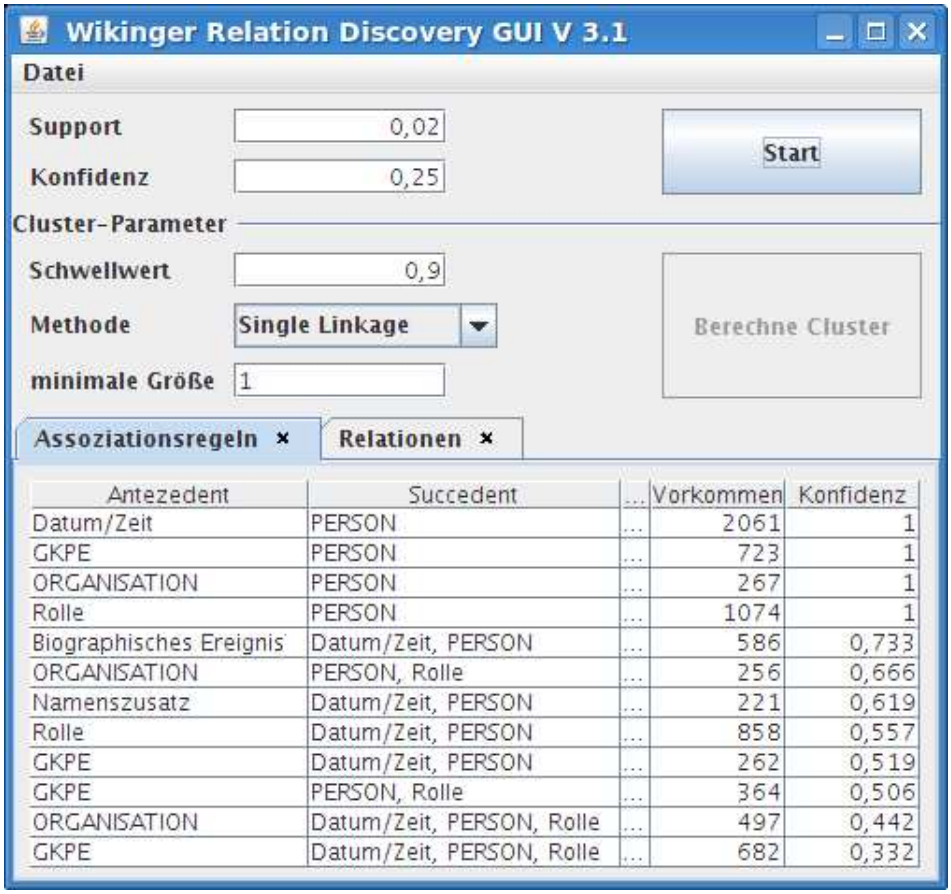
Das Resultat dieses Schritts ist eine Menge von Clustern, die Kandidaten für verschiedene Relationen im Rahmen der aktiven Assoziationsregel sind. Aufgrund der Struktur der Daten im Pilotprojekt ist eine automatische Bestimmung der Relationen nicht möglich, daher müssen die Label manuell bestimmt werden. Dazu bietet das WIKINGER-System die WIKINGER Relation Discovery Gui an, die im nachfolgenden Abschnitt beschrieben wird.

WiReD Gui - WIKINGER Relation Discovery Gui

Das WIKINGER Relation Discovery Gui ist eine Java-Applikation, die zur Steuerung des Prozesses der Relationserkennung dient. Der erste Schritt dabei ist die Auswahl der zu betrachtenden Texte, für deren Verarbeitung durch den apriori-Algorithmus die Parameter eingestellt werden können.

Abbildung 6.5 zeigt die Standardansicht des Programms. Im oberen Teil des Fensters können die Parameter für den apriori-Algorithmus eingegeben werden, in diesem Fall 0,02 für den Support und 0,25 für die Konfidenz. Die dabei entstehenden Assoziationsregeln werden tabellarisch im unteren Teil des Fensters in einem Reiter angezeigt, in Abbildung 6.5 ist er gerade ausgewählt. Die Regeln werden mit Angabe von Antezedent, Sukzedent, der absoluten Anzahl der Vorkommen und der Konfidenz angezeigt. Nach jeder der Spalten kann auf- und absteigend sortiert werden. Im mittleren Teil des Fensters können die Parameter für den Clustering-Schritt ausgewählt und gesetzt werden. Dazu ist es vorher nötig, eine der Assoziationsregeln zu markieren.

Abbildung 6.6 zeigt das Ergebnis eines Clusterdurchgangs für eine der Regeln, in diesem Fall für die Regel $Organisation \rightarrow Person, Rolle$, unter Verwendung des Single Linkage Clusterings, mit einem Schwellwert von 0,9 für die Ähnlichkeit und einer Mindestgröße der einzelnen Cluster von 3. Für jeden Clustervorgang wird ein eigener Reiter angelegt, der mit dem Regelnamen, dem verwendeten Kriterium und dem Ähnlichkeitsschwellwert markiert ist. Auf der linken Seite sind die verschiedenen Relationskandidaten zu sehen, auf der



Antezedent	Succedent	...	Vorkommen	Konfidenz
Datum/Zeit	PERSON	...	2061	1
GKPE	PERSON	...	723	1
ORGANISATION	PERSON	...	267	1
Rolle	PERSON	...	1074	1
Biographisches Ereignis	Datum/Zeit, PERSON	...	586	0,733
ORGANISATION	PERSON, Rolle	...	256	0,666
Namenszusatz	Datum/Zeit, PERSON	...	221	0,619
Rolle	Datum/Zeit, PERSON	...	858	0,557
GKPE	Datum/Zeit, PERSON	...	262	0,519
GKPE	PERSON, Rolle	...	364	0,506
ORGANISATION	Datum/Zeit, PERSON, Rolle	...	497	0,442
GKPE	Datum/Zeit, PERSON, Rolle	...	682	0,332

Abbildung 6.5: Assoziationsregeln in WiReD Gui

rechten Seite die Muster, die zu einem markierten Kandidaten gehören. Auf Knopfdruck kann zwischen der Musteransicht und einer Detailansicht gewechselt werden, in der die Urtexte der Muster zu sehen sind. Abbildung 6.7 zeigt diese Ansicht.

Im unteren Teil des Fensters sind verschiedene Schaltflächen angeordnet, die zur Nachbearbeitung der Kandidaten dienen. Damit lassen sich gleichartige Cluster zusammenfassen, umbenennen oder löschen. Abbildung 6.8 zeigt die Kandidaten für die o.g. Regel nachdem die Nachbearbeitung abgeschlossen wurde. Die Kandidatenmenge wurde vom Experten auf die vier übrig gebliebenen reduziert, zur besseren Übersicht wurden sie von ihm auch benannt. Diese Benennung dient nur als Hilfestellung für die Domänenexperten, sie hat keine Auswirkung auf das semantische Netz, weswegen hier auch Leer- und Sonderzeichen verwendet werden können, die in RDF maskiert werden müssten, um den Anforderungen von XML zu genügen.

Ist die Menge der Relationskandidaten gefunden, sind die Arbeiten für eine Regel fast abgeschlossen. Was bleibt, ist die Benennung der Eigenschaften, die in RDF die einzelnen Entitäten mit dem Relationsobjekt verbinden werden. An dieser Stelle setzt normalerweise

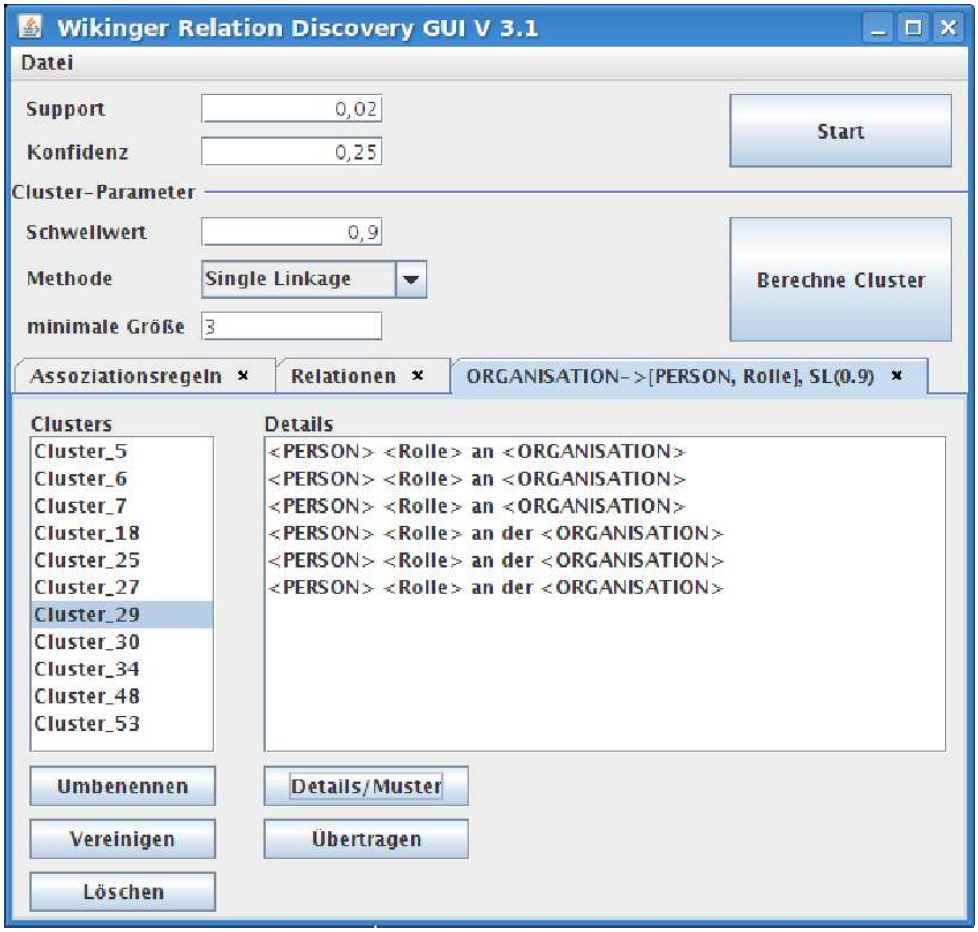


Abbildung 6.6: Musteransicht eines Clusters in WiReD Gui

das semiautomatische Labeling an, das z.B. im Text vorkommende Verben verwendet. Aufgrund der Struktur der Fußnoten im Pilotprojekt ist das hier nicht möglich, weswegen die Domänenexperten zumindest für die Verbindung der Person zum Relationsobjekt ein Label manuell vergeben müssen. Dies geschieht unter Verwendung des Dialogs, der in Abbildung 6.9 gezeigt wird. Der oberste Relationsbezeichner muss manuell eingegeben werden, die übrigen sind mit einem Label vorbelegt, das bei Bedarf aber auch geändert werden kann.

Sobald die Domänenexperten alle Relationskandidaten aus den Assoziationsregeln ausgewählt haben, die sie in das semantische Netz integrieren möchten, können diese zum WIKINGER-Server geschickt werden, wo sie zur Relationsklassifikation eingesetzt werden können.

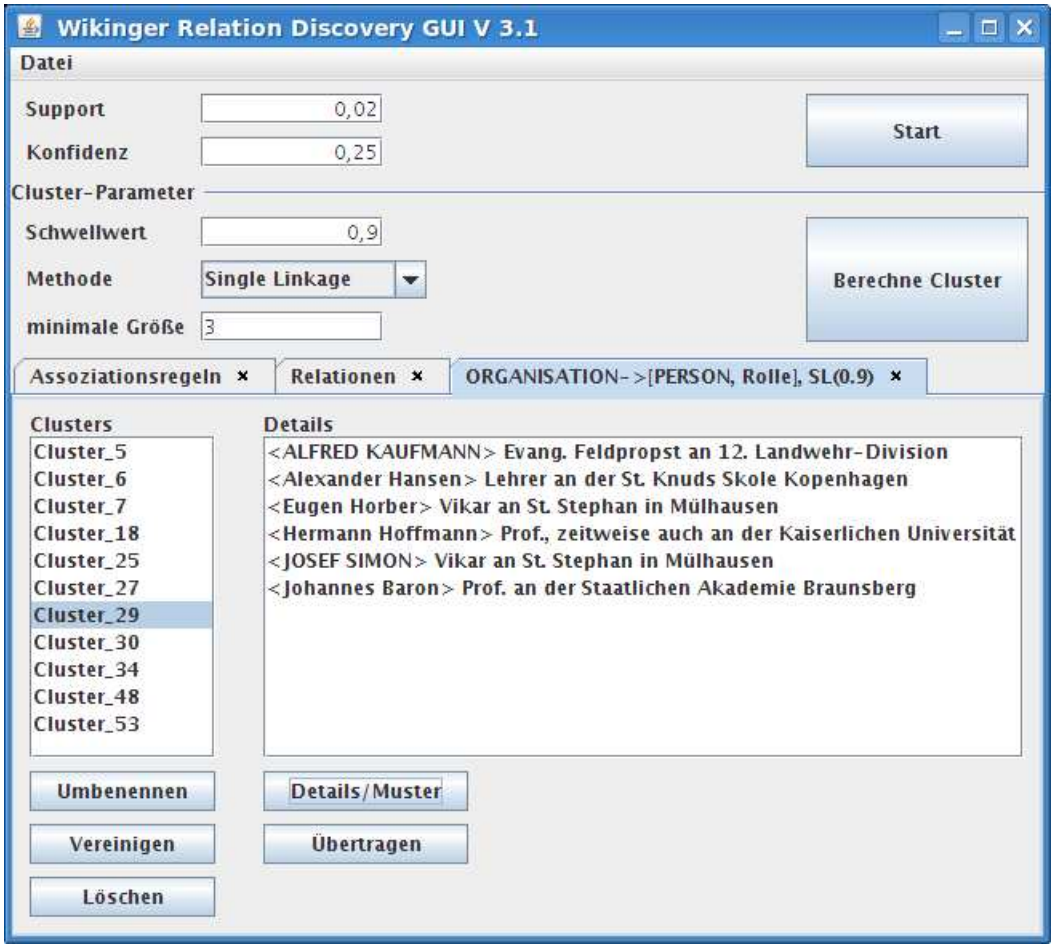


Abbildung 6.7: Detailansicht eines Clusters in WiReD Gui

Relationsklassifikation

Die von den Experten ausgewählten Relationen werden auf dem WIKINGER-Server dazu verwendet, die Relationskandidaten im restlichen Dokumentenbestand zu klassifizieren. Dazu wird ein eigener Web Service verwendet. Beim Aufruf erhält der Service eine Liste der zu bearbeitenden Dokumente, deren Annotationen vom WIKINGER Annotationsserver zur Verfügung gestellt werden. Diese werden in den gleichen Schritten vorverarbeitet wie die manuell annotierten Dokumente, d.h. sie werden segmentiert und durch den apriori-Algorithmus verarbeitet. Von den dort erstellten Assoziationsregeln werden nur diejenigen weiter berücksichtigt, die auch in den von den Experten ausgewählten Relationen vorkommen.

Die eigentliche Klassifikation verwendet ein ähnliches Verfahren wie das in Abschnitt 6.2.5 beschriebene. Zu jeder Assoziationsregel, die den manuell bestimmten Relationen zugrunde liegt, wird die passende Assoziationsregel aus den Kandidaten bearbeitet. Deren

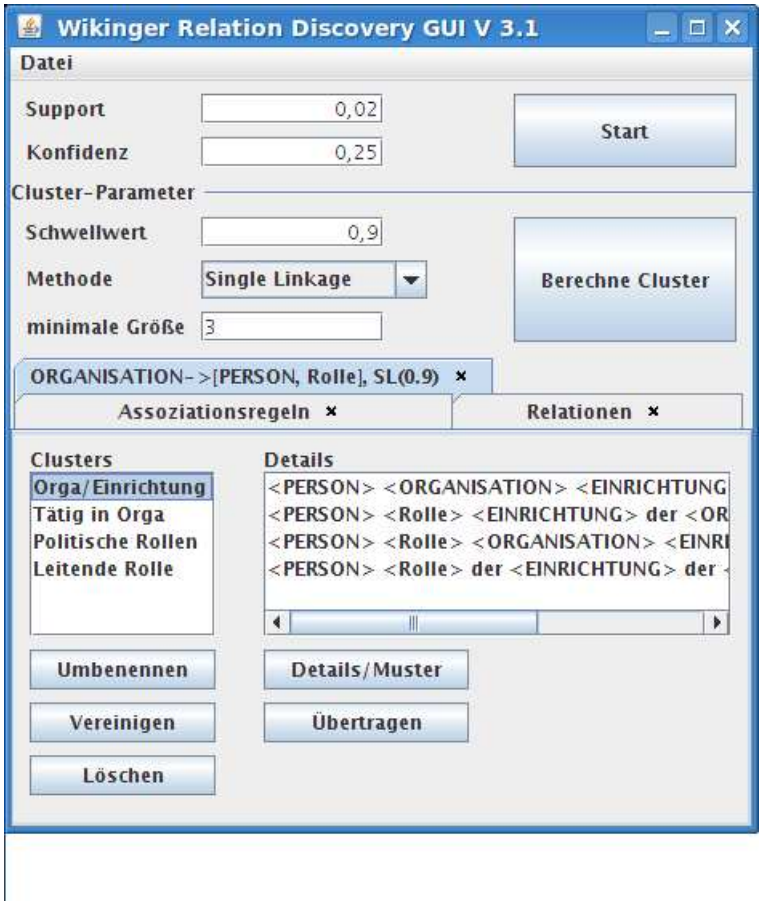


Abbildung 6.8: Nachbearbeitung der Kandidaten in WiReD Gui

Relationsvorkommen werden unter Verwendung von Lucene in Wortvektoren mit tf*idf-Gewichtung umgewandelt. Daran schließt sich eine Clusterphase an, die unter Verwendung der gleichen Schwellwerte und Distanzmaße durchgeführt wird, die auch bei der Erzeugung der Referenzrelationen zum Einsatz kamen.

Die so entstandenen Cluster werden mit den Relationsclustern verglichen und der Relation zugeschlagen, zu der sie die größte Ähnlichkeit aufweisen. Um Fehler zu reduzieren, wird hierbei allerdings ein (variabel gestaltbares) Mindestähnlichkeitsmaß vorausgesetzt. Als Ergebnis dieses Schritts sind diejenigen Relationsvorkommen klassifiziert, die vollautomatisch aus dem Dokumentenbestand extrahiert und Relationen aus der manuell erstellten Referenzmenge zugewiesen werden konnten.

6.2.6 Ontologieverwaltung

Die Ontologieverwaltung umfasst verschiedene Aspekte, nämlich die eigentliche Verwaltung der Ontologiedaten, die Erstellung der Ontologie aus den durch die Relationserkennung

Abbildung 6.9: Relationsbenennungsdialog in WiReD Gui

und der NER erzeugten Informationen, die Aktualisierung der Ontologie in Reaktion auf Änderungen im Datenbestand und schließlich die Verfügbarmachung für die verschiedenen Nutzungsdienste. Diese Arbeiten sind auf verschiedene Komponenten aufgeteilt, um die Wartung der einzelnen Bestandteile zu vereinfachen. In den folgenden Abschnitten werden die Bestandteile dieses Moduls vorgestellt.

