Technische Hochschule
Brandenburg
University of
Applied Sciences

Fachbereich Informatik und Medien

MASTERARBEIT

Konzeption und prototypische Implementierung eines Softwaretools
zur dynamischen Erstellung von Befundberichten
mit Hilfe ontologiebasierter Methoden

Vorgelegt von: Sebastian Busse

am: 02.03.2017

zum
Erlangen des akademischen Grades

MASTER OF SCIENCE

(M.Sc.)

Erstgutachter: Prof. Dr.-Ing. Jochen Heinsohn
Zweitgutachter: Dipl.-Inform. Ingo Boersch

Zusammenfassung

Die vorliegende Arbeit beschreibt einen Ansatz, nach dem pathologische Befundberichte strukturiert und vollständig erstellt werden können. Die implementierte Software nutzt Vorlagen der ICCR (International Collaboration on Cancer Reporting), um ein formales Modell der drei Report-Typen zur Erstellung von Endometrium-, Haut- und Prostatakrebsbefundberichten zu erstellen. Bei den erzielten Dokument-Repräsentationen handelt es sich um Wissensbasen, welche in der Web Ontology Language (OWL) formuliert und somit nicht nur maschinenlesbar, sondern darüber hinaus maschinenverständlich sind. Durch die formal spezifizierte Semantik des entsprechenden Formats lassen sich die Berichte unter Verwendung des HermiT-Reasoners auf Vollständigkeit überprüfen. Des Weiteren wird die Verknüpfung der modellierten Report-Bestandteile zu externen medizinischen Wissensbasen wie SNOMED CT, NCIT und PathLex betrachtet. Die Beschreibung des ontologiebasierten Verfahrens und die prototypische Implementierung des Softwaretools sollen eine mögliche Darstellungsform aufzeigen, nach der Befundberichte im Bereich der anatomischen Pathologie digital, dynamisch sowie durch vorgegebene Strukturelemente präzise und vollständig erstellt und verarbeitet werden können.

Abstract

This work discusses an approach to create pathology reports in a structured and complete manner. The implemented software uses datasets of the ICCR (International Collaboration on Cancer Reporting) as templates to create a formal model for reporting three types of cancer (endometrial cancer, invasive melanoma, and prostate cancer). The obtained document representations embody knowledge bases, which are formulated in the Web Ontology Language (OWL) and are thus not only machine-readable but even machine-understandable. Due to the formal specified semantics of the corresponding format, the reports can be checked for completeness using the HermiT Reasoner. Furthermore, the mapping of the modeled report components to external medical knowledge bases such as SNOMED CT, NCIT and PathLex is considered. The intention of describing the ontology-based procedure and the prototypal implementation of the software tool is to demonstrate a possible representation, according to which anatomic pathology reports can be generated and processed digitally and dynamically by being both precise and complete due to the use of predetermined structural elements.

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Die Strukturierung von Daten erleichtert das Erfassen ihrer inhärenten Informationen. Der erste Überblick über ein komplexes Thema lässt sich für den Menschen anhand einer geeigneten Gliederung schneller gewinnen als aus einem unformatierten Fließtext.

Trotz dieser Tatsache werden Daten im pathologischen Bereich heutzutage immer noch papiergebunden oder vereinzelt in Form von elektronischen Dokumenten ohne vordefinierte Struktur erfasst und verbreitet.

Für den reinen Informationsaustausch zwischen Menschen untereinander scheint dies kein Problem zu sein. So konnte beispielsweise während einer Studie von Sistrom und Honeyman-Buck [SHB05] kein nennenswerter Unterschied zwischen der Auswertung von unstrukturierten und der von strukturierten radiologischen Befundberichten festgestellt werden. Beide Berichtsformate wurden in derselben Zeit mit derselben Effizienz und Erfolgsrate auf fallspezifische Informationen durchsucht. Dennoch gaben die teilnehmenden Probanden an, die strukturierte Version zu bevorzugen.

Das enorme Wachstum der in der heutigen Zeit anfallenden Dokumente im klinischen Alltag macht eine Auswertung aller Daten von Menschenhand unmöglich. Um Zeit und damit Kosten zu sparen, müssen diese Dokumente auf elektronischem Weg vollautomatisiert, oder zumindest durch unterstützende Prozeduren teilautomatisiert ausgewertet werden können. Besonders Einrichtungen, bei denen eine große Menge von Daten zusammenläuft, wie zum Beispiel die klinischen und epidemiologischen Krebsregister sowie die Organzentren benötigen diese Daten in einheitlicher Form [OLA$^+$13, Scope]. Strukturierung bedeutet je nach Konsequenz, mit der diese letztendlich eingehalten werden muss, einen gewissen Grad an Vereinheitlichung. Dadurch ist eine maschinelle Verarbeitung der Daten denkbar, was wiederum Automatismen beim Datenabgleich ermöglicht und somit für eine besser kontrollierbare Qualitätssicherung sorgt.

Eine große Herausforderung ist die Interpretierbarkeit der Informationen eines von Menschen frei formulierten Fließtexts für Maschinen. Aus diesem Grund gibt es zahlreiche Unternehmungen, die darauf abzielen, Kerninformationen aus uneinheitlichen Textdaten herauszufiltern und in Form strukturierter Daten maschinenverständlich bereitzustellen (z. B. [SM03], [SAMK05], [SPNS09]).