Verwaltung des Ontologie

Die Pläne für die Referenzarchitektur sehen für die Verwaltung der Ontologie einen eigenen Service vor, der als Schnittstelle zwischen Ontologie und dem restlichen System dient. Diese Trennung entspricht den Kapselungsbestrebungen, die in der objekt-orientierten Softwareerstellung verfolgt werden. Der Dienst bietet dabei eine Schnittstelle an, über die jeglicher Zugriff auf die Ontologie zu erfolgen hat. Der Vorteil davon liegt darin, dass Entwickler nicht für Eigenheiten der unterliegenden Implementierungen, sondern anhand einer implementierungsunabhängigen Schnittstelle programmieren. Dadurch wird eine enge Koppelung der Ontologieverwaltung an die auf die Ontologie zugreifenden Applikationen verhindert und die Wartbarkeit des Gesamtsystems vereinfacht.

In der Praxis macht sich an dieser Stelle jedoch die Schnittstellenproblematik der Web Services eklatant bemerkbar: Die Übertragung von Objektreferenzen über Web Services ist nicht möglich. Dies ist für die verschiedenen Module innerhalb der Ontologieverwaltung ein derart großer Nachteil, dass entschieden worden ist, darauf zu verzichten, die Verwaltungsschnittstelle der Ontologie als Web Service zu implementieren.

Die Ontologie wird über Jena in einer relationalen Datenbank vorgehalten. Service-Entwickler müssen sich nicht mit den Details der Speicherung auseinandersetzen, sie erhalten ein Java API, über das sie Zugriff auf das RDF-Modell erhalten können. So sind die Implementationsdetails des Modells immer noch gekapselt und es sind nur die Zugriffsmethoden möglich, die auch im Java-Interface definiert sind. Der einzige Verlust durch diese Vorgehensweise besteht in einer engen Koppelung an Jena als Ontologie-Engine.

Angesichts des Funktionsumfangs, der weiten Verbreitung und der Tatsache, dass das

Paket unter einer Open-Source-Lizenz verfügbar ist, sind die Risiken der Koppelung zu vernachlässigen. Höchstens im Fall eines Wechsels der Ontologie-Engine sind größere Aufwände zu leisten, um die API zu ändern und die Dienste anzupassen, die darauf zugreifen. Dabei handelt es sich allerdings lediglich um die hier aufgeführten Dienste, direkter Zugriff auf die Ontologie wird externen Diensten nicht gewährt.

Erstellung des Netzes

Der Dienst zur Netzerstellung setzt auf den Ergebnissen der Relationsklassifikation und des Annotationsservers auf. Zunächst werden im Modell die semantischen Klassen und die verschiedenen Relationen angelegt, ehe die Daten zu spezifischen Instanzen und deren Relationen verarbeitet werden, die aus der Relationklassifikation stammen.

Die Integration der Instanzen der verschiedenen semantischen Klassen wird über eine Parameter-Datei gesteuert. In ihr ist verzeichnet, in welcher Form die Instanzen in das Netz eingetragen werden sollen, ob als Ressourcen oder als Literale. Letzteres eignet sich besonders für Datumsangaben und Zahlenwerte. Soll eine Instanz als Literal eingepflegt werden, so findet sich in der Datei der XML-Datentyp, der diesen Literalen zugeordnet werden soll.

Im Fall, dass Instanzen als URI hinzugefügt werden sollen, wird folgendermaßen vorgegangen: Ressourcen werden über einen URI, einen unique resource identifier, identifiziert. Dieser URI setzt sich aus dem Basis-URI der Anwendung, gewissermaßen dem Host-Namen, und einem in diesem Basis-URI eindeutigen Schlüsselteil zusammen. Dieser Schlüssel wird vom Service erzeugt, es handelt sich dabei um einen so genannten Global Unique Identifier (GUID).

Dieses Vorgehen wird in der Welt der relationalen Datenbanken bereits seit langem praktiziert und vermeidet Probleme, die sich ergäben, verwendete man natürlichsprachliche Bezeichner als Ressourcen-Schlüssel: In Bezeichnern vorkommende Sonderzeichen wie Umlaute, Leerzeichen und Satzzeichen sind in URI immer noch problematisch, zusätzlich kann es verschiedene gleichnamige Entitäten gleichen Typs geben, die so nicht mehr unterschieden werden könnten.

Im Erstellungsprozess werden Daten aus verschiedenen Dokumenten miteinander kombiniert. Um zu verhindern, dass für jede Erwähnung einer Entität eine eigene Ressource erstellt wird, hält das System in einer Datenstruktur nach, welche Bezeichner verschiedener Typen bereits im Netz mit einem URI versehen worden sind. Damit verschiedene Entitäten gleichen Namens nicht irrtümlich kombiniert werden, kann dieser Teil des Dienstes um komplexere Unterscheidungsmerkmale ergänzt werden (die etwa neben dem Namen noch das Geburtsdatum berücksichtigen oder neben dem Ortsnamen noch das Bundesland).

Die Anlage komplexer Relationen folgt dem Ansatz in Kapitel 5.2.7, der Hilfskonstrukte verwendet. Hierbei wird für jedes Vorkommen einer Relation eine anonyme Ressource (eine

so genannte „blank node“) angelegt, die über die von den Experten festgelegte Relation mit dem jeweiligen Kopf der Relation verbunden ist. Die anderen Bestandteile der Relation werden über binäre Relationen mit standardisierten Namen (konfigurierbar über die Parameter-Datei) an diese anonyme Ressource angebunden. Zur Verdeutlichung sei noch einmal auf Abbildung 5.3 verwiesen.

Um die Herkunft der verschiedenen Relationen und Objekte nachhalten zu können, wird das Quelldokument ebenfalls in die Relationen eingearbeitet. Bei binären Relationen wird die Provenienz der Aussage als Reifikation ausgedrückt[13]. Damit lässt sich belegen, aus welchen Dokumenten bestimmte Behauptungen stammen. Wenn z.B. zwei Autoren für die gleiche Person unterschiedliche Geburtsdaten angeben, wird so die Entscheidung darüber, welche der Aussagen nun stimmt, vereinfacht.

Der Dienst zur Netzerstellung kann mit einzelnen Dokumenten, aber auch mit Teilmengen der Dokumentenbasis angesprochen werden.

Update der Ontologie

Der Update-Dienst ist Teil des Analyse-Dienstes. Er wird dann benötigt, wenn sich Dokumente geändert haben, die bereits Teil des Korpus und deren Daten bereits in die Ontologie eingeflossen sind. Um sicherzustellen, dass die Ontologie nicht veraltete Daten berichtet, müssen geänderte Dokumente daraufhin analysiert werden, ob sich Auswirkungen für die Ontologie ergeben. Für statische Korpora ist die Aktivierung dieses Dienstes nicht erforderlich, er ist jedoch unverzichtbar, wenn man es mit dynamischen Korpora zu tun hat (wie z.B. die Wiki-Texte in der Pilotanwendung).

Der Dienst arbeitet auf den Daten, die zu der neuen Revision des Dokuments vorliegen. Die in der Ontologie enthaltenen Daten, die auf die alte Revision zurückzuführen sind, werden entfernt, anschließend wird der Erstellungsdienst mit der aktuellen Revision getriggert, der dann eine Neuauswertung vornimmt und die Daten in die Ontologie einpflegt.

Es wäre denkbar, stattdessen eine Überprüfung der semantisch relevanten Unterschiede durchzuführen und nur die Änderungen einzupflegen, allerdings ist zu beachten, dass dieses Vorgehen wesentlich mehr Zeit beanspruchen würde und aus Gründen der Kontinuität trotzdem alle unverändert gebliebenen Aussagen mit dem Stempel der neuen Revision versehen werden müssten. Das *tabula rasa*-Vorgehen ist also tatsächlich ressourcensparender.

SPARQL-Server

Das WIKINGER-Framework bietet auch einen SPARQL-Server an, über den Anfragen an die Ontologie gestellt werden können. Dieser Server dient als Schnittstelle für alle Komponenten der Applikationsebene, die auf Daten der Ontologie arbeiten, kann aber auch

[13] Vgl. hierzu 5.2.7

nach außen zur Verfügung gestellt werden, wenn auch Applikationen neben der auf dem WIKINGER-Framework aufsetzenden auf die Ontologie zugreifen können sollen.

Als Server wird Joseki eingesetzt, die SPARQL-Implementierung des Jena-Projekts. Joseki läuft als Servlet in einem Java Application Server. Die Quell-Ontologie lässt sich in der Konfigurationsdatei des Servers festlegen, Joseki kann direkt auf die datenbankgestützte Persistierung der Ontologie zugreifen. Zur Durchsatzsteigerung ist die in Kapitel 5.3.2 beschriebene Schemavariante gewählt worden.

Die Nutzung des Servers durch externe Applikationen gefährdet nicht die Integrität der Ontologie, da SPARQL als reine Anfragesprache konzipiert ist und daher keine Funktionalitäten zur Datenmanipulation enthält. Allerdings sollte sichergestellt werden, dass keine Anfragen übergeben werden, die sich anderweitig als Sicherheitslücke ausnutzen lassen (etwa durch SQL-Injection). Dies kann dadurch geschehen, dass ein Web Service vor den Joseki-Server geschaltet wird, der die Anfragen im Vorhinein auf ihre Gültigkeit überprüft. Damit lässt sich auch eine Zugangskontrolle, bzw. eine Anfragenbeschränkung implementieren.

Zusammenfassung

Diese Komponenten bilden das Modul zur Verwaltung der Ontologie. Die Zugriffsmöglichkeiten sind dabei so verteilt wie in den Planungen in Kapitel 5.3.1 vorgesehen. Die Realisierung der für den tatsächlichen Zugriff auf die Ontologie benötigten Komponente wurde zur Effizienzsteigerung nicht als Service implementiert, die Kapselung der Funktionalitäten ist dennoch eingehalten worden.

Modifizierenden Zugriff auf die Ontologie haben lediglich solche Dienste, die das Rohmaterial analysieren, alle Dienste, die lediglich Daten zur Anzeige aufbereiten, müssen über den SPARQL-Server zugreifen, der lediglich Anfragen stellen, aber keine Daten verändern kann.

6.3 Automatische Ontologieerstellung

Im Rahmen der Arbeiten an semiautomatischen Verfahren zur Erstellung von Ontologien kamen die Fragen auf, welche Qualität Ontologien haben, die automatisch erzeugt worden sind und wofür sich solche Ontologien einsetzen ließen. Diesen Fragen widmet sich der folgende Ansatz, der sie anhand des Problems der Nachrichtenfilterung in einem dynamischen Umfeld untersucht.

Dank gebührt meinen Kollegen Stefan Paal und Dieter Strecker am Fraunhofer-Institut IAIS, die unterstützende Software zu diesem Ansatz beigesteuert haben. Die Arbeit hat zu zwei Veröffentlichungen geführt, bei der ODBASE 2006 in Montpellier (Springer LNCS

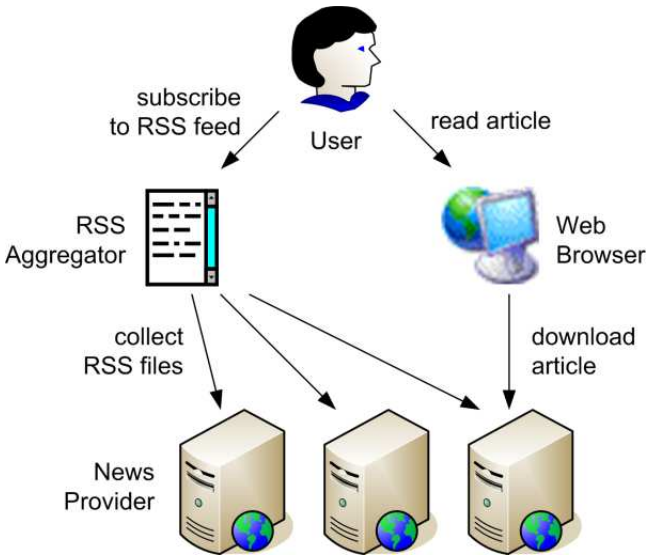


Abbildung 6.10: Typisches Szenario zur Verwendung von RSS-Feeds

4277 [21]) und bei der IADIS Internet/WWW 2006 in Murcia (IADIS Press [22]).

6.3.1 Ausgangslage

Im heutigen World Wide Web sind immer mehr Websites anzutreffen, die so genannte RSS-Feeds anbieten, mit denen man sich einen Überblick über neue Artikel auf den Seiten dieser Site verschaffen kann. RSS steht für RDF (oder auch Rich) Site Summary, dabei handelt es sich um eine Familie von XML-Sprachen, die Artikel auf Websites beschreiben. Eine typische RSS-Datei enthält Daten über die Herkunftssite, sowie Einträge für die einzelnen Artikel, die jeweils mit Überschrift, einer kurzen Zusammenfassung und dem Herkunftslink erfasst werden. Feed-Anbieter stellen solche Dateien an einem festen URL der Site zur Verfügung und aktualisieren sie bei Bedarf oder in festgelegten Intervallen. Abbildung 6.10 zeigt das typische Einsatzszenario von RSS Feeds. Die rechte Seite der Grafik zeigt den klassischen Weg an. Nutzer greifen über ihren Browser auf Web Sites zu und lesen die Artikel, die sie interessieren. Verfolgt man mehrere Sites, ergibt sich das Problem, dass es schwierig ist, immer auf dem Laufenden zu bleiben, welche Nachrichten auf welchen Sites neu hinzu gekommen sind. Hier setzen RSS Feeds an, die durch spezialisierte Programme, sogenannte Feed Aggregatoren, von Web Sites heruntergeladen werden und dem Nutzer anschließend in einer Oberfläche anzeigen können, welche neuen Artikel es auf den sie interessierenden Seiten gibt. Man spricht hier auch davon, dass Nutzer RSS Feeds abonnieren.

Feed-Aggregatoren funktionieren vergleichbar zu Usenet-Readern. Der Vorteil dieser Technik liegt in der Verfügbarkeit von Informationen über die Aktualisierung von Nachrichtenquellen in einem maschinell auswertbaren Format, so dass die Aufgabe der Ände-

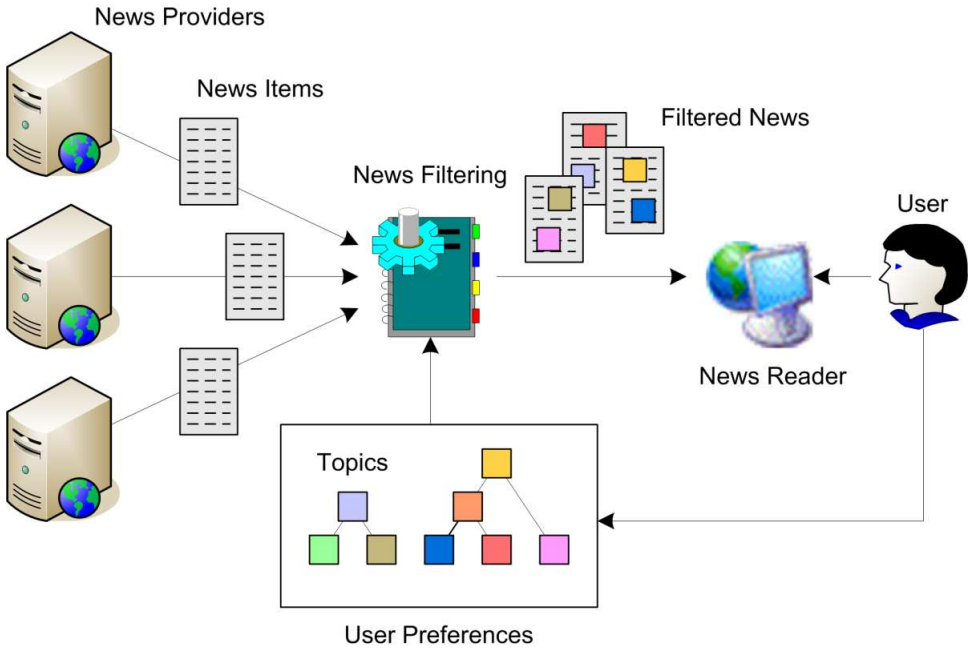


Abbildung 6.11: Schematischer Überblick über den Ansatz

rungsverfolgung von Nutzern auf den Computer delegiert werden kann.

Allerdings gibt es auch einige Nachteile bei diesen Systemen. RSS Feeds enthalten nur Informationen über die Artikel, die auch gerade auf einer Web Site zu sehen sind, d.h. sobald ein Artikel von den Seiten z.B. ins Archiv verschwunden ist, ist er auch nicht mehr im Feed enthalten. Wenn also der Aggregator nicht kontinuierlich läuft, kann es sein, dass Artikel auf der Seite erschienen und wieder verschwinden, ohne das der Nutzer es bemerkt.

Ferner steigt mit der Anzahl der abonnierten Feeds auch zwangsläufig die Anzahl der angezeigten Artikel. Um nicht den Überblick über die wichtigen Artikel zu verlieren, ist es an dieser Stelle notwendig, sich Gedanken über eine intelligentere Verwaltung der Artikel zu machen. Im Moment gibt es hierzu keine sinnvollen Mechanismen. Diese Arbeit zeigt einen Weg auf, eine semantische Filterung von RSS Feeds durchzuführen, die sich an der Interessenlage der Nutzer orientiert.

6.3.2 Semantische Filterung

Abbildung 6.11 zeigt eine symbolische Ansicht des vorgeschlagenen Ansatzes. Mehrere Nachrichtenanbieter stellen RSS Feeds für ihre Inhalte zur Verfügung, etwa aufgrund eines Abonnement durch den Benutzer. Diese Feeds werden über einen Filter geleitet, der seine Filterkriterien über eine vom Nutzer vorgegebene Repräsentation der Interessen bezieht. Da sich diese Interessen mit der Zeit ändern können, müssen die Kriterien auch verändert

werden können. Nur den Kriterien entsprechende Artikel passieren den Filter, so dass der Nutzer die Filterung nicht mehr selbst vornehmen muss. Die für das Passieren des Filters ausschlaggebenden Gründe sind den Artikeln als Metadaten beigefügt, so dass diese für weitere Verarbeitungsschritte (Archivierung, Indexierung, Weiterverwendung in anderen Applikationen) ausgewertet werden können.

Die folgenden Abschnitte stellen die Designziele für ein solches System dar und arbeiten die sich daraus ableitenden Anforderungen heraus.

Ziele

Der Kern des Systems liegt in der Definition und Verwertung der Nutzerinteressen. Zwar ließe sich so ein System mit der Eingabe von Schlagworten zur Themendefinition, gefolgt von einer klassischen Volltextsuche auf dem Material, realisieren, allerdings entspricht diese Vorgehensweise eher der Tätigkeit des Suchens, als der des Filterns. Bei der Suche ist anzunehmen, dass die Nutzer eine einigermaßen sichere Vorstellung haben, welche Information sie suchen. Diese lässt sich folgerichtig auch mit Schlagworten gut identifizieren. Beim Filtern von Informationen ist jedoch das Interesse üblicherweise viel breiter.

Hier sind eher Spezifikationen wie „Ich interessiere mich für deutsche Innenpolitik“ anzutreffen, die sich nicht wirklich gut mit einigen Schlagworten erschöpfend definieren lassen. Dabei besteht immer die Gefahr der Überspezifikation der Filter, bei der viele eigentlich interessante Artikel verloren gingen.

Das Ziel muss also sein, Themeninformationen zu erhalten oder zu generieren, die eine breitere Abdeckung des für den Nutzer interessanten Themas ermöglichen. Damit werden Sprachen des Semantic Web interessant, da diese die Formulierung von Konzepten und ihrer Beziehungen untereinander ermöglichen. Natürlich können die Nutzer keine komplette Ontologie zu einem Thema aufstellen, deshalb muss der Filter in der Lage sein, die gegebenen Ontologien sinnvoll automatisch zu erweitern, um eine bessere Abdeckung des Themengebiets zu erreichen.

Ein Filtersystem für RSS Feeds muss außerdem in der Lage sein, schnell auf Änderungen sowohl bei den Inhalten als auch bei den Kriterien zu reagieren. Neue Artikel müssen schnell an die Nutzer weitergeleitet werden, denn nichts ist uninteressanter als alte Nachrichten. Zudem sind die Nutzerinteressen gerade im Nachrichtenbereich sehr volatil. Während der Nutzung der Filter können sowohl graduelle Verschiebungen als auch abrupte Wechsel in den Interessen der Nutzer vorkommen. Letztere koinzidieren üblicherweise mit Skandalen, Katastrophen oder zeitgebundenen Ereignissen, zu denen eine Vielzahl von Artikeln erscheinen, die das Geschehen aus den verschiedensten Winkeln abdecken. Genauso schnell wie sie wichtig geworden sind, können sie allerdings auch wieder uninteressant werden. Ein Filter muss in der Lage sein, sich schnell gerade auch auf solche abrupten Wechsel einzustellen.

Anforderungen

Mit den oben genannten Zielen lassen sich vier Anforderungen an einen solchen semantischen Filter aufstellen:

1. Der Filter muss Ontologien entgegen nehmen können, in denen die Nutzerinteressen kodiert sind.

2. Im Falle eines Wechsels der Nutzerinteressen müssen die Filter-Ontologien geändert werden können.

3. Der Filter darf keine Annahmen über die Nutzerinteressen machen, somit zur Erweiterung der Ontologien nur inhärente Sprachfeatures und die vorliegenden Texte verwenden.

4. Der Filter muss sich schnell auf Änderungen der Interessenlage einstellen können.

6.3.3 Ansatz

Dieser Abschnitt beschreibt den gewählten Ansatz zur Realisierung eines Systems, das die Anforderungen aus dem vorangegangenen Abschnitt zu erfüllen versucht. Dazu kommen Ad-hoc - Ontologien zum Einsatz. Dieser Begriff wird nachfolgend zuerst definiert, ehe anschließend der Algorithmus vorgestellt wird.

Ad-Hoc - Ontologien

Die Anforderungen an das System stellen eine große Herausforderung dar, besonders die Nummern 2 und 3. Anforderung 2 verlangt die Fähigkeit zur Ontologie-Erzeugung für beliebige Themengebiete, während Anforderung 3 den Spielraum für diese Erstellung empfindlich einschränkt, da sie quasi komplett die Nutzung externer Ontologien oder Lexika ausschließt. Zusätzlich gilt es, das Ganze möglichst schnell zu erledigen (Anforderung 4), wodurch die Auswahl verfügbarer Techniken weiter schrumpft. Die Hauptaufgabe ist es jedoch, die erstgenannten Anforderungen zu erfüllen.

Ziel muss es also sein, automatisch eine Ontologie für ein bestimmtes Themengebiet zu erzeugen, die sich für die Filterung von Nachrichtenquellen eignet. Diese Art der Anwendung ist von der herkömmlichen, nämlich als Grundlage für eine Wissensbasis, deutlich zu unterscheiden. Dort möchte man einen Teil der Welt akkurat in einer Ontologie modellieren, um diese anschließend möglichst lange verwenden zu können. Zum Filtern reicht aber eine Ontologie aus, die eine verhältnismäßig einfache Klassifikationsaufgabe erfüllen kann: Muss dieser Artikel weitergereicht werden oder nicht? Zudem ist jederzeit damit zu

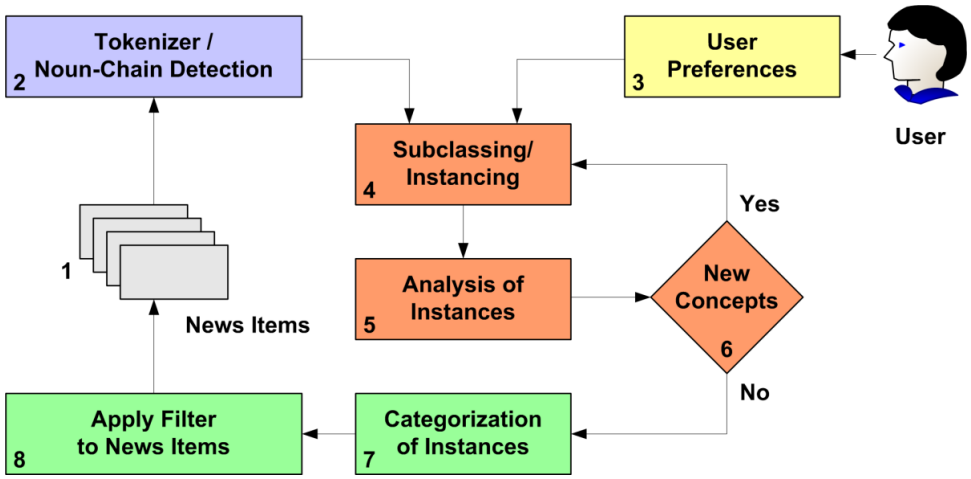


Abbildung 6.12: Ablaufplan des Filteralgorithmus

rechnen, dass die Ontologie gegen eine andere ausgetauscht wird, die das neue Nutzerinteresse abbildet. Solche Ontologien nennen wir *Ad-hoc - Ontologien*. Sie können durch die folgenden Eigenschaften charakterisiert werden:

1. Sie werden automatisch aus einem kleinen Kern von Konzepten und auf einer kleinen Dokumentenbasis erzeugt.

2. Sie enthalten üblicherweise recht wenige Relationstypen außer denen, die der verwendeten Modellierungssprache inhärent sind.

3. Sie haben eine niedrige Tiefe in der Richtung der Klasse-Subklassen-Beziehung.

Diese drei Eigenschaften sorgen für eine Eingruppierung der Ad-Hoc - Ontologien in die Klasse der einfachen Ontologien gemäß der Definition der unterschiedlichen Ontologietypen durch Deborah McGuiness in [39, 40] (siehe auch 3.2.1).

Algorithmus

Dieser Abschnitt stellt den Algorithmus vor, mit dem die Filter-Ontologie aufgebaut wird. Vor Beginn der Erklärung ist anzumerken, dass der Algorithmus nur für deutschsprachigen Text entwickelt worden ist, da Strukturen ausgenutzt werden, die sprachabhängig sind (Komposita und Großschreibungsregeln). Abbildung 6.12 zeigt die Schritte, die der Algorithmus abarbeitet.

Im ersten Schritt werden die Volltexte zu den Einträgen in den abonnierten RSS Feeds beschafft. Hierzu ist unter Umständen die Programmierung von HTML Scrapern notwendig, die den Quelltext der HTML-Seiten auswerten. Für die Erläuterung des Algorithmus

wird davon ausgegangen, dass solche Hilfsmittel zur Verfügung stehen und angewendet worden sind. Im Anschluss an diesen Schritt werden die Volltexte ausgewertet. Dazu werden sie zunächst automatisch in Sätze zerlegt, ehe Stoppworte durch Platzhalter ersetzt werden. Als Stoppwort werden alle klein geschriebenen Wörter angesehen, zusätzlich werden noch einige häufig vorkommende Satzanfänge gelöscht (Präpositionen, Verbundwörter). In den Texten sind damit nur noch Nomen, Zahlen, Platzhalter und Satzzeichen enthalten.

Nun wird nach Wortketten gesucht, d.h. Vorkommen von aufeinander folgenden Worten, bzw. Zahlen. Diese werden als Ergebnis an die nachfolgenden Schritte weitergegeben. Der Satz "CDU-Verteidigungsminister Jung sieht keinen Handlungsbedarf für ein stärkeres Engagement der Bundeswehr im gefährdeten Süden Afghanistans." wird durch die Stoppwortentfernung zu "CDU-Verteidigungsminister Jung _ _ Handlungsbedarf _ _ _ Engagement _ Bundeswehr _ _ Süden Afghanistans." umgewandelt. Davon werden die Ketten "CDU-Verteidigungsminister Jung" und "Süden Afghanistans" in der weiteren Bearbeitung berücksichtigt.

Schritt drei dient zur Sammlung der Nutzerinteressen in Form einer Kernontologie. Diese Ontologie macht von einigen wenigen Elementen Gebrauch: Die Definition von Klassen, Subklassen, Relationen, Subrelationen und Instanzen ist möglich. Kernontologien können schon mit ungefähr 10 Elementen für eine Filterung eingesetzt werden, ihre Erstellung bedeutet also keine große Arbeit für die Nutzer. Das Resultat dieses Schrittes ist die Kernontologie, die zusammen mit den Hauptwortketten aus Schritt zwei in die Verarbeitung des nächsten Prozessschritts eingegeben wird.

In diesem Schritt wird untersucht, ob die Namen von Elementen der Ontologie in den Hauptwortketten vorkommen. In dem Fall werden die entsprechenden Worte als Subklassen der Konzepte eingefügt, die die komplette Kette als Instanz erhalten. Ist für das obige Beispiel *CDU* als Konzept in der Ontologie enthalten, so wird *CDU-Verteidigungsminister* als Subklasse von *CDU* eingefügt, die *CDU-Verteidigungsminister Jung* als Instanz erhält. Diese Schritte werden für die alle Ketten abgearbeitet, es ist möglich, dass eine Kette Instanz verschiedener Konzepte wird.

In Schritt fünf wird eine tiefergehende Analyse der Instanzen durchgeführt, die im letzten Schritt erstellt worden sind. Dabei wird die Position des Wortes ausgewertet, das für die Instanziierung verantwortlich war. Dabei gilt es, drei Fälle zu unterscheiden: Steht der Identifikator vorne, so ist der Rest des Strings wahrscheinlich eine Spezialisierung davon, also eine Instanz. Dann wird der Identifikator aus dem Instanzlabel entfernt, ansonsten passiert nichts. Steht der Identifikator in der Mitte, ist die Wahrscheinlichkeit groß, dass der Identifikator eine Spezialisierung des Teils davor ist, während der Teil danach eine Instanz darstellt. In diesem Fall wird der Start des Strings genauer untersucht. Handelt es sich dabei um einen Genitiv, so wird der Nominativ davon in ein Kandidatenset überführt, ansonsten wird er entfernt und der Rest behandelt wie im ersten Fall. Nachdem alle Instanzen untersucht worden sind, wird das Kandidatenset näher ausgewertet.

Dazu wird zuerst für jeden Kandidaten ein Konzept angelegt, das mit dem Konzept

verbunden ist, mit dem es zusammen in einem String gefunden wurde. Das neue Konzept erhält eine “hat_{Name der Subklasse}“-Eigenschaft, das Konzept des Identifikators eine “hat_Kontext“-Eigenschaft. So enthält die Ontologie eine Verbindung zwischen den beiden Konzepten, ohne nähere Angaben über die Art der Beziehung wissen zu müssen. Sollte sich im Verlauf der Nutzung herausstellen, dass die entsprechenden Beziehungen wichtig für die Funktionsweise des Filters sind, so können sie durch den Nutzer explizit gemacht werden.

Im sechsten Schritt wird überprüft, ob die Ausführung der vorhergehenden Schritte Änderungen ergeben haben, die eine erneute Ausführung nötig machen. Dies wird immer dann der Fall sein, wenn im letzten Schritt neue Konzepte erstellt worden sind. Diese müssen gegen die Hauptwortketten getestet werden, um herauszufinden, ob es jetzt neue Instanzen gibt. Dadurch wird eine erschöpfende Suche auf den Ketten durchgeführt, die zu einer Ausweitung und Befüllung der Ontologie sorgt. Wenn sich hier keine Änderungen mehr ergeben, wird im siebten Schritt die Nachverarbeitung durchgeführt.

Darin wird versucht, die bisher nicht typisierten Instanzen anhand vom Nutzer definierter Relationen oder anhand des Klassen-Subklassen-Schemas zu typisieren, ehe im achten Schritt die eigentliche Annotation der Artikel unter Verwendung der Ontologie stattfindet. Die Schranke für die Weitergabe der Artikel kann parametrisiert werden, im Standardfall wird ein Artikel dann weitergereicht, wenn er mindestens eine Annotierung aus der Ontologie enthält.

Implementierung

Der Algorithmus ist prototypisch in Java implementiert worden. Zur Verwaltung der Ontologien in RDFS kam Jena zum Einsatz. Eine Java-GUI ermöglicht den Zugriff auf die gefilterten Artikel, die Definition von Filterkriterien wurde für die Überprüfung des Filters nicht interaktiv implementiert, sondern erfolgt über die Kommandozeile. Es wurden zwei HTML-Scraper implementiert, die in der Lage sind, Artikel von den Webseiten www.spiegel.de und www.stern.de auszuwerten und auf ihren reinen Artikeltext zu reduzieren. Die Implementierung speichert die extrahierten Hauptwortketten ab, um mehr Material zur Ontologieerweiterung zur Verfügung zu haben, wenn eine Änderung der Nutzerinteressen zu einer Neuberechnung der Ontologie führt. Ferner ist es möglich, einzustellen, ab welcher Anzahl enthaltener Annotationen ein Artikel weitergeleitet wird oder nicht.

6.3.4 Experimente

Auf dem Prototypen sind verschiedene Experimente durchgeführt worden. Die Datenbasis hierfür ist eine über zwei Wochen durchgeführte Sammlung von Artikeln der Quellen www.spiegel.de und www.stern.de zum Themengebiet Politik. Beide Quellen stellen auf dieses Gebiet eingeschränkte RSS Feeds zur Verfügung, unterscheiden dabei aber nicht

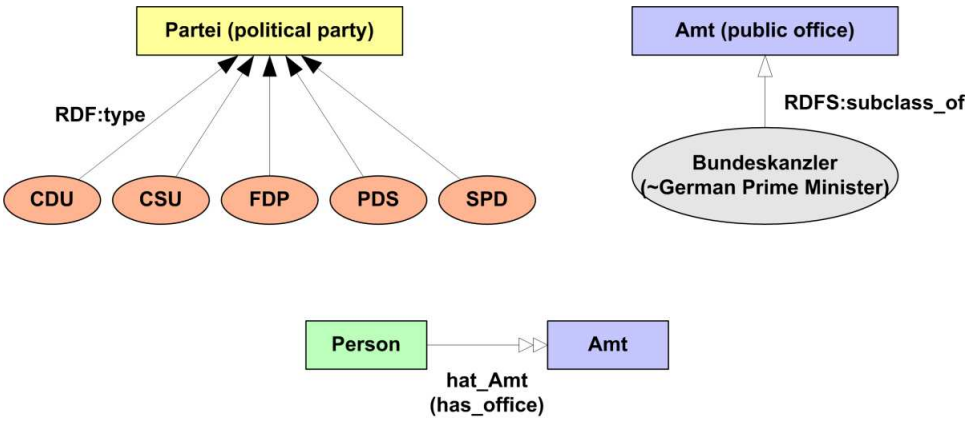


Abbildung 6.13: Kernontologie für das erste Experiment

zwischen Innen- und Außenpolitik. Die Kernontologien sind dergestalt in den Experimenten ausgewählt worden, dass für sie jeweils nur ein Teil der Artikel thematisch relevant ist.

Experiment 1: Erstellung einer Filterontologie

In diesem Experiment sollte die grundsätzliche Funktionalität des Filter-Algorithmus überprüft werden. Dafür wurde die in Abbildung 6.13 abgebildete Kernontologie in das System eingegeben. Das ausgesuchte Themengebiet war *Deutsche Politik*, die dazu ausgewählten Konzepte sind die Klasse *Partei*, dazu fünf Instanzen deutscher Parteien. Außerdem wird das Konzept des (politischen) Amts eingeführt, mit der Unterklasse *Bundeskanzler*.

Schließlich wird eine Klasse Person eingeführt, die über eine Relation mit Amt verbunden ist und modelliert, dass Personen Ämter einnehmen. Alles in allem werden zehn Elemente in der Ontologie modelliert, dazu zwei Relationstypen verwendet, die von RDFS zur Verfügung gestellt werden. Als Eingangsmaterial stand die erwähnte Sammlung von Nachrichtenartikeln zur Verfügung, insgesamt 383 Stück. Die Verarbeitung der Sammlung gemäß der Kernontologie benötigte 320 Sekunden bis zur Existenz der Hauptwortketten, die Erzeugung der Filterontologie benötigte anschließend 19 Sekunden. Nach der ersten Abarbeitung von Schritt vier enthielt die Ontologie 91 Konzepte. Bis zu diesem Zeitpunkt ist das Ergebnis des Algorithmus auch noch mit einer normalen Volltextsuche zu erzielen, da nur die Konzeptnamen aus der Kernontologie in den Hauptwortketten gesucht werden. Der Vorteil des Algorithmus erwächst aus den folgenden Iterationen: Die endgültige Ontologie enthielt 428 Elemente, davon rund 200 Instanzen des Typs *Person*.

Diese große Zahl hat ihre Ursache in der recht einfachen Form des Reasoning, das in Schritt sieben des Algorithmus stattfindet. Das Vorgehen lässt sich für das gegebene Beispiel so illustrieren: Instanzen des Konzepts *Bundeskanzler* müssen *Personen* sein, denn nach der Ontologie können nur *Personen* ein *Amt* haben - und *Bundeskanzler* ist eine

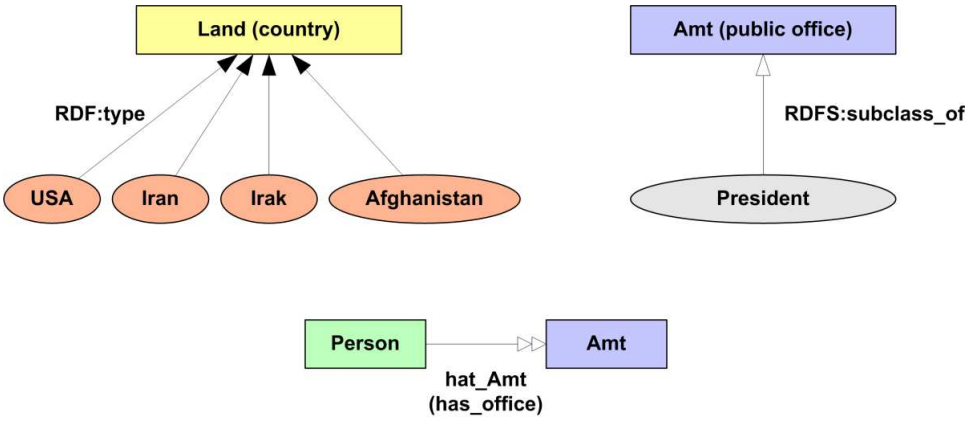


Abbildung 6.14: Kernontologie für das zweite Experiment

Subklasse von *Amt*. Ist eine der Instanzen von *Bundeskanzler* gleichzeitig Instanz einer anderen Klasse, dann müssen auch deren andere Instanzen *Personen* sein. Der Algorithmus iteriert, solange neue zu untersuchende Klassen gefunden werden.

Diese Art des Reasoning ist sehr einfach, trotzdem erzielt der Algorithmus bei den Personen eine Rate korrekter Klassifikationen von über 80%. Das zeigt die Tragfähigkeit der Annahme, auch wenn intelligenteres Reasoning bessere Ergebnisse zeitigen sollte. Zudem ist eine der Schwachstellen des Algorithmus in der Erzeugung der Ketten zu suchen, bessere Algorithmen hier würden sich ebenfalls positiv auf das Ergebnis auswirken. Ferner zeigt die Laufzeit im Experiment, dass es besser ist, die Hauptwortketten zwischenzuspeichern, sobald es sie gibt, da deren Erzeugung deutlich teurer (Faktor 16!) ist, als die eigentliche Ontologie-Erstellung.