Die vorliegende Arbeit zeigt eine Methode, nach der pathologische Befundberichte in vordefinierter Form erstellt werden können. Dies soll zukünftig den Umweg ersparen, einen Befundbericht zunächst in Fließtext zu verfassen und erst anschließend dessen Inhalt mittels diverser Computerlinguistik-Methoden, wie beispielsweise des Text Mining, maschinenverständlich zu machen.

Darüber hinaus erläutert diese Arbeit ein Verfahren, mit dem entsprechend strukturierte Befundberichte auf ihre inhaltliche Vollständigkeit überprüft werden können. Auf diese Weise soll der Pathologe dabei unterstützt werden, die Ergebnisse seiner Beobachtungen in vordefinierter Form und lückenlos mit allen relevanten Daten zu dokumentieren.

Durch geeignete Verbindungen zu etablierten medizinischen Terminologien wird die semantische Bedeutung der Berichtsinhalte verstärkt, was zum einen den elektronischen Datenaustausch zwischen informationstechnischen Systemen erleichtert und zum anderen eine softwaregesteuerte Suche nach definierten Parametern ermöglicht. So können beispielsweise gespeicherte Befunde nach bestimmten Diagnosen durchsucht werden. Diese Möglichkeit ist vor allem für klinische Studien im Bereich der anatomischen Pathologie interessant.

1.2 Aufgabenstellung

Ziel der Arbeit ist ein implementiertes Softwaretool, mit dessen Hilfe pathologische Befundberichte drei verschiedener Krebserkrankungen beispielhaft dynamisch erzeugt und auf inhaltliche Vollständigkeit überprüft werden können. Bei den drei Erkrankungen handelt es sich um Endometrium-, Haut- und Prostatakrebs. Die Software soll sowohl zum Erstellen des Berichts als auch für den Vollständigkeits-Check eine eigens dafür konzipierte Ontologie nutzen.

Des Weiteren sollen für einige innerhalb dieser erstellten Ontologie repräsentierten Inhalte eines Befundberichts Mappings zu Inhalten anderer medizinischer Terminologien wie beispielsweise SNOMED CT in der Arbeit demonstriert werden.

1.3 Vorgehensweise

Zu Beginn wird erläutert, warum sich dazu entschlossen wurde, ein solches Softwaretool zu entwickeln und welche Anforderungen an das Programm gestellt werden. Damit einhergehend erfolgt eine Analyse des Ist-Zustands, um sich mit bisherigen Definitionen und Lösungsansätzen im Umfeld der geplanten Software vertraut zu machen. Hierbei wird auch untersucht, ob und welche unternommenen Bestrebungen das aktuelle Vorhaben unterstützen und sich auch für dieses verwenden lassen.

Im Anschluss werden die bei der Entwicklung der Software verwendeten ontologiebasierten Methoden „Reasoning", „Ontology Learning" und „Mapping" vorgestellt.

Aus den gewonnen Kenntnissen folgen Konzeption und Implementierung der als Ziel dieser Arbeit formulierten Softwarekomponente. Es wird beschrieben, welche externen Daten dem Programm als Input dienen, wie die Software aufgebaut ist, mit Hilfe welcher Entwicklungsumgebungen und Bibliotheken sie entwickelt wurde und wie sie funktioniert. Dabei wird sowohl erklärt, wie die Befundberichte dynamisch erstellt werden, als auch, wie das Ergebnis des Vollständigkeits-Checks mittels Inferenz geschlussfolgert wird.

Um die semantische Aussagekraft der innerhalb der Ontologie definierten Klassen zu erhöhen, wird gezeigt, wie Inhalte verschiedener Wissensbasen mit Hilfe von Mapping miteinander verknüpft werden können. Dazu werden beispielhaft einige durch Klassen repräsentierte Inhalte eines Befundberichts mit Inhalten anderer Terminologien verlinkt.

1.4 Abgrenzung

Bei der im Zuge dieser Arbeit entwickelten Software handelt es sich um einen Prototypen, das bedeutet, dass gewonnene Kenntnisse und erläuterte Methoden beispielhaft innerhalb des Programms umgesetzt sind. Das Layout der dynamisch erstellten Befundberichte spielt dabei keine besondere Rolle. Die zur Programmlaufzeit getätigten klinischen Angaben orientieren sich an möglichen Fallbeispielen, sind aber an keine reale Person gebunden und insofern frei erfunden. Es wird nicht die Vollständigkeit von existierenden Dokumenten getestet, sondern erläutert, wie sich entsprechend aufgebaute Beispieldokumente mit Hilfe der implementierten Software und der konstruierten Ontologie auf einzelne Inhalte überprüfen lassen. Das Wissen darüber, welche Angaben in einem Befundbericht empfohlen oder gar zwingend erforderlich sind, wird aus vorhandenen anerkannten Expertenmeinungen extrahiert und nicht im Zuge dieser Arbeit neu erschlossen.

In medizinischer Hinsicht handelt es sich bei betrachteten Inhalten ausschließlich um den Diskursbereich der drei gewählten Befundberichte. Dieser umfasst die Befundung einer Probe mit Verdacht