Experiment 2: Wechsel des Nutzerinteresses

Dieses Experiment dient zur Klärung der Frage wie sich die Änderung des Nutzerinteresses auf die Performanz des Filters auswirkt. Dazu wurde die erste Hälfte der Artikel mit der Kernontologie aus dem ersten Experiment verarbeitet, wonach die Ontologie aus Abbildung 6.14 in das System eingegeben wurde. Diese Ontologie beschäftigt sich mit Außenpolitik[14], speziell mit dem Themenkomplex des Irak-Kriegs. zur Gewinnung einer Baseline wurde die zweite Hälfte der Dokumente einmal ohne weiteren Durchlauf des Algorithmus annotiert, was zu einer Fehlkategorisierung von rund 27% führte. Anschließend wurde die zweite Hälfte der Artikel noch einmal annotiert, dieses Mal unter Verwendung des Algorithmus.

Die endgültige Ontologie enthielt 191 Konzepte, davon 44 als Personen klassifiziert, 4 davon fälschlicherweise. Die Fehlerquote bei der Artikelklassifizierung verringerte sich auf 13%. Das zeigt sowohl, dass die Anpassung an Nutzerinteressenwechsel funktioniert, als

[14]Jedenfalls aus deutscher Sicht

auch, dass die Qualität der Klassifikationen des Systems mit der Zeit anwächst. Auch hier zeigte sich allerdings, dass die Erstellung der Hauptwortketten deutlich länger dauert, als die anschließende Einarbeitung der Ketten in die Ontologie.

6.4 Zusammenfassung

Dieses Kapitel hat das e-Science-Programm des BMBF vorgestellt, in dem die GRID-Aktivitäten sowie Arbeiten zur Wissensvernetzung zusammengefasst sind. Ein Teil dieser Arbeiten ist das vom Fraunhofer IAIS koordinierte WIKINGER-Projekt, das die Referenzimplementierung des in Kapitel 5 entwickelten Ansatzes darstellt.

Die einzelnen Bestandteile des Systems sind detailliert beschrieben, etwaige Abweichungen von den Ansätzen sind notiert und motiviert worden. Die Beschreibung des Systems enthält die Anteile der anderen Projektpartner, diese sind jedoch entsprechend gekennzeichnet und keine Ergebnisse der in dieser Arbeit entwickelten Ansätze. Die vom IAIS beigesteuerten Bestandteile des Projekts sind unter der Leitung des Autors entstanden und stellen die exemplarische Umsetzung der Ansätze aus Kapitel 5 dar.

Darüber hinaus ist ein System vorgestellt worden, anhand dessen die Möglichkeiten und Grenzen automatisch erzeugter Ontologien untersucht worden sind. Der dort verwendete Algorithmus verwendet einen Ontologie-Kern, um daraus automatisch eine Sicht auf die Inhalte eines Textkorpus zu erstellen. Die Ausdrucksfähigkeit einer solchen Ontologie ist zwar sehr beschränkt, für gewisse Anwendungen ist sie allerdings hinreichend. Eine Szenario für eine solche Anwendung ist mit dem Problem der semantischen Filterung von Newsletter-Strömen gegeben worden.

So sind in diesem Kapitel beide Enden des Spektrums der Möglichkeiten der automatischen Unterstützung bei der Ontologieerstellung betrachtet worden.

Kapitel 7

Evaluierung

Dieses Kapitel beschäftigt sich mit der Evaluierung der Ansätze, die in Kapitel 5 vorgestellt worden sind. Dabei soll auf verschiedene Aspekte eingegangen werden: Neben der Qualität der Ergebnisse der Verfahren wird auch die Geschwindigkeit, in der sie erreicht werden, überprüft. Die Untersuchungen werden auf Basis der im WIKINGER-Projekt erstellten Referenzimplementierung durchgeführt.

Das Kapitel ist entlang der zu untersuchenden Gebiete organisiert, d.h. die zu einem Modul gehörenden Untersuchungen sind jeweils in einem Unterkapitel zusammengefasst. Untersucht werden die Relationserkennung, die Relationsklassifikation mittels der von den Experten ausgewählten Relationsmenge, sowie die Netzerstellung aus den automatisch gefundenen Entitäten und Relationen.

7.1 Relationserkennung

Dem Modul zur Relationserkennung kommt eine große Bedeutung innerhalb des Systems zu, da es die Kanten liefert, die zur Vervollständigung des semantischen Netzes benötigt werden. Die Steuerung des Vorgangs erfolgt über eine interaktive Benutzerschnittstelle, neben der Qualität der Ergebnisse sind also auch die Geschwindigkeit des Moduls sowie die Ergonomie der GUI zu evaluieren.

7.1.1 Vorbemerkungen

Die Experimente werden auf der Basis des im WIKINGER-Projekts erstellten Korpus durchgeführt. Für die Evaluierung der Relationserkennung werden die Dokumente verwendet, die von den Experten der KfZG zum Training der Eigennamenextraktion manuell annotiert worden sind. Bei diesen Dokumenten ist von einer sehr hohen Erkennungsgenau-

igkeit der Entitäten auszugehen, so dass Fremdeinflüsse bei Fehlern der Module weniger ins Gewicht fallen - wenn die Ergebnisse schlecht sein sollten, dann liegt es an den Algorithmen, nicht an falsch erkannten Entitäten.

Im WIKINGER-Projekt werden die folgenden Entitätsklassen verwendet:

- Biographisches Ereignis
- Datum/Zeit
- Einrichtung
- Geo-Politische Entität (GPE), ein klar abgrenzbarer Ort, der eine eigene Regierung hat, also Dörfer, Städte, Bundesländer oder -staaten und Nationen
- Organisation
- Ort, nicht klar abgrenzbare Georeferenzen, z.B. „Schwarzwald“
- Person
- Rolle
- Titel
- Bedeutendes Ereignis

Die Klasse *Person* hat eine besondere Stellung innerhalb des Korpus: Die im Projekt untersuchten Relationsmuster stehen alle implizit im Bezug zu einer Person, auch wenn sie nicht direkt in der Auflistung der zur Relation gehörenden Entitäten vorkommt. Das liegt daran, dass die Relationen aus biographischen Texten stammen. Dadurch ist die Person eine Invariante der Relationserkennung und wird im Folgenden nicht weiter ausgewiesen.

7.1.2 Qualität der Relationserkennung

Zur Überprüfung der Qualität der Relationserkennung wurden die acht Dokumente herangezogen, die für das Training der Eigennamenerkenner von den Domänenexperten manuell annotiert worden sind. Die Erstellung der Assoziationsregeln ergab die in Tabelle 7.1 gezeigt Liste von 13 Regeln mit zusammen 2095 Vorkommen.

Darin sind binäre Regeln in der Überzahl, allerdings gibt es auch zwei mehrstellige Regeln, die zudem zusammen gut ein Viertel der absoluten Vorkommen ausmachen. Da ein Muster nur zu einer Regel gehören kann, sind keine Überlappungen zwischen den Regeln $GPE \rightarrow Datum/Zeit, Rolle$ und $GPE \rightarrow Rolle$ möglich. Alle Muster, die zur spezielleren

Assoziationsregel	# Vorkommen
$BiographischesEreignis \rightarrow Datum/Zeit$	428
$GPE \rightarrow Datum/Zeit, Rolle$	359
$Rolle \rightarrow Datum/Zeit$	337
$GPE \rightarrow Datum/Zeit$	202
$Organisation \rightarrow Datum/Zeit, Rolle$	201
$Organisation \rightarrow GPE$	155
$GPE \rightarrow Rolle$	127
$Organisation \rightarrow Rolle$	93
$Organisation \rightarrow Datum/Zeit$	50
$Titel \rightarrow Datum/Zeit$	47
$Einrichtung \rightarrow Datum/Zeit$	38
$Einrichtung \rightarrow GPE$	32
$Einrichtung \rightarrow Rolle$	26

Tabelle 7.1: Ergebnisse der Assoziationsregelerstellung

Regel passen, wurden aus der generelleren entfernt. Für die Überprüfung der Relationserkennung wurden diese beiden Regeln ausgewählt und zunächst die darin enthaltenen Relationen manuell bestimmt. Für die Regel $GPE \rightarrow Datum/Zeit, Rolle$ waren das 14 Relationen, die Regel $GPE \rightarrow Rolle$ enthielt 7 Relationen.

Evaluierung

Die Evaluierung der Qualität erfolgt anhand der zwei Kriterien Cluster-Zusammensetzung und Relationserkennung. Dazu wird pro automatisch erstelltem Cluster eine Hauptrelation bestimmt, nämlich diejenige, die die meisten stützenden Vorkommen aufweist. Dies sind die für diesen Cluster richtigen Vorkommen V_r, alle anderen sind für diesen Cluster falsch, V_f. Die manuell für die betreffende Relation bestimmten Vorkommen sind mit V_m bezeichnet. Damit lassen sich Recall, Precision und F_0-Measure wie folgt angeben:

Recall (R): $R = \frac{V_r}{V_m}$

Precision (P): $P = \frac{V_r}{V_r+V_f}$

F_0-Measure (F): $F = \frac{2RP}{R+P}$

Cluster-Zusammensetzung Die Resultate der beiden ausgewählten Regeln sind in Tabelle 7.2 abgebildet. Die Werte für die Regel $GPE \rightarrow Datum/Zeit, Rolle$ sind insgesamt

Association rule	R	P	F
$GPE \rightarrow Datum/Zeit, Rolle$	0,85	0,69	0,75
$GPE \rightarrow Rolle$	0,79	0,65	0,71

Tabelle 7.2: Recall, Precision und F-Measure der Relations-Cluster

etwas besser. Der Recall ist in beiden Fällen höher als die Precision, liegt allerdings deutlich unter 100%. Dies ist auf Vorkommen zurückzuführen, deren Struktur sich deutlich von denen anderer unterschied, so dass sie eigene Cluster bildeten und so aufgrund des Nichteinhaltens der Mindestgröße aussortiert wurden.

Die einzelnen Relationscluster zeigen teils deutlich höhere Werte. Je allgemeiner eine Relation ist, desto höher Precision und Recall, wohingegen speziellere Relationen sehr hohe Precision erreichen, zum Teil bis 100%, allerdings bei drastisch niedrigerem Recall. Das beeinflusst die Gesamtwerte deutlich.

Relationserkennung Der zweite Teil der Evaluierung beschäftigt sich mit der Untersuchung, welche der manuell bestimmten Relationen sich in den Relations-Clustern wiederfinden lassen. Die Verarbeitung der Regel $GPE \rightarrow Datum/Zeit, Rolle$ ergab neun Cluster, deren Hauptrelationen alle in der manuell erstellten Liste enthalten waren. Die übrigen fünf erwarteten Relationen verteilten sich zum größten Teil über die Cluster, die aufgrund der nicht erreichten Mindestgröße aussortiert worden waren. Eine dieser Relationen war komplett in dem größten Cluster aufgegangen. Experimente mit noch strengeren Schwellwerten lösten diese Relation erfolgreich aus diesem Cluster heraus, trennten allerdings auch weitere Cluster auf, so dass sich das Gesamtergebnis in diesem Fall eher verschlechtert hätte.

Die Verarbeitung der Regel $GPE \rightarrow Rolle$ ergab ein ähnliches Bild: Fünf Cluster wurden automatisch erstellt, die darin abgebildeten Relationen waren alle Teil der vorher erstellten Liste. Die übrigen zwei Relationen waren über solche Cluster verteilt, die aufgrund ihrer Größe nicht weiter berücksichtigt wurden.

Die für diesen Algorithmus problematischen Relationen lassen sich in drei Klassen unterteilen:

1. Relationen, deren Vorkommen kaum Merkmale teilen
2. Relationen, die andern Relationen sehr ähnlich sind, von diesen aber mit anderen Parameterwerten getrennt werden können
3. Relationen, die anderen so stark ähneln, dass deren Muster sich nicht automatisch trennen lassen

Die Vorkommen der Relationen der ersten Klasse unterscheiden sich oft aufgrund der Verwendung von Synonymen deutlich voneinander, so dass sie entweder in viele Cluster zer-

fallen oder teilweise zu anderen Clustern zugeschlagen werden. Die der zweiten Klasse sind durch eine Verschärfung der Kriterien zu extrahieren, allerdings kann dadurch die Gesamtperformanz drastisch in Mitleidenschaft gezogen werden, wenn dadurch andere Cluster ebenfalls aufgebrochen werden. Die Relationen der dritten Klasse schließlich unterscheiden sich oftmals nur subtil in ihrer Bedeutung von anderen existierenden Relationen, so dass eine Trennung mit rein statistischen Methoden schwerlich zu erreichen sein wird, so man nicht extrem große Korpora verwenden kann.

Bewertung

Die Ergebnisse der Evaluierung der Clusterzusammensetzung zeigen, dass das Verfahren gut in der Lage ist, die Vorkommen mit rein statistischen Mitteln auf verschiedene Cluster zu verteilen. Hierbei sind die Precision-Werte zwar geschmälert durch fehlerhafte Zuweisungen, das liegt allerdings nicht zuletzt an der Art der gesuchten Relationen, deren manuelle Zusammenstellung Kombinationen ermöglicht, die sich so automatisch nicht nachbilden lassen, ohne die Unabhängigkeit von der verwendeten Sprache zu verlieren.

Die Überprüfung der Cluster zeigte, dass diese in der Tat Relationen abbilden, die auch manuell aus den Vorkommen zusammengestellt werden würden, der Relations-Recall liegt in beiden Fällen bei über 60% (64% bzw. 71%), die Precision hingegen bei 100%, da alle gebildeten Cluster gesuchte Relationen abbildeten. Damit ergeben sich Werte für F_0-Measure von 82% bzw. 86% – sehr gute Ergebnisse für die Relationserkennung. Dazu ist anzumerken, dass diese Werte natürlich von der Granularität der manuell festgelegten Relationen abhängen; die Domänenexperten legten in einem Vergleichstest weniger Relationen fest, die Bedeutungsnuancen spielten für deren Bedürfnisse nicht so eine große Rolle, obgleich sie das Angebot der möglichen Relationen durchaus zu schätzen wussten. Dies hat klare Implikationen für die Funktionalitäten der Nutzerschnittstelle (siehe dort).

7.1.3 Geschwindigkeit

Da die Relationserkennung eingebettet in eine graphische Benutzeroberfläche stattfindet, ist deren Geschwindigkeit von großer Bedeutung. Benötigt eine Applikation zu lange für die Berechnung der Relationskandidaten, so wird sie von den Nutzern nicht akzeptiert und dann auch von ihnen nicht verwendet werden. In der WiReD GUI[1] sind zwei rechenintensivere Arbeitsschritte enthalten: Die Assoziationsregelbestimmung und die Relationserkennung. Messungen der Laufzeiten dieser beiden Schritte sind die Aufgabe der folgenden Tests.

WiReD ist dafür ausgelegt, von Fachexperten auf deren Arbeitsplatzrechnern lokal gestartet zu werden, was es unmöglich macht, globale Performanzaus- bzw. -zusagen zu ma-

[1] Vgl. Kapitel 6.2.5

Laufzeiten (in s)	Rechner	
	AMD Duron	Core2Duo
Assoziationsregeln	67,2	16,8
Relationserkennung auf 2-stelliger Regel	5,1	1,0
Relationserkennung auf 3-stelliger Regel	6,2	1,4
Relationserkennung auf 4-stelliger Regel	6,3	1,3

Tabelle 7.3: Laufzeit der Relationserkennung auf verschiedenen Rechnern

chen. Das Ziel dieser Tests ist daher, zu sinnvollen Hardware-Empfehlungen zu gelangen, die eine hinreichende Performanz der GUI ermöglichen.

Evaluierung

Die Tests werden auf zwei verschiedenen PC-Systemen durchgeführt, um eine gewisse Hardware-Bandbreite abzudecken. Am unteren Ende des Spektrums steht ein AMD Duron mit 850 MHz aus dem Jahr 2001, am oberen Ende ein Intel Core2Duo 6400, ein PC mit zwei Prozessoren mit je 2133 MHz aus dem Jahr 2007. Das der ältere Rechner langsamer sein wird, steht außer Frage, interessant ist die Frage *wieviel* langsamer er ist und ob die erreichte Zeit für einen interaktiven Prozess noch zumutbar ist.

Die Messungen werden anhand des WIKINGER-Korpus durchgeführt, unter Verwendung der von den Experten manuell annotierten zwölf Dokumente. Zunächst werden die Assoziationsregeln bestimmt. Die Zeitmessungen wurden fünfmal durchgeführt und anschließend gemittelt, um Schwankungen aufgrund von Hintergrundprozessen des Betriebssystems zu filtern. Für die Relationserkennung werden Regeln mit verschiedenen Stelligkeiten ausprobiert und ebenfalls je fünfmal deren Relationskandidaten bestimmt, die Laufzeit gemessen und gemittelt.

Tabelle 7.3 zeigt die Resultate der Testreihen.

Bewertung

Die Messergebnisse für den Intel sind sehr zufriedenstellend, die Wartezeiten während der Berechnungen liegen in einem Rahmen, den Nutzer von ihren Browsern gewohnt sind. Zudem müssen die Assoziationsregeln nur einmal berechnet werden, sie fallen also bei längerer Arbeit mit der GUI nicht ins Gewicht.

Die Relationserkennung sorgt nicht für spürbare Verzögerungen im Arbeitsablauf, mit Laufzeiten unterhalb 1,5 Sekunden über die getesteten Stelligkeiten. Die binären Regeln sind zwar die mit den meisten Vorkommen, werden allerdings trotzdem am schnellsten bearbeitet. Das liegt in der begrenzteren Anzahl möglicher Relationsmuster begründet.

Die kleineren mehrstelligen Regeln weisen dagegen eine höhere Varianz auf, die sich in mehr Kandidatenclustern und damit einem längeren Clusteringprozess niederschlägt.

Bei der Relationserkennung gilt, dass die erste Berechnung etwas länger dauert (4-5 Sekunden), da dabei Klassen in den Speicher nachgeladen werden, die bei den folgenden Aufrufen direkt zur Verfügung stehen. Diese laufen dann deutlich schneller ab, benötigen zwischen 0,1 und 0,5 Sekunden.

Die für den Duron gemessenen Werte sind deutlich schlechter. Hier muss eine gute Minute auf die Assoziationsregeln gewartet werden. Die Relationserkennung hingegen läuft deutlich schneller. Auch bei diesem Rechner sind längere Laufzeiten für die komplexeren Regeln gegenüber den größeren, aber einfacheren zu beobachten. Aber die Laufzeiten liegen mit unter sieben Sekunden noch im Rahmen. Die erste Relationsberechnung dauert quer über alle Regelarten ungefähr 20 Sekunden, die nachfolgenden Berechnungen werden dann in zwischen 0,7 und 3 Sekunden abgearbeitet.

Zusammenfassend lässt sich festhalten, dass sich die Vorarbeiten für die Relationserkennung auch auf älterer Hardware durchführen lassen, wenn auch manchmal kürzere Wartezeiten in Kauf genommen werden müssen. Diese Rechnergeneration sollte allerdings zunehmend seltener in Büroumgebungen anzutreffen sein. Auf einigermaßen aktueller Hardware aus dem unteren bis mittleren Leistungssegment ist WiReD ohne Wartezeiten lauffähig. Es hat sich gezeigt, dass bei der Laufzeit die Komplexität der Regeln eine deutlich größere Rolle spielt als die Menge der zu betrachtenden Beobachtungen. Das deckt sich mit der Erwartung, dass n-stellige Regeln mehr Nuancen im Ausdruck der enthaltenen Relationen aufweisen, als die einfacheren binären Regeln.

7.1.4 Nutzerschnittstelle

Die Evaluierung von Nutzerschnittstellen ist naturgemäß eine sehr schwierige Aufgabe, da hierbei sehr stark individuelle Vorlieben und Vorkenntnisse der Testpersonen einfließen. Im WIKINGER-Projekt stehen Usability-Tests der Oberflächen nicht im Fokus, trotzdem stand die Entwicklung der Schnittstellen zur Relationserstellung auch unter der Maßgabe, eine Steuerung der Prozesse gerade auch für Fachfremde zu ermöglichen.

Bei der Realisierung von WiReD wurde daher auf die optisch klare Trennung der verschiedenen Aufgaben geachtet. Die Zahl der einzustellenden Parameter ist bewusst klein gehalten: Der ganze Prozess wird über vier Parameter gesteuert, die zudem mit Standardwerten vorbelegt sind, so dass sinnvolle Ergebnisse auch ohne Veränderung der Parameter erzielt werden können.

Die Clustering-Phase des Workflows hat einen ausgeprägt explorativen Charakter, da sich hier die Auswirkungen von Parameteränderungen gut beobachten lassen. Um dies zu unterstützen, werden die Clustering-Ergebnisse jeweils in einem eigenen Tab gezeigt, so dass diese einfach miteinander verglichen werden können.

Schulungsaufwand

Die Domänenexperten aus dem WIKINGER-Projekt sind jeweils ca. eine Stunde in die Verwendung von WiReD eingeführt worden, wonach sie in der Lage waren, eigenständig die für ihren Anwendungsfall wichtigen Relationen zu bestimmen und zu benennen. Dieser Aufwand ist doppelt so hoch wie der für die Einführung in WALU benötigte, allerdings ist auch der Erklärungsaufwand für die benötigten Arbeitsschritte höher. WALU kann auf der Metapher des Markierens relevanter Textpassagen aufbauen, so eine Metapher fehlt bei der Bestimmung von Relationen.

Technik

Die Installation von WiReD gestaltet sich recht einfach, im Prinzip ist mit dem Entpacken eines Archivs in ein beliebiges Verzeichnis die Hauptarbeit bereits erledigt. WiReD ist in Java geschrieben, erfordert also ein Java Runtime Environment auf dem Zielrechner, das vielerorts durch Web Browser bereits installiert sein dürfte. An Hauptspeicher und Prozessorgeschwindigkeit stellt WiReD keine großen Anforderungen, siehe 7.1.3.

Zur Übertragung der Relationsdaten an den WIKINGER-Server ist ein Netzzugang erforderlich, alternativ können die Ergebnisse auch lokal gespeichert und anderweitig auf den Server übertragen werden. Hier zeigte sich auch eine Schwachstelle der Oberfläche: Die zu übertragenden Relationsdaten hatten eine beträchtliche Größe, so dass die Übertragung an den Server sehr lange dauerte. Dies führte mehrmals zu Übertragungsabbrüchen durch Time Outs auf Serverseite oder unbeabsichtigte Abbrüche durch die Nutzer. Daraufhin wurde das Relationsformat einer Prüfung unterzogen und die Menge der enthaltenen Daten drastisch reduziert (Faktor 10).

Bewertung

Die Erfahrungen mit WiReD zeigen deutlich, dass es möglich ist, Oberflächen auch für komplexere Aufgaben des Text Mining so zu entwerfen, dass Domänenexperten damit eigenständig umgehen können. Entscheidende Beiträge dazu leisten klare optische Trennungen der einzelnen Teilaufgaben, eine sinnvolle Beschränkung der einzustellenden Parameter und deren Vorbelegung mit experimentell bestimmten sinnvollen Werten.

Auf Anregung der Experten sind einige Funktionalitäten zusätzlich in die Oberfläche aufgenommen worden, zum Beispiel die Option, einzelne Sätze aus den Clustern zu löschen. Dies war anfangs nicht vorgesehen, um die Experten nicht zu verwirren, es hat sich allerdings gezeigt, dass diese Zusatzfunktion von ihnen gut aufgenommen und nicht exzessiv zum händischen Nachbearbeiten genutzt wird.

7.2 Relationsklassifikation

Das Modul der Relationsklassifikation ist verantwortlich für das automatische Auffinden und Klassifizieren von Relationsmustern im Restkorpus. Dies ist ein reiner Backend-Prozess, d.h. es findet keine direkte Interaktion mit den Nutzern statt. Daher werden für die Evaluierung nur die Faktoren Qualität und Geschwindigkeit herangezogen.

7.2.1 Vorbemerkungen

Die Relationsklassifikation setzt die Existenz von Relationen voraus, die in der Zielontologie enthalten sein sollen. Um die Performanz des Moduls unter realistischen Bedingungen testen zu können, wird der Relationssatz von ca. 30 Relationen verwendet, den die Domänenexperten im WIKINGER-Projekt bestimmt haben. Die Klassifikation bedient sich des WIKINGER-Korpus, d.h. rund 160 Bücher, jeweils zwischen 300 und 1000 Seiten lang. Diese sind bereits automatisch mit den in 7.1.1 aufgeführten Entitätsklassen annotiert worden.

Für die Tests wird eine Servermaschine mit zwei Xeon-Prozessoren und 4 Gigabyte Hauptspeicher eingesetzt. Auf der Maschine laufen außerdem einige andere Web-Dienste, sie eignet sich daher gut für Tests unter Praxisbedingungen.

7.2.2 Qualität der Klassifikation

In dieser Evaluierung wird untersucht, welcher Prozentsatz der automatisch extrahierbaren Relationsvorkommen tatsächlich von den automatischen Prozessen gefunden und richtig zugeordnet werden kann. Die Qualität der automatischen Klassifikation ist von entscheidender Bedeutung für das Funktionieren des gesamten Ansatzes, da sich nur durch Automation der Zuordnungsarbeit eine Massenverarbeitung digital vorliegender Datenbestände realisieren lässt.

Evaluierung

Zur Evaluierung wurden zufällig vier Dokumente ausgewählt, die durch die automatische Eigennamenerkennung vorverarbeitet wurden. Zusätzlich wurden die Assoziationsregeln mit ihren Vorkommen bestimmt. Diese Vorkommen wurden anschließend manuell anhand der vorliegenden Relationen klassifiziert.

Die Evaluierung verwendet diese manuellen Zuordnungen als Richtschnur für die Messung der Qualität der automatischen Klassifikation. Tabelle 7.4 zeigt die dabei erzielten Ergebnisse für Precision (P), Recall (R) und F_0-Measure (F_0), jeweils für die am besten und

	Beste			Schlechteste			Global		
	R	P	F_0	R	P	F_0	R (R')	P	F_0 (F_0')
Dokument 1	1	1,	1	0	0	0	0,52 (0,73)	0,96	0,74 (0,84)
Dokument 2	1	1	1	0	0	0	0,59 (0,67)	1,00	0,80 (0,83)
Dokument 3	1	1	1	0	0	0	0,61 (0,65)	1,00	0,81 (0,82)
Dokument 4	1	1	1	0	0	0	0,72 (0,78)	1,00	0,86 (0,89)

Tabelle 7.4: Precision und Recall der automatischen Relationsklassifikation

die am schlechtesten klassifizierte Relation und den Mittelwert über das Dokument. R' und F_0' bezeichnen die Werte, die erreicht werden, wenn die bei der manuellen Klassifikation als irrelevant klassifizierten Muster aus der Gesamtmenge der Vorkommen im Dokument getilgt werden. Die Tabelle zeigt in den beiden ersten Hauptspalten ein ausgeglichenes Bild. In jedem Dokument gibt es Relationen, deren Vorkommen komplett richtig automatisch klassifiziert worden sind und solche, deren Vorkommen nicht gefunden worden sind. Das ist für sich genommen nicht sonderlich aussagekräftig, zusammen mit der letzten Hauptspalte ergibt sich jedoch ein differenziertes Bild: Die Klassifikationspräzision liegt üblicherweise bei 100%, die klassifizierten Muster sind also in der Regel auch korrekt zugewiesen. Der Recall, also der Anteil der zugewiesenen Muster an den manuell bestimmten, liegt zwischen 52% und 72%, was sich in F_0-Maßen zwischen 74% und 86% Prozent niederschlägt. Betrachtet man R' und F_0', so verbessern sich die Werte noch einmal, der Recall auf 65% bis 78% und F_0 auf 82% bis 89%.

Bewertung

Die in der Tabelle aufgeführten Werte zeigen, dass die automatische Klassifikation ihre Aufgabe im Schnitt sehr gut erfüllt. Jeweils mehr als 65% der gültigen Relationsmuster wurden gefunden und bis auf eine Ausnahme auch korrekt zugeordnet. Die manuell als nicht relevant erachteten Muster wurden auch von der automatischen Klassifikation verworfen.

Gleichwohl treten Ausfälle in der Klassifikation auf, es gibt immer wieder Relationen, die sich nicht automatisch klassifizieren lassen. Dies sind allerdings durchweg solche, die nur über sehr wenige Beispiele im Korpus bestimmt worden sind (in der Testmenge hatte keine dieser Relationen mehr als zwei Beispiele). Dies lässt Rückschlüsse auf die Aussagekraft der von den Experten ausgewählten Relationsmuster zu. Hier kann angesetzt werden, um die Qualität der Klassifikatoren zu verbessern, wodurch sich der Recall zusätzlich erhöhen lassen dürfte. Die Recall-Werte ließen sich sicherlich noch steigern, berücksichtigte man sprachabhängige Faktoren bei der Klassifikation, etwa, indem man eine Lemmatisierung durchführte sowie Synonyme auflöste. Allerdings büßte man so die Sprachunabhängigkeit des rein statistischen Ansatzes ein, was die generelle Einsatzfähigkeit des Systems beeinträchtigte. Angesichts der erzielten Werte kann allerdings davon Abstand genommen werden, die Ergebnisse auf diese Weise zu verbessern.

Laufzeiten (in s)	Paketgröße			
	5	10	20	25
Paket	0,5	0,9	2,1	2,9
Dokument	0,1	0,1	0,1	0,1
Vorverarbeitung Paket	147,2	168,1	360,9	415,3
Vorverarbeitung Dokument	29,4	16,8	18,0	16,6

Tabelle 7.5: Klassifikationsgeschwindigkeiten bei verschiedenen Paketgrößen

Zusammenfassend lässt sich festhalten, dass die Qualität der automatischen Klassifikation den in sie gesetzten Ansprüchen genügt. Nimmt man den Schnitt über die betrachteten Dokumente, so liegt der Recall bei 71% und die Precision bei 99%, woraus sich ein F_0-Maß von 83% ergibt.

7.2.3 Geschwindigkeit

In Kapitel 5.3.3 ist bereits auf die Speicherintensität der Relationsklassifikation eingegangen worden. Daher wird in diesem Teil der Evaluierung geprüft, welche Paketgrößen für die effiziente Abarbeitung der Dokumentensammlung unter Praxisbedingungen zu wählen ist. Dabei sind verfügbare Speichermengen ebenso zu beachten wie die Overhead-Kosten, die durch die Service-Kommunikation entstehen.

Evaluierung

Um die Auswirkungen unterschiedlicher Speicherbedingungen auf die Performanz untersuchen zu können, wird die Zeit gemessen, die für die Verarbeitung des Korpus bei unterschiedlichen Paketgrößen benötigt wird. Dazu werden Klassifikationen mit Paketgrößen von 5, 10, 20 und 25 Dokumenten auf der in 7.2.1 angegebenen Maschine durchgeführt.

Tabelle 7.5 zeigt die dabei erzielten Ergebnisse für die Pakete und umgerechnet auf ein einzelnes Dokument, sowie die für die Vorverarbeitung benötigten Zeiten. Die Zeiten sind über fünf verschiedene Messungen gemittelt, um Sondereffekte durch andere auf dem Server laufende Prozesse zu minimieren.

Bewertung

Die Ergebnisse der Geschwindigkeitsmessung zeigen, dass die Klassifikation zwar ein speicherintensiver Vorgang ist, allerdings keine langen Laufzeiten aufweist. Die Laufzeiten für die Klassifikation unterscheiden sich lediglich um eine Vierzigstelsekunde pro Dokument, unterschiedliche Paketgrößen fallen hier also nicht so sehr ins Gewicht. Geht man nur nach dieser Größe, dann sind Pakete mit 10 Büchern am besten für die Verarbeitung geeignet.

Wenn man die Vorverarbeitung ebenfalls berücksichtigt, also die Anforderung der Dokumente vom Repository, das Parsing und die Erstellung der Assoziationsregeln, dann zeigt sich die Notwendigkeit für unterschiedliche Paketgrößen sehr deutlich. Die kleinsten Pakete benötigen unverhältnismäßig länger für die Verarbeitung als die größeren, dies ist der Tatsache geschuldet, dass der Anteil der Service-Kommunikation und der Startup-Zeit der JVM an der Gesamtlaufzeit hier deutlich höher ist als bei den anderen Paketgrößen.

Interessant ist der Fakt, dass die Pakete mit Größe 25 in diesem Test leicht führen. Die Auswirkungen der verringerten Kommunikation sind hier also deutlich größer als die zu erwartenden Nachladezeiten von Speicherseiten aus dem Swap. Wenn man die Gesamtlaufzeiten der Paketgrößen 10 und 25 direkt vergleicht, so skalieren diese ungefähr mit dem Wert 2.46, also annähernd linear.

Nimmt man die beiden Teile des Geschwindigkeitstests zusammen, so lässt sich festhalten, dass die Paketgröße so gewählt werden kann, dass der zur Verfügung stehende Speicher möglichst optimal ausgenutzt wird, um die Verzögerung durch die Service-Kommunikation möglichst klein zu halten, Selbst wenn dabei zum Teil auf den Sekundärspeicher zurückgegriffen werden sollte.

7.3 Netzerstellung

Die Netzerstellung ist der letzte Bestandteil der automatischen Verarbeitungskette. Hierbei werden die Ergebnisse der vorangegangenen Schritte zusammengebracht und in einer maschinenauswertbaren Form als Ontologie abgelegt. Diese Ontologie unterstützt dann bei der weiteren Arbeit mit dem Wissen aus der Dokumentensammlung.

Bei der Evaluierung des Moduls zur Netzerstellung ist die Geschwindigkeit zu testen, mit der die Entitäten und Relationen in eine Ontologie überführt werden können.

7.3.1 Geschwindigkeit

Als Datenbasis für die Messungen wurde die Dokumentensammlung aus dem WIKINGER-Projekt verwendet. Da die Klassifikation in der Prozesskette der Netzerstellung direkt vorgeschaltet ist, werden für die Transformation der Entitäten und Relationen in die Ontologie ebenfalls die Auswirkungen unterschiedlicher Paketgrößen berücksichtigt. Die Messungen sind über fünf Durchläufe gemittelt, angezeigt sind die Zeiten für die Pakete und runtergerechnet für einzelne Dokumente. Tabelle 7.6 zeigt die Ergebnisse der Geschwindigkeitsmessungen.

Laufzeiten (in s)	Paketgröße			
	5	10	20	25
Paket	5,173	16,255	30,365	46,196
Dokument	1,035	1,626	1,518	1,848

Tabelle 7.6: Laufzeit der Netzerstellung bei verschiedenen Paketgrößen

Bewertung

Die kleinste Paketgröße zeigt die mit Abstand beste Leistung im Feld. Die Dokumentlaufzeiten der anderen Größen liegen zwischen 50% und 80% höher. Das würde für die Verwendung kleinerer Größen bei der Netzerstellung sprechen. Allerdings hängen die Paketgrößen mit den in der Klassifikation verwendeten zusammen – und dort ist die Laufzeit der Dokumente in der kleinsten Paketgröße so deutlich länger als bei den anderen Größen, dass der Vorteil bei der Netzerstellung nicht ins Gewicht fällt. Nimmt man die Werte der Klassifikation und der Netzerstellung zusammen, so haben Dokumente bei der 25er Größe im Schnitt 18,6s Laufzeit, während bei der 5er Größe 30,6s benötigt werden. Es empfiehlt sich also letztlich auch in diesem Fall, die größte Größe zu wählen.

7.4 Zusammenfassung

In diesem Kapitel sind die Module evaluiert worden, die die Kernfunktionalitäten für die Erfüllung der Ziele aus Kapitel 5 zur Verfügung stellen. Untersucht wurden die Relationserkennung, die Relationsklassifikation und die Erstellung des semantischen Netzes auf der Basis der Ergebnisse beiden vorhergehenden Schritte. Dabei standen verschiedene Gesichtspunkte im Fokus: Geschwindigkeit, Qualität und Nutzerfreundlichkeit.

Die Ergebnisse der einzelnen Tests zeigen die Tragfähigkeit des vorgestellten Systems. Die Relationserkennung erzielt zwischen 82 und 86 Prozent F_0-Maß, liefert also den Experten eine sehr gute Vorauswahl der möglichen Relationen innerhalb des untersuchten Korpus. Die Laufzeit der dabei verwendeten Algorithmen erlaubt die Integration in eine interaktive Benutzeroberfläche, sogar auf PC-Systemen, die bereits 6-7 Jahre alt sind. Besagte Oberfläche kann mit sehr wenig Schulungsaufwand eingesetzt werden, da darauf geachtet wurde, nur eine überschaubare Anzahl von Parametern anzubieten, mit denen sich der Erkennungsprozess beeinflussen lässt.

Die Klassifikation trägt die Hauptlast in der automatischen Verarbeitung der Textkorpora. Sie setzt auf den Ergebnissen der automatischen Eigennamenerkennung auf, ist also nicht zuletzt auch von deren Fehlerraten abhängig. Trotzdem zeigen die Tests sehr gute Werte, mit Recall-Werten von mehr als zwei Dritteln der relevanten Relationsmuster, bei einer globalen Präzision oberhalb von 99%. Damit lassen sich große Teile des in den Korpora enthaltenen Wissens automatisch für die Ontologie bereit stellen.

Die für die Klassifikation und die Netzerstellung zusammen benötigte Zeit für die komplette Verarbeitung des WIKINGER-Korpus (160 Bücher) inklusive Vorverarbeitung beträgt nicht einmal eine ganze Stunde. Damit steht einer Anwendung des Systems auf dynamische Datenbestände nichts im Wege, denn diese finden im betrachteten Hauptanwendungsfall in deutlich kleineren Dokumenten statt. Es kann also in recht kurzem Abstand zu den Änderungen reagiert werden, so dass Nutzer nicht lange auf die Reflektion der neuen Situation durch das semantische Netz warten müssen.

Bei größerer Last auf dem System, etwa beim Einsatz als Navigationshilfe in einem Dokumentenserver, können einzelne Komponenten des Systems skaliert werden, was aufgrund der paketweisen Abarbeitung der geänderten Dokumente kein Problem darstellt. Die Netzerstellung ist in diesem Fall der Flaschenhals, diese sollte nicht parallelisiert werden, um umständliche Merge-Operationen zu vermeiden. Angesichts des Geschwindigkeitsvorteils der Netzerstellung gegenüber der Klassifikation (Faktor 10 bis 12) ist dies erst bei mehr als 10 Instanzen der Klassifikation relevant.

Zusammenfassend lässt sich konstatieren, dass die Referenzimplementierung des vorgestellten Systems seinen Anforderungen gerecht wird und die weitestgehend automatische semantische Erschließung digitaler Textsammlungen ermöglicht. Dabei wird bewusst davon Abstand genommen, sprachabhängige Optimierungen vorzunehmen, auch wenn dies wahrscheinlich eine Verbesserung der Klassifikationsqualität bedeuten würde. Allerdings erhöhte das den Aufwand, der für eine Anpassung des Systems an andere Sprachen geleistet werden müsste, deutlich, da zumindest die Komponente für die Vorverarbeitung der Relationsklassifikation komplett ausgetauscht werden müsste.

Kapitel 8

Bewertung und Ausblick

Dieses Kapitel dient zur abschließenden Rückschau auf das im Rahmen der Arbeiten an dieser Dissertation Erreichte, gemessen an den Anforderungen, die in Kapitel 2.3 aufgestellt worden sind. Darüber hinaus haben sich im Verlauf der Arbeiten sowie durch die Referenzimplementierung im WIKINGER-Projekt Themen für interessante wissenschaftliche und software-technische Anschlussarbeiten ergeben, die als solche nicht in den Rahmen dieser Arbeit passten. Diese werden in Kapitel 8.2 näher beleuchtet.

Zu guter Letzt wird ein Ausblick auf weitere Entwicklungen auf dem Gebiet des Semantic Web gegeben, die ihrerseits weitere Impulse für die künftige Weiterentwicklung des Prototypen geben können.

8.1 Einschätzung des Erreichten

Das Ziel dieser Dissertation war die Entwicklung von Algorithmen und Verfahren, mit denen sich Textsammlungen weitestgehend automatisch inhaltlich erschließen lassen sollten. Die Formate für die Ergebnisse der Erschließung entstammen dem Semantic Web, einem Projekt des World Wide Web Consortiums. Durch dessen weltweite Verbreitung wird sichergestellt, dass die Erschließungsergebnisse einer weiten Nutzung zugeführt werden können; zusätzlich sind die Formate offen und rein textbasiert, was gerade für eine nachhaltige Archivierungsstrategie unabdingbar ist.

Eine erste Umschau nach bereits existierenden Software-Tools auf dem Gebiet der Ontologieerstellung offenbarte einen eklatanten Mangel: Die Anwenderzielgruppe dieser Tools sind nicht etwa Domänenexperten, die eine Ontologie über ihren Bestand legen wollen, sondern Ontologie-Experten, die hauptberuflich Ontologien für ihre Kunden erstellen. Diese sind damit aber auch auf die Dienste eines Ontologie-Experten angewiesen, was die (sowieso schon beträchtlichen) Kosten für die Erstellung von Domänen-Ontologien weiter erhöht - weshalb viel zu wenige Sammlungen semantisch maschinenlesbar erschlossen wer-

den. Benötigt werden daher Systeme, die es Domänenexperten ermöglichen, eine Ontologie für ihr Fachgebiet in Eigenregie zu erstellen.

Auf dieser Basis entstanden die Ziele und Anforderungen aus Kapitel 2. Diese skizzieren ein System, mit dessen Hilfe Ontologien weitestgehend automatisch aus Volltextsammlungen erstellt werden können. Das System ist modular gehalten, um verschiedene Einsatzszenarien und Eingabeformate unterstützen zu können. Die Referenzimplementierung wurde im Rahmen des WIKINGER-Projekts (siehe Kapitel 6.1) erstellt und auf seine Tauglichkeit hinsichtlich der Zielvorstellungen evaluiert (Kapitel 7).

Die Ziel- und Anforderungsdefinitionen sind in drei Abschnitte unterteilt: Semiautomatische Netzerstellung, Infrastruktur und Nutzung. Es ist sinnvoll, die Zieldefinition zusammen mit den daraus abgeleiteten Anforderungen zu betrachten, da letztere konkrete Ziele definieren, die das geplante System ausmachen.

8.1.1 Semiautomatische Netzerstellung

Der erste Abschnitt definierte das Kernziel, nämlich die stark automatisierte Erschließung der Inhalte von Textkorpora, in Abhängigkeit der Bedürfnisse der Domänenexperten, die später mit der so erstellten Ontologie arbeiten sollen.

Dieses Ziel kann getrost als erfüllt angesehen werden. Die Ontologie enthält nur diejenigen Konzepte, die von den Experten vorgegeben worden sind und die enthaltenen Relationen sind aus der Menge der automatisch extrahierbaren Relationsmuster der Textsammlung ausgewählt worden. Die eigentliche Erzeugung des semantischen Netzes erfolgt automatisch, lediglich in der Vorverarbeitung gibt es zwei Stellen, an denen die Interaktion der Experten erforderlich ist: Die Annotation von Beispielen für die Entitätssuche und die Auswahl der zu berücksichtigenden Relationen, inklusive deren Benennung. Diese Tätigkeiten werden durch nutzerfreundliche graphische Oberflächen unterstützt, die Einarbeitung der Experten beträgt jeweils ca. eine Stunde.

8.1.2 Infrastruktur

Der zweite Abschnitt der Zieldefinitionen beschäftigte sich mit der Infrastruktur, die ein System wie das geplante benötigen würde. Im Mittelpunkt standen dabei die Erweiterbarkeit und die Performanz des geplanten Systems. Die Erweiterbarkeit eines Software-Systems ist entscheidend von dessen Architektur abhängig, weshalb nach Untersuchung verschiedener Architekturoptionen eine Entscheidung für die Verwendung von Web Services fiel, da diese ein hohes Maß an Flexibilität bieten und sich auch nachträglich noch neue Funktionalitäten in ein laufendes System integrieren lassen. Dieser Vorteil überwiegt den Nachteil der gesteigerten Komplexität in der Definition der Austauschformate zwischen den einzelnen Services, zumal es für diese Aufgabe mittlerweile Tools gibt, die einen Großteil der Arbeit

abfedern.

Da logisch zusammengehörende Schritte in jeweils eigene Web Services gebündelt werden können, ist auch die Ersetzung kompletter Module durch solche mit besseren, bzw. anderen Algorithmen vereinfacht, solange diese auf den gleichen Programmierschnittstellen aufbauen. Dies erleichtert die Wartung bestehender Services beträchtlich, da nicht das gesamte System gestoppt werden muss, wenn ein Service ausgetauscht werden muss.

In Sachen Performanz der Algorithmen sei auf die Evaluierung der Laufzeiten der einzelnen Komponenten in Kapitel 7 verwiesen. Die dort gemessenen Laufzeiten beziehen sich auf den Fall, dass alle Dienste auf einer Maschine installiert sind. Will man weitere Geschwindigkeitszuwächse erzielen, so lassen sich die Arbeiten auch auf mehrere, spezialisierte Maschinen verteilen.

Auch für das Gebiet der Infrastruktur lässt sich festhalten, dass die Ziel erfüllt worden sind. Für die Realisierung eines erweiterbaren Systems zur Erstellung und Verwaltung von Ontologien für Volltextarchive hat sich die Verwendung eines Web Service-basierten Systems als die beste Option herausgestellt. Zudem ergab die Evaluierung der entwickelten Algorithmen, dass deren Laufzeit sich in einem Rahmen bewegt, der auch die Verarbeitung größerer Textcorpora erlaubt, zumal relativ einfach mehrere Instanzen der jeweiligen Web Services verwendet werden können.

8.1.3 Nutzung

Diesem Bereich sind zwei Ziele der Dissertation zugeordnet: Die Ermöglichung der semantischen Suche und die Bearbeitung dynamischer Korpora. Die semantische Suche bedient sich der Anfragesprache SPARQL, deren Komplexität allerdings die Entwicklung vereinfachter Oberflächen erforderlich macht, damit auch Laien Anfragen stellen können. Dazu sind verschiedene Ansätze entwickelt worden, von denen einer (die formularbasierte semantische Suche) exemplarisch im Rahmen einer Bachelorarbeit realisiert worden ist.

Neben der Komplexität der Anfragesprache ist allerdings auch die Geschwindigkeit, in der die Anfragen bearbeitet werden können, ein kritischer Faktor für die tatsächliche Anwendbarkeit der Technologie. Hier brachte der Einsatz des in Kapitel 5.4.2 erwähnten spezialisierten Datenbankschemas der HP Labs auch in lokalen Messungen eine deutliche Geschwindigkeitssteigerung, mit der Anfragen auch in für Webseiten typischen Zeiträumen beantwortet werden können.

Darüber hinaus hat die Verwendung der vom W3C entwickelten Referenzanfragesprache weitere Vorteile: Über den Suchserver lassen sich recht einfach Daten für eine Verwendung in spezialisierten Visualisierungstools extrahieren und in die benötigte Form bringen. Die Spezifikation des XML-Ausgabeformats ist Bestandteil von SPARQL, eine Umwandlung in ein beliebiges Format lässt sich also mittels XSLT verhältnismäßig einfach durchführen. So lässt sich zum Beispiel ein Dienst aufsetzen, der Daten aus verschiedenen Quellen, unter

anderem einem WIKINGER-System, zieht, aggregiert und dann in einer ansprechenden Form aufbereitet zur Verfügung stellt. Alles ohne, dass das WIKINGER-System speziell darauf angepasst werden muss.

Mit dem Referenzsystem lassen sich auch dynamische Korpora verarbeiten, die dafür benötigten Funktionalitäten sind alle bereits Teil der Architektur: Dokumente werden versioniert vorgehalten, zusätzliche Metadaten werden daher mit Bezug auf eine bestimmte Dokumentenversion erfasst. Das semantische Netz enthält zu jeder Relation einen Verweis auf das Dokument, dem diese entstammt, so dass nach der Verarbeitung einer neueren Version verglichen werden kann, welche Inhalte des Netzes noch aktuell sind, welche geändert bzw. gelöscht werden müssen und welche neuen Relationen in das Netz aufgenommen werden sollten. Diese Funktionalitäten lassen sich alle mit den entwickelten Services abdecken, so dass der Verarbeitung dynamischer Korpora nichts im Wege steht.

Zusammenfassend lässt sich also auch für die Zielvorgaben aus dem Bereich Nutzung festhalten, dass sie durch das vorliegende System erreicht worden sind.

8.1.4 Zusammenfassung

Die vorhergehenden Abschnitte haben die Zielvorgaben mit den erreichten Ergebnissen verglichen und dabei festgestellt, dass die Referenzimplementierung den Anforderungen genügt und zur Erfüllung der Ziele geeignet ist.

Wichtiger als die reine Erfüllung der Anforderungen ist jedoch, die Qualität der Ergebnisse einzuschätzen. Die im vorigen Kapitel beschriebene Evaluierung hat hier eindeutige Ergebnisse geliefert. In den verschiedenen Arbeitsschritten wurden jeweils gute Ergebnisse erzielt. Hierbei besonders erfreulich ist die exzellente Präzision, mit der die automatische Klassifikation arbeitet. Die Ausbeute ist noch steigerungsfähig, aber die Anzahl von false positives ist vernachlässigbar gering. Die Steigerung des Recall-Werts ist jedoch nicht ohne Weiteres ohne die Verwendung weiteren Vorwissens über die verwendete Sprache und ihrer Eigenheiten zu erzielen, wodurch das System aber auch Gefahr läuft, auf die Eigenheiten einer bestimmten Sprache hin optimiert zu werden. Innerhalb des WIKINGER-Konsortiums bestand denn auch die Übereinkunft, mit möglichst wenig Vorwissen auszukommen, um dem System eine möglichst breite Einsatzspanne zu erhalten. Die mit den rein statistisch arbeitenden Algorithmen erzielten Ergebnisse halten jedoch dem direkten Vergleich mit anderen Arbeiten, z.B. die in 4.3.1 erwähnte Arbeit von Hasegawa et al., locker Stand - zumal in der vorliegenden Arbeit nicht nur binäre Relationen berücksichtigt werden.

Zusammenfassend lässt sich also festhalten, dass das vorliegende System in der Lage ist, weitreichende Unterstützung bei der inhaltlichen Erschließung digital vorliegender Textsammlungen zu leisten.

8.2 Anschlussmöglichkeiten

Bei den Arbeiten an den Algorithmen und Verfahren des Dissertation sind eine Reihe interessanter weiterer Forschungsfragen aufgedeckt worden, die allerdings im Rahmen dieser Arbeit nicht weiter verfolgt werden konnten. Ebenso sind bei den Arbeiten am WIKINGER-Prototypen Ideen für weitere Module und Dienste der Plattform entstanden, die unter Umständen noch in der Laufzeit des Projekts realisiert werden können.

Dieser Abschnitt führt die wissenschaftlichen Fragestellungen und Service-Ideen auf, deren Beantwortung bzw. Realisierung in näherer Zukunft angegangen werden sollen, sei es im Rahmen anderer Projekte oder im Rahmen von Graduierungsarbeiten.

8.2.1 Wissenschaftliche Anschlussmöglichkeiten

Im wissenschaftlichen Bereich sind drei besonders vielversprechende Anschlussarbeiten identifiziert worden, ihr Bandbreite reicht von der Ausweitung auf andere Inputarten bis hin zu Strukturierungsalgorithmen in der resultierenden Ontologie. Darüber hinaus sind weitere Arbeiten in anderen Richtungen denkbar, etwa Ontology Matching zum Verbinden mit Ontologien anderer Einrichtungen.

Auswertung relationaler Datenbanken

Da ist zunächst der Import der Daten relationaler Datenbanken. Diese halten die Spezifika der einzelnen Datensätze in einem Entity-Relation Schema fest, das sowohl die einzelnen Attribute von Entitäten, als auch die Beziehungen verschiedener Entitäten untereinander spezifiziert. Das ER-Schema definiert damit eine eigene Ontologie.

Die Aufgabe ist damit, die Teile des ER-Schemas zu identifizieren, deren Berücksichtigung in der zu erstellenden Ontologie den größten Nutzen verspricht. Dieses Problem fällt in den Bereich des Ontology Mapping, das sich auf das Graph Matching Problem zurückführen lässt und in seiner generellsten Form NP-vollständig ist. Eine komplette Automatisierung ist damit illusorisch, ein gewisser Grad manueller Zuordnung der einzelnen Entitätstypen mit Auswahl der relevanten Attribute ist also nicht zu vermeiden. Die Aufgabe teilt sich damit in die algorithmische Bearbeitung des partiellen Mappings der beiden Ontologien und die Erstellung einer graphischen Oberfläche, die manuelle Zuordnungen erlaubt.

Die Vorteile des Datenbankimports für die Arbeitsabläufe des vorgestellten Systems sind an zwei Stellen zu sehen: Die Eigennamenerkennung erhält eine Vielzahl gesicherter Entitätslabels, wodurch sich die Trainingsphase verkürzen lässt, zudem sind die Label üblicherweise bereits in einer normalisierten Form in der Datenbank gespeichert. Daneben liefert das ER-Schema noch Informationen über existierende Relationen zwischen verschie-

denen Entitätstypen, wodurch weitere Hinweise für die Relationserkennung zur Verfügung gestellt werden.

Die Bearbeitung dieser Aufgabe wird im weiteren Verlauf des WIKINGER-Projekts begonnen und vermutlich im Anwendungsszenario CONTENTUS[1] des vom BMWI geförderten THESEUS-Programms[2] weiter bearbeitet werden.

Relationserkennung in Fließtexten unter Sprachberücksichtigung

Im vorliegenden System wurde die Relationserkennung mit rein statistischen Mitteln durchgeführt, sprachliche Besonderheiten wurden nicht berücksichtigt, auch wenn diese dazu angetan wären, die Erkennungsergebnisse zu verbessern. Im Rahmen einer Magisterarbeit an der Universität Bonn wird momentan untersucht, inwiefern sich die Anwendung computerlinguistischer Verfahren auf die Erkennungsergebnisse auswirkt. Für diese Arbeit wird der TIGER-Korpus[3] verwendet, da dieser umfangreich computerlinguistisch getagged ist. Zum Labeling gefundener Relationen kann unter Umständen das Wortschatz-Projekt der Universität Leipzig[4] zum Einsatz kommen, das Wortinformationen für die deutsche Sprache in der Form von Web Services zur Verfügung stellt, alternativ aber auch das GermaNet. Daneben ist von Interesse, inwieweit sich die Performanz der bereits bestehenden Algorithmen ändert, wenn eine sprachabhängige Vorverarbeitung durchgeführt wird.

Einer der Vorteile dieser Vorverarbeitung ist der, dass die Chance besteht, synonym verwendete Verben zu erkennen und somit einerseits Relationen zusammenfassen zu können, andererseits aber auch den Recall insgesamt zu erhöhen, da völlig neue Muster Berücksichtigung finden können. Zudem ist eine Dimensionsverringerung der Wortvektoren zu erwarten, wenn Worte normiert und auf ihre Stammform zurückgeführt werden. Dies sollte die Speicheranforderungen der bereits bestehenden Algorithmen zusätzlich verringern.

Die Magisterarbeit ist im Dezember 2007 angemeldet worden, geplante Abgabe ist zur Jahresmitte 2008.

Subgruppenerkennung in Instanzlabels

Im heutigen System werden lediglich die von den Experten ausgewählten Entitätsklassen zur Strukturierung der Inhalte verwendet, d.h. innerhalb dieser Klassen gibt es keine weitere Unterteilung. Für die Klasse *Person* ist das genau das richtige Vorgehen, Subgruppen auf der Basis der Namenssyntax würden auch keinen Sinn ergeben. Anders sieht das bei Klassen aus, in denen Attribute aufgezählt werden, zum Beispiel für die Klasse *Rolle*. Diese sind momentan ebenfalls flach organisiert, um die Eigennamenerkennung zu vereinfachen,

[1] Siehe http://www.theseus-programm.de/scenarios/de/contentus
[2] Siehe http://www.theseus-programm.de
[3] Siehe `http://www.ims.uni-stuttgart.de/projekte/TIGER/TIGERCorpus/`
[4] wortschatz.uni-leipzig.de

allerdings lassen sich in den Entitätslisten mitunter mögliche Subgruppen innerhalb der Klasse zu entdecken.

So gibt es im WIKINGER-Korpus verschiedenste Arten von Abgeordneten in der Klasse *Rolle*. Diese ließen sich gut in einer eigenen Subklasse der Klasse Rolle unterbringen. Diese Art der Strukturierung kann anhand des Korpus vorgenommen werden und bietet sich für eine Reihe weiterer Beispiele innerhalb der Klassen an: Es gibt verschiedene Arten von Präsidenten, Professoren sowie Vorsitzenden innerhalb der Klasse Rolle; in der Klasse der geo- und kirchenpolitischen Entitäten (GKPE) sind einige Städte samt einiger Stadtteile im Datenmaterial enthalten und auch einige Unterteilungen verschiedener Organisationen ließen sich unter einem gemeinsamen Konzept fassen. Daneben können aber auch Rangsysteme genutzt werden, die komplett in die Ontologie übertragen werden. Passende Entitäten könnten dann mit diesen Systemen abgeglichen und in die entsprechenden Subgruppen eingetragen werden.

Wenn die Erkennung und Verarbeitung dieser Subgruppen hinreichend automatisiert werden könnte, ließe sich der Strukturierungsgrad der Ontologien deutlich steigern, was sich positiv auf die Anfragebeantwortung niederschlüge. Natürlich sollten aber die Domänenexperten die Entscheidung über einzuführende Subklassifizierungen treffen.

Die Untersuchung der hierfür benötigten Mechanismen und Nutzeroberflächen, sowie weiterführend, inwiefern sich diese automatisch generierten Subgruppen auch in passenden Subrelationen niederschlagen sollten, bietet sich für eine oder mehrere Graduierungsarbeiten an.

8.2.2 Ausbaumöglichkeiten für die Plattform

Die vorliegende Referenzimplementierung ist auf den Anwendungsfall des WIKINGER-Projekts abgestimmt. In diesem Abschnitt werden Entwicklungsrichtungen für die Plattform beleuchtet, die das Funktionsspektrum des Systems erweitern und es auch für andere Anwendungsfälle einsetzbar machen.

Visualisierungen

Bisher werden die Daten der Ontologie allein in einer Textansicht präsentiert. Dies ist zwar die Visualisierungsform, die von den meisten Nutzern bevorzugt wird, allerdings bietet es sich an, für bestimmte Subsets der Daten über alternative Visualisierungen nachzudenken. Dies betrifft besonders zeit- und ortsbezogene Informationen. Für erstere bietet sich die Visualisierung in Form eines Zeitstrahls an, hierzu gibt es bereits interessante Module, zum Beispiel im Rahmen des SIMILE-Projekts des MIT. Dort ist das Modul „Timeline“ entstanden, eine konfigurierbare Zeitstrahlvisualisierung auf der Basis von DHTML und AJAX, die sich auch in andere Projekte leicht einbinden lässt.

Für die Visualisierung geographischer Daten bieten sich natürlich Kartendarstellungen an, in die weitere Informationen integriert werden können. Neben Ortsinformationen könnte die Größe eines Punktes auf der Karte etwa die Anzahl der mit diesem Ort verbundenen Entitäten anzeigen oder Orte könnten durch Linien verbunden werden, wenn es in der Ontologie entsprechende Verknüpfungen geben sollte. Verbindet man die beiden Formen der Visualisierung, könnte auf einer Karte etwa der Lebensweg einer Person nachgezeichnet werden.

Es ist schwer, für stark domänenspezifische Anwendungen generische, interessante und außerdem nützliche Visualisierungen zu schaffen, da allerdings Orte und Datumsangaben in vielen Ontologien immer wieder eine Rolle spielen dürften, kann man mit diesen beiden speziellen Visualisierungen nichts verkehrt machen.

Nutzerseitig erzeugte Verknüpfungen zulassen

Das Nutzungsszenario des WIKINGER-Projekts geht davon aus, dass die Relationen der Ontologie einmal auf Serverseite durch Experten festgelegt werden und nur von diesen geändert werden können. Für verschiedene Anwendungsfälle ist es jedoch sinnvoll, von Nutzern generierte Verknüpfungen zuzulassen, etwa beim Einsatz der Plattform zur Strukturierung eines Dokumentenservers. Gerade dort ist es unerlässlich, den Nutzern die Möglichkeit zu geben, zusätzlich zu einer global vorgegebenen auch noch ihre eigene Strukturierung über den Bestand legen zu können. Dies macht komplexere Nutzerschnittstellen erforderlich. Zusätzlich kann darüber nachgedacht werden, Nutzern die Möglichkeit zu geben, ihre Strukturierungen mit anderen Nutzern zu teilen. Diese Anwendungsfälle werden für das Projekt CONTENTUS antizipiert, inwieweit diese umgesetzt werden können, ist allerdings noch offen.

Multimediale Daten

Momentan wird nur auf Textdaten gearbeitet. Zukünftig werden aber insbesondere auch Ton- und Videoarchive verstärkt in den Fokus rücken. Die Vermarktung solcher Archive verspricht ein noch größeres Geschäft als die Vermarktung von Textarchiven. Gleichzeitig sind diese Archive aber schwerer inhaltlich zu erschließen als Textarchive. Von daher ist es sinnvoll, das vorliegende System um Funktionalitäten für die Verarbeitung multimedialer Daten zu erweitern. Im Audiofall ist das Problem im Wesentlichen auf das der Textarchive zurückführbar, jedenfalls dann, wenn Sprache in einem verwertbaren Transkript vorliegt. Paragraphen werden dann nicht mehr Seiten in einem Buch, sondern Tonsegmenten auf einem Band oder einer Tondatei zugeordnet, ansonsten ändert sich an der Verarbeitung nicht viel.

Anders sieht das bei Videoarchiven aus. Neben der Tonkomponente kommt hier erschwerend hinzu, dass eigentlich für eine komplette Erschließung eines Films alle Objekte erfasst

werden müssten, die in irgendeinem der Bilder zu sehen sind - da im Vorhinein nicht bekannt ist, wonach einmal gesucht werden soll. Erschwerend kommt hinzu, dass in Archiven der Sendeanstalten oftmals nach Sequenzen gesucht wird, die eine bestimmte Stimmung transportieren, um damit andere Sendeinhalte zu untermalen. Diese Klassifikation muss die automatische Verarbeitung einstweilen schuldig bleiben. Das WIKINGER-Framework ist für solche Anwendungen also sowohl auf die Ergebnisse der automatischen low-level Verarbeitung, als auch auf semantisch höherwertige manuelle Klassifikationen angewiesen. Hier wird der Bogen zum vorhergehenden Erweiterungspunkt geschlagen, da dieser eine bedarfsgetriebene Erschließung des Archivmaterials befördert.

Zumindest für die Notation der automatisch erzielbaren Ergebnisse gibt es verschiedene Formate, am vielversprechendsten scheint momentan MPEG7, ein Metadatenformat für Audio- und Videodaten. Das macht die Integration einer Verarbeitungslinie für MPEG-Dateien zu einer interessanten Erweiterung für die WIKINGER-Plattform.

8.3 Ausblick

Die Bemühungen um das Semantic Web sind in den letzten Jahren einen großen Schritt weiter gekommen. Mit der Verabschiedung von SPARQL als Empfehlung des W3C am 15.1.2008 ist auch der letzte Baustein der mittleren Schicht des Semantic Web Stacks fertig [89]. Zusammen mit den übrigen bereits verabschiedeten Spezifikationen können Ontologien somit für die Verwaltung und den Zugriff auch umfangreicher Datenbestände genutzt werden. Die noch fehlenden großen Teile des Projekts, *Reasoning* und *Trust*, sind vor allem für die Realisierung autonom agierender Software-Agenten notwendig. Solange es allerdings keine großen, öffentlich verfügbare kommerziell eingesetzte Ontologien gibt, sind diese Agenten eher Gegenstand der wissenschaftlichen Debatte, denn Werkzeuge für das tägliche Leben.

Mit den bereits verfügbaren Werkzeugen kann hingegen schon heute sinnvoll gearbeitet werden, um Datenbestände inhaltlich zu erschließen und so für Anwendungen in Inter- oder Intranet zur Verfügung zu stellen. Dank der *open world assumption* kann dies sogar über die Grenzen einer einzelnen Einrichtung hinweg erfolgen, wodurch diese in die Lage versetzt werden, ihre Inhalte mit den Daten der jeweils anderen Einrichtungen zu vernetzen.

Damit wird die Achillesferse des Semantic Web berührt: Die Akzeptanz der neuen Technologien steht und fällt mit der Menge an verfügbaren Inhalten. Hier sieht es momentan noch nicht so gut für die Zukunft des Semantic Web aus, da der Großteil der Anstrengungen der wissenschaftlichen Gemeinde auf die Weiterentwicklung und Definition der benötigten Sprachen konzentriert ist. Außerhalb der beteiligten Fachdisziplinen ist der Nutzen des Semantic Web hingegen noch nicht so bekannt wie es wünschenswert wäre, was nicht zuletzt auch daran liegt, dass es kaum Tools gibt, mit denen auch Laien eigene Inhalte passend annotieren können. Der überwältigende Erfolg des WWW ist schließlich auch darauf zurück

zu führen, dass Webseiten mit einfachen Texteditoren unter Kenntnis eines guten Dutzend Markupelementen erstellt werden konnten. Dieser leichte Zugang ist es, der dem Semantic Web momentan fehlt.

Aufgrund der gesteigerten Komplexität - damals Markup von Texten zur strukturierten Darstellung im Web, heute semantisches Markup von Textbestandteilen nach den Regeln der verschiedensten Ontologien - wird die Erstellung von Inhalten für das Semantic Web nie so einfach sein wie im Fall des WWW, allerdings gibt es Potenziale für die maschinelle Unterstützung bei der Inhaltserzeugung. Dieser Forschungsrichtung wird nach Auffassung des Autors jedoch nach wie vor zu wenig Aufmerksamkeit der wissenschaftlichen Community zuteil. Vor diesem Hintergrund entstanden die Idee zu dieser Dissertation und zu dem sie flankierenden Forschungsprojekt WIKINGER. Die darin entwickelte Software-Plattform zeigt, dass die Ontologieerstellung soweit automatisiert werden kann, dass auch Nicht-Informatiker in die Lage versetzt werden, semantische Beschreibungen ihrer Domänen zu verfassen und auf Datenbestände anzuwenden.

Die weitere Verbesserung der dabei eingesetzten Algorithmen ist sicherlich wünschenswert, zum Beispiel, um die Plattform für den Einsatz der kommenden Bausteine des Semantic Web Stacks vorzubereiten oder um die Ausbeute bei den Relationen weiter zu erhöhen. So wie sie jetzt ist, ist die Plattform jedoch bereits einsetzbar, um verschiedenste Fachkorpora für die Verwendung in Semantic Web Anwendungen aufzubereiten.

Die Weiterentwicklungsoptionen der vorliegenden Arbeit sind in den vorangegangenen Kapiteln bereits ausgeführt worden. Die hier entwickelten Algorithmen werden in weitere Forschungsprojekte des IAIS einfließen, zum Beispiel in CONTENTUS, einem Unterprojekt des vom BMWI geförderten Projekts THESEUS. CONTENTUS wird zusammen mit der Deutschen Nationalbibliothek (DNB) bearbeitet. Teile der WIKINGER-Plattform werden dabei für die inhaltliche Erschließung verschiedener Korpora der DNB verwendet werden. Es ist vorgesehen, die Erschließung auch auf Audiodokumente auszuweiten, um eine kombinierte inhaltliche Suche auf den verschiedenen in den Fachkorpora vorhandenen Formaten anbieten zu können. Perspektivisch sollen die im Rahmen von CONTENTUS entwickelten Technologien für die Erschließung des Gesamtbestands der DNB eingesetzt werden.

Mit der Aufbereitung von Teilen der Bestände der DNB werden Inhalte für das Semantic Web erzeugt, die auch für breite Nutzerschichten interessant sind. Gekoppelt mit einer intelligenten Oberfläche, die auch komplexere semantische Suchen erlaubt, kann die Sichtbarkeit der Vorteile des Semantic Web deutlich erhöht und so einer potenziell stärkeren Verbreitung entsprechender Inhalte Vorschub geleistet werden.

Literaturverzeichnis

[1] Semantic Web Activity Statement. Activity statement, World Wide Web Consortium, 2001. Erhältlich unter `http://www.w3.org/2001/sw/Activity.html`.

[2] Apache Axis, `http://ws.apache.org/axis/`.

[3] Apache Lucene, `http://lucene.apache.org/`.

[4] Apache ObJectRelational Bridge, OJB, `http://db.apache.org/ojb/`.

[5] Aron Culotta and Andrew McCallum and Jonathan Betz. Integrating Probabilistic Extraction Models and Data Mining to Discover Relations and Patterns in Text. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, 2006.

[6] Aron Culotta and Jeffrey Sorensen. Dependency Tree Kernels for Relation Extraction. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 423–429, 2004.

[7] S. Auer, C. Bizer, G. Kobilarow, J. Lehmann, R. Cyganiak, and Z. Ives. DBPedia: A Nucleus for a Web of Open Data. In *Proceedings of the 6th International Semantic Web Conference (ISWC)*, 2007.

[8] David Aumüller and Sören Auer. Towards a semantic wiki experience – desktop integration and interactivity in WikSAR. In *Proc. of 1st Workshop on The Semantic Desktop - Next Generation Personal Information Management and Collaboration Infrastructure, Galway, Ireland, Nov. 6th, 2005*, 2005.

[9] R. Basili, M.T. Pazienza, and P. Velardi. An Empirical Symbolic Approach to Natural Language Processing. *Artificial Intelligence*, (85):59–99, 1996.

[10] Sean Bechhofer, Ian Horrocks, Carole Goble, and Robert Stevens. OilEd: A Reasonable Ontology Editor for the Semantic Web. volume 2174 of *Lecture Notes in Computer Science*, pages 396–408, 2001.

[11] Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuiness, Peter F. Patel-Schneider, and Lynn Andrea Stein. OWL Web Ontology Language Reference. Recommendation, World Wide Web Consortium, Februar 2004. Erhältlich unter `http://www.w3.org/TR/owl-ref/`.

[12] Kent Beck and Cynthia Andres. *Extreme Programming Explained: Embrace Change, 2nd Ed.* Addison-Wesley, 2004.

[13] Dave Beckett and Jeen Broekstra. SPARQL Query Results XML Format. Recommendation, World Wide Web Consortium, Januar 2008. Erhältlich unter `http://www.w3.org/TR/rdf-sparql-XMLres/`.

[14] Tim Berners-Lee. Primer: Getting into RDF and Semantic Web using N3. Technical report, World Wide Web Consortium, 2000. Erhältlich unter `http://www.w3.org/2000/10/swap/Primer.html`.

[15] Tim Berners-Lee, Roy Fielding, and Larry Masinter. RFC 2396 - Uniform Resource Identifiers (URI): Generic Syntax. RFC, IETF, August 1998.

[16] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, pages 216–218, May 2001.

[17] Dan Brickley and R. V. Guha (Eds). RDF Vocabulary Description Language 1.0: RDF Schema. Recommendation, World Wide Web Consortium, Februar 2004. Erhältlich unter `http://www.w3.org/TR/rdf-schema/`.

[18] Lars Bröcker. Wikis als Mittel zur Ontologieverfeinerung. In *Informatik 2005 - Informatik LIVE - Band 2, Beiträge der 35. Jahrestagung der Gesellschaft für Informatik e.V. (GI)*, volume p-68 of *Lecture Notes in Informatics*, pages 119 – 123. Gesellschaft für Informatik e.V., 2005.

[19] Lars Bröcker. WIKINGER - Semantically Enhanced Knowledge Repositories for Scientific Communities. *ERCIM News*, 66:50 – 51, 2006.

[20] Lars Bröcker. The WIKINGER Project - Knowledge Capturing for Domain Experts. *ERCIM News*, 72:20 – 21, 2008.

[21] Lars Bröcker and Stefan Paal. Filtering Internet News Feeds using Ad-Hoc Ontologies. In *On the Move to Meaningful Internet Systems 2006: OTM Workshops*, volume 4277 of *Lecture Notes in Computer Science*, pages 44 – 45. Springer, 2006.

[22] Lars Bröcker and Stefan Paal. Semantic Filtering of Internet News Feeds. In *Proceedings of IADIS International Conference WWW/Internet 2006, Murcia, Spain.* IADIS Press, 2006.

[23] Lars Bröcker, Stefan Paal, Marc Rössler, Andreas Wagner, Wolfgang Hoeppner, Andreas Burtscheidt, and Bernhard Frings. Wikinger - Wiki Next Generation Enhanced Repositories. In *Online Proceedings of the 1st German E-Science Conference*. Max-Planck-Gesellschaft, 2007. Erhältlich unter `FIXME`.

[24] Lars Bröcker, Marc Rössler, and Andreas Wagner. Knowledge Capturing Tools for Domain Experts. In *Proceedings of the Semantic Authoring, Annotation and Knowlege Markup Workshop (SAAKM2007) located at the 4th International Conference on Knowledge Capture (KCAP07)*, volume 289 of *CEUR Workshop Proceedings*. CEUR, 2007. erhältlich unter: `http://ceur-ws.org/Vol-289`.

[25] Bundesministerium für Bildung und Forschung. Bekanntmachung des Bundesministeriums für Bildung und Forschung von Richtlinien für die Förderung zu "e-Science und vernetztes Wissensmanagement". Bekanntmachung, BMBF, November 2004. Siehe `http://www.bmbf.de/foerderungen/3179.php`.

[26] Bundesministerium für Bildung und Forschung. Bekanntmachung über die Förderung von Forschungsvorhaben auf dem Gebiet "e-Science und Grid-Middleware zur Unterstützung wissenschaftlichen Arbeitens" im Rahmen der deutschen D-Grid-Initiative. Call 2004: "Community-Grids" und "Grid-Middleware-Integrationsplattform". Bekanntmachung, BMBF, August 2004. Siehe `http://www.pt-it.de/in/escience/docs/E-science_Call04_PTIN.pdf`.

[27] Stefano Emilio Campanini, Paolo Castagna, and Roberto Tazzoli. Platypus wiki: a semantic wiki wiki web. In *Semantic Web Applications and Perspectives, Proceedings of 1st Italian Semantic Web Workshop*, DEC 2004.

[28] Jorge Cardoso. The Semantic Web Vision: Where Are We? *IEEE Intelligent Systems*, 22(5):84–88, 2007.

[29] Jeremy J. Carroll and Jos De Roo. OWL Web Ontology Language Test Cases. Recommendation, World Wide Web Consortium, Februar 2004. Erhältlich unter `http://www.w3.org/TR/owl-test/`.

[30] Vinay K. Chaudhri, Adam Farquhar, Richard Fikes, Peter D. Karp, and James Rice. OKBC: A Programmatic Foundation for Knowledge Base Interoperability. In *Proceedings of the 15th National Conference on Articial Intelligence (AAAI-98) and of the 10th Conference on Innovative Applications of Articial Intelligence (IAAI-98)*, pages 600–607. AAAI Press, 1998.

[31] R. Chinnici, J.-J. Moreau, A. Ryman, and S. Weerawarana. Web Services Description Language WSDL 2.0: Core Language. Recommendation, World Wide Web Consortium, Juni 2007. Erhältlich unter `http://www.w3.org/TR/2007/REC-wsdl20-20070626/`.

[32] Philipp Cimiano and Johanna Völker. Text2Onto - A Framework for Ontology Learning and Data-driven Change Discovery. In *Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems (NLDB)*, 2005.

[33] Kendall Grant Clark, Lee Feigenbaum, and Elias Torres. SPARQL Protocol for RDF. Recommendation, World Wide Web Consortium, Januar 2008. Erhältlich unter `http://www.w3.org/TR/rdf-sparql-protocol/`.

[34] Hamish Cunningham. GATE, a General Architecture for Text Engineering. *Computers and the Humanities*, 36:223 – 254, 2002.

[35] Ward Cunningham and Kent Beck. A Laboratory For Teaching Object-Oriented Thinking. *SIGPLAN Notices*, 24(10), 1989.

[36] D-Grid-Initiative. e-Science in Deutschland: F&E-Rahmenprogramm 2005 bis 2009. Strategiepapier, D-Grid-Initiative, 2004. Zu finden unter `http://grid.desy.de/d-grid/RahmenprogrammEndfassung.pdf`.

[37] Hasan Davulcu, Srinivas Vadrevu, and Saravanakumar Nagarajan. OntoMiner: Bootstrapping Ontologies from Overlapping Domain-Specific Web Sites. In *Proceedings of the 13th International world Wide Web Conference, May 17-22 2004, NY*, pages 50 – 502. ACM, 2004.

[38] Hasan Davulcu, Srinivas Vadrevu, S. Natarajan, and I.V. Ramakrishnan. OntoMiner: Bootstrapping and Populating Ontologies from Domain-Specific Web Sites. *IEEE Intelligent Systems*, 18(5):23 – 33, 2003.

[39] Deborah. L. McGuiness . *Ontologies Come of Age*, chapter 6, pages 171–194. In [40], 2003.

[40] Dieter Fensel and James Hendler and Henry Lieberman and Wolfgang Wahlster [Eds]. *Spinning the Semantic Web: Bringing the World Wide Web to its Full Potential*. MIT Press, 2003.

[41] Homepage: `http://www.docbook.org`.

[42] Dave Beckett (Ed). RDF/XML Syntax Specification (Revised). Recommendation, World Wide Web Consortium, Februar 2004. Erhältlich unter `http://www.w3.org/TR/rdf-syntax-grammar/`.

[43] Nilo Mitra (Ed). SOAP Version 1.2 Part 0: Primer. Recommendation, World Wide Web Consortium, Juni 2003. Erhältlich unter `http://www.w3.org/TR/soap12-part0/`.

[44] Patrick Hayes (Ed). RDF Semantics. Recommendation, World Wide Web Consortium, Februar 2004. Erhältlich unter `http://www.w3.org/TR/rdf-mt/`.

[45] Steffen Staab (Ed). Why evaluate Ontology Technologies? Because it Works! *IEEE Intelligent Systems*, 19(4):74 – 81, 2004.

[46] April 2005. Brief der Staatschefs von F, PL, I, ES, HU, D an die EU-Kommission bzgl. Einrichtung einer Europäischen Digitalen Bibliothek: `http://europa.eu.int/information_society/activities/digital_libraries/% doc/letter_1/index_en.htm`.

[47] J. Farrell and H. Lausen. Semantic Annotations for WSDL and XML Schema. Recommendation, World Wide Web Consortium, August 2007. Erhältlich unter `http://www.w3.org/TR/sawsdl/`.

[48] J. Ferraiolo, J. Fujisawa, and D. Jackson. Scalable Vector Graphics (SVG) 1.1 Specification. Recommendation, World Wide Web Consortium, Januar 2003. Erhältlich unter `http://www.w3.org/TR/SVG11/`.

[49] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California at Irvine, 2000. Online verfügbar unter `http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm`.

[50] Jan Grant and Dave Beckett (Eds). RDF Test Cases. Recommendation, World Wide Web Consortium, Februar 2004. Erhältlich unter `http://www.w3.org/TR/rdf-testcases/`.

[51] T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5:199–220, 1993.

[52] Nicola Guarino and Christopher Welty. Evaluating ontological decisions with ontoclean. *Communications of the ACM*, 45(2):61–65, 2002.

[53] M. Gudgin, M. Hadley, N. Mendelsohn, J.-J. Moreau, and H. F. Nielsen. SOAP Version 1.2 Part 1: Messaging Framework. Recommendation, World Wide Web Consortium, Juni 2003. Erhältlich unter `http://www.w3.org/TR/soap12-part1/`.

[54] M. Gudgin, M. Hadley, N. Mendelsohn, J.-J. Moreau, and H. F. Nielsen. SOAP Version 1.2 Part 2: Adjuncts. Recommendation, World Wide Web Consortium, Juni 2003. Erhältlich unter `http://www.w3.org/TR/soap12-part2/`.

[55] Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin. An evaluation of knowledge base systems for large owl datasets. Technical report, Lehigh University, Bethlehem, PA. Zu finden unter `http://swat.cse.lehigh.edu/pubs/guo04d.pdf`.

[56] Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin. An evaluation of knowledge base systems for large owl datasets. In *Proceedings of the Third International Semantic Web Conference 2004 (ISWC 2004)*, volume 3298 of *Lecture Notes in Computer Science*, pages 274 – 288. Springer, 2002.

[57] V. Haarslev and R. Moeller. Racer: A core inference engine for the Semantic Web. In *2nd International Workshop on Evaluation of Ontology-based Tools(EON-2003)*, volume 87 of *CEUR Workshop Proceedings*, 2003. Siehe http://ceur-ws.org/Vol-87.

[58] V. Haarslev and R. Moeller. Racer: An OWL reasoning agent for the Semantic Web. In *Proc. of the International Workshop on Applications, Products and Services of Web-based Support Systems, in conjunction with 2003 IEEE/WIC International Conference on Web Intelligence*, pages 91 – 95, 2003.

[59] Jeff Heflin. OWL Web Ontology Language Use Cases and Requirements. Recommendation, World Wide Web Consortium, Februar 2004. Erhältlich unter `http://www.w3.org/TR/webont-req/`.

[60] Johan Hjelm. *Creating the Semantic Web with RDF.* Wiley Computer Publishing, 2001.

[61] ISO 10744:1992 - HyTime. International standard, International Organization for Standardization, 1992.

[62] ISO 13250:2003 - SGML Applications - Topic Maps. International standard, International Organization for Standardization, 2003.

[63] Jena – A Semantic Web Framework for Java, `http://jena.sourceforge.net/`.

[64] Kai-Uwe Carstensen and Christian Ebert and Cornelia Endriss and Susanne Jekat and Ralf Klabunde. *Computerlinguistik und Sprachtechnologie. Eine Einführung.* Spektrum Akademischer Verlag, 2004.

[65] Graham Klyne and Jeremy J. Carroll (Eds). Resource Description Framework (RDF): Concepts and Abstract Syntax. Recommendation, World Wide Web Consortium, Februar 2004. Erhältlich unter `http://www.w3.org/TR/rdf-concepts/`.

[66] Markus Kroetzsch, Denny Vrandecic, and Max Voelkel. Wikipedia and the semantic web - the missing links. In *Proceedings of the WikiMania2005*, 2005.

[67] Carl Lagoze and Herbert Van de Hempel et al. The open archives initiative protocol for metadata harvesting 2.0. Technical report, Open Archives Initiative, OAI. Zu finden unter `http://www.openarchives.org/OAI/openarchivesprotocol.html`.

[68] Alexander Maedche. *Ontology Learning for the Semantic Web.* Kluwer Academic Publishers, 2002.

[69] Alexander Maedche. *The Text-To-Onto Environment*, chapter 7. In [68], 2002.

[70] Alexander Maedche and Steffen Staab. Mining Ontologies from Text. In *Proceedings of the European Knowledge Acquisition Conference (ECAW-2000)*, volume 1937 of *Lecture Notes in Computer Science*, pages 189–202. Springer, 2000.

[71] Alexander Maedche and Steffen Staab. Ontology Learning for the Semantic Web. *IEEE Intelligent Systems*, 16(2):72 – 79, 2001.

[72] Frank Manola and Eric Milles (Eds). RDF Primer. Recommendation, World Wide Web Consortium, Februar 2004. Erhältlich unter `http://www.w3.org/TR/rdf-primer/`.

[73] Marina Reinus. Realisierung einer Benutzerschnittstelle für eine semantische Suchmaschine. Bachelor-arbeit, Fachhochschule Bonn-Rhein-Sieg, 2008.

[74] Daniel Bachlechner Martin Hepp and Katharina Siorpaes. Ontowiki: Community-driven ontology engineering and ontology usage based on wikis. In *Proceedings of the 2005 International Symposium on Wikis (WikiSym 2005)*, 2005.

[75] Daniel Bachlechner Martin Hepp and Katharina Siorpaes. Harvesting wiki consensus - using wikipedia entries as ontology elements. In *Proceedings of the 1st Workshop: SemWiki2006, co-located with ESWC 2006*, 2006.

[76] Brian McBride. Jena: Implementing the RDF Model and Syntax Specification. Technical report, Hewlett-Packard, 2001. Zu finden unter `http://www.hpl.hp.com/personal/bwm/papers/20001221-paper/`.

[77] Deborah L. McGuiness and Frank van Harmelen. OWL Web Ontology Language Overview. Recommendation, World Wide Web Consortium, Februar 2004. Erhältlich unter `http://www.w3.org/TR/owl-features/`.

[78] Medical Subject Headings Thesaurus, United States National Library of Medicine, `http://www.nlm.nih.gov/mesh/`.

[79] M. Missikoff, R. Navigli, and P. Velardi. Integrated approach to Web ontology learning and engineering. *IEEE Computer*, 35(11):60 – 63, 2002.

[80] M. Missikoff, R. Navigli, and P. Velardi. The Usable Ontology: An Environment for Building and Assessing a Domain Ontology. In *Proceedings of the First International Semantic Web Conference 2002 (ISWC 2002)*, volume 2342 of *Lecture Notes in Computer Science*, pages 39 – 53. Springer, 2002.

[81] Roberto Navigli, Paola Velardi, and Aldo Gangemi. Ontology Learning and Its Application to Automated Terminology Translation. *IEEE Intelligent Systems*, 18(1):22 – 31, 2003.

[82] Natalia F. Noy, R. Fergerson, and M. Musen. The Knowledge Model of Protege-2000: Combining Interoperability and Flexibility. In *Proceedings of the 12th International Conference on Knowledge Engineering and Knowledge Management (EKAW) 2000*, volume 1937 of *Lecture Notes in Computer Science*, pages 17 – 32. Springer, 2000.

[83] Natalya F. Noy. Why evaluate ontology technologies? because it works! chapter evaluation by ontology consumers. *IEEE Intelligent Systems*, 19(4):74–81, 2004.

[84] Natalya F. Noy, Michael Sintek, Stefan Decker, Monica Crubezy, Ray W. Fergerson, and Mark A. Musen. Creating Semantic Web Contents with Protege-2000. *IEEE Intelligent Systems*, 16(2):60–71, 2001.

[85] Eyal Oren. Semperwiki: A semantic personal wiki. In *Proceedings of the Semantic Desktop Workshop at the ISWC2005*, 2005.

[86] Eyal Oren, Max Völkel, John G. Breslin, and Stefan Decker. Semantic wikis for personal knowledge management. In *Proceedings of the International Conference on Database and Expert Systems Applications (DEXA)*, 2006.

[87] Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks. OWL Web Ontology Language Semantics and Abstract Syntax. Recommendation, World Wide Web Consortium, Februar 2004. Erhältlich unter `http://www.w3.org/TR/owl-semantics/`.

[88] Steve Pepper. Euler, Topic Maps, and Revolution. In *Proceedings of XML Europe 99 Conference*, 1999.

[89] Eric Prud'hommeaux and Andy Seaborne. SPARQL Query Language for RDF. Recommendation, World Wide Web Consortium, Januar 2008. Erhältlich unter `http://www.w3.org/TR/rdf-sparql-query/`.

[90] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules. In *Proceedings of the 20th VLDB conference*, pages 487–499, 1994.

[91] Razvan C. Bunescu and Raymond J. Mooney. A Shortest Path Dependency Kernel for Relation Extraction. In *Conference on Empirical Methods in Natural Language Processing*, pages 724–731, 2005.

[92] Ruslan Mitkov. Anaphora Resolution: The State of the Art. Technical report, University of Wolverhampton, UK, 1999. Zu finden unter `http://clg.wlv.ac.uk/papers/mitkov-99a.pdf`.

[93] Marc Rössler. *Korpus-adaptive Eigennamenerkennung*. PhD thesis, Universität Duisburg-Essen, 2006. Online verfügbar unter `http://www.duepublico.uni-due.de/servlets/DerivativeServlet/Derivate-1% 6089/diss_final2007_DS.pdf`.

[94] Ali Shiri. Schemas and ontologies, building a semantic infrastructure for the grid and digital libraries. *Library Hi Tech News*, 20(7), 2003.

[95] Derek Sleeman and Quentin Reul. Cleanonto: Evaluating taxonomic relationships in ontologies. In *Proceedings of the 15th international conference on the World Wide Web, WWW2006*, 2006.

[96] Michael K. Smith, Chris Welty, and Deborah L. McGuiness. OWL Web Ontology Language Guide. Recommendation, World Wide Web Consortium, Februar 2004. Erhältlich unter `http://www.w3.org/TR/owl-guide/`.

[97] SNOMED Clinical Terms, United States National Library of Medicine, `http://www.snomed.org/snomedct/index.html`.

[98] Adam Souzis. Rhizome position paper. In *Proceedings of the 1st Workshop on Friend of a Friend, Social Networking and the Semantic Web*, 2004.

[99] Adam Souzis. Building a Semantic Wiki. *IEEE Intelligent Systems*, 20(5):87 – 91, 2005.

[100] Y. Sure, J. Angele, and S. Staab. OntoEdit: Multifaceted Inferencing for Ontology Engineering. In *Journal on Data Semantics*, volume 2800 of *Lecture Notes in Computer Science*, pages 128 – 152. Springer, 2003.

[101] Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer, and D. Wenke. OntoEdit: Collaborative Ontology Development for the Semantic Web. In *Proceedings of the First International Semantic Web Conference 2002 (ISWC 2002)*, volume 2342 of *Lecture Notes in Computer Science*, pages 221 – 235. Springer, 2002.

[102] Y. Sure, S. Staab, and J. Angele. OntoEdit: Guiding Ontology Development by Methodology and Inferencing. In *Proceedings of the International Conference on Ontologies, Databases and Applications of Semantics ODBASE 2002*, volume 2519 of *Lecture Notes in Computer Science*, pages 1205 – 1222. Springer, 2002.

[103] R. Grishman T. Hasegawa, S. Sekine. Discovering relations among named entities from large corpora. In *Proceedings of the 42nd Annual Conference of the Association for Computational Linguistics, ACL*, pages 15–22, 2004.

[104] Roberto Tazzoli and Paolo Castagna Stefano Emilio Campanini. Towards a semantic wiki wiki web, poster track at the 3rd international semantic web conference (iswc 2004). 2004.

[105] TNS Infratest. (N)ONLINER Atlas 2007 - Eine Topographie des Digitalen Grabens durch Deutschland. Studie, TNS Infratest, 2007. Zu finden unter `http://www.nonliner-atlas.de/`.

[106] UIMA - Unstructured Information Management Architecture, Apache Incubator Project, `http://incubator.apache.org/uima/`.

[107] Unified Medical Language System, United States National Library of Medicine, http://umlsinfo.nlm.nih.gov/.

[108] Max Voelkel, Markus Kroetzsch, Denny Vrandecic, Heiko Haller, and Rudi Studer. Semantic wikipedia. In *Proceedings of the 15th international conference on the World Wide Web, WWW2006*, 2006.

[109] Johanna Völker, Denny Vrandecic, and York Sure. Automatic evaluation of ontologies (aeon). In Y. Gil, E. Motta, V. R. Benjamins, and M. A. Musen, editors, *Proceedings of the 4th International Semantic Web Conference (ISWC2005)*, volume 3729 of *LNCS*, pages 716–731. Springer Verlag Berlin-Heidelberg, NOV 2005.

[110] Juni 2008. W3C Implementation Survey for SPARQL: `http://www.w3.org/2001/sw/DataAccess/tests/implementations`.

[111] Andreas Wagner and Marc Rössler. WALU - Eine Annotations- und Lern-Umgebung für semantisches Tagging. In *GLDV Frühjahrstagung*, 2007.

[112] D. Winer. XML-RPC Specification. Technical report, Userland, Juni 1999. Erhältlich unter `http://www.xmlrpc.com/spec`.

[113] Yoav Freund and Robert E. Schapire. Large Margin Classification Using the Perceptron Algorithm. *Machine Learning*, 37(3):277–296, 1999.

[114] Yusuke Shinyama and Satoshi Sekine. Preemptive information extraction using unrestricted relation discovery. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL*, pages 304–311, 2006.

[115] D. Zelenko, C. Aone, and A. Richardella. Kernel methods for relation extraction. *Journal of Machine Learning Research*, 3(8), 2003.

Lebenslauf

12/2004 - heute	Promotionsstudium der Informatik an der Rheinischen Friedrich-Wilhelms-Universität Bonn
7/2001 - heute	Wissenschaftlicher Mitarbeiter am Fraunhofer IAIS, ehemals Fraunhofer IMK, in Sankt-Augustin
10/1995 - 7/2001	Studium der Informatik mit Nebenfach Photogrammetrie an der Rheinischen Friedrich-Wilhelms-Universität Bonn. Abschluss Diplom mit Note Sehr Gut

Veröffentlichungen

Begutachtete Konferenzbeiträge

1. L. Bröcker, M. Rössler und A. Wagner. Knowledge Capturing Tools for Domain Experts. In *Proceedings of the Semantic Authoring, Annotation and Knowlege Markup Workshop (SAAKM2007) located at the 4th International Conference on Knowledge Capture (KCAP07)*, 2007. CEUR Workshop proceedings 289(3). Erhältlich unter: `http://ceur-ws.org/Vol-289`

2. L. Bröcker. Semiautomatic Creation of Semantic Networks. In *Proceedings of the Knowledge Web PhD Symposium located at ESWC2007*, 2007. CEUR Workshop Proceedings 275(12). Erhältlich unter: `http://ceur-ws.org/Vol-275`

3. L. Bröcker, S. Paal, M. Rössler, A. Wagner, W. Hoeppner, A. Burtscheidt und B. Frings. Wikinger - Wiki Next Generation Enhanced Repositories. In *Online Proceedings of the 1st German E-Science Conference*, 2007. Max-Planck-Gesellschaft, Open-Access-Server.

4. L. Bröcker und S. Paal. Filtering Internet News Feeds using Ad-Hoc Ontologies. In *On the Move to Meaningful Internet Systems 2006: OTM Workshops*, LNCS 4277, Seiten 44-45, Springer, 2006.

5. L. Bröcker und S. Paal. Semantic Filtering of Internet News Feeds. In *Proceedings of IADIS International Conference WWW/Internet 2006*, IADIS Press, 2006.

6. L. Bröcker. Wikis als Mittel zur Ontologieverfeinerung. In *Informatik 2005 - Informatik LIVE - Band 2, Beiträge der 35. Jahrestagung der Gesellschaft für Informatik e.V. (GI)*, Lecture Notes in Informatics, p-68, pp 119-123, Gesellschaft für Informatik e.V., 2005.

7. L. Bröcker. Topic Maps - Semantische Verknüpfungen für Sammlungen. In: *Conference Proceedings of EVA - Electronic Imaging & the Visual Arts*. Stanke, Bienerert et al. (Eds), pp 42-46, 2003.

8. L. Bröcker. Improving the Performance of CBIR Systems through global Application of User Feedback. In: *Proceedings of WebNet 2001 - World Conference on the WWW and Internet*, pp 111-116. AACE - Association for the Advancement of Computing in Education, 2001.

9. L. Bröcker, M. Bogen, A. B. Cremers. Bridging the semantic gap in content-based image retrieval systems. In: *Internet Multimedia Management Systems II - Proceedings of SPIE*, pp. 54-62. SPIE - The International Society for Optical Engineering, 2001.

10. L. Bröcker, M. Bogen, A. B. Cremers. Improving the retrieval performance of content-based image retrieval systems: The GIVBAC approach. In: *Proceedings of the Fifth International Conference on Information Visualisation*, pp. 659-663. IEEE Computer Society, 2001.

11. M. Borowski, L. Bröcker, S. Heisterkamp, J. Löffler. Structuring the Visual Contents of Digital Libraries Using CBIR Systems. In: *Proceedings of the 4th IEEE Conference on Information Visualisation*, pp. 288-293. IEEE Computer Society, 2000

Begutachtete Zeitschriftenartikel

1. L. Bröcker. The WIKINGER Project - Knowledge Capturing for Domain Experts. *ERCIM News, Special Issue 'The Future Web'*, 72:20-21, 2008

2. L. Bröcker. WIKINGER - Semantically Enhanced Knowledge Repositories for Scientific Communities. *ERCIM News, Special Issue 'European Digital Library'*, 66:50-51, 2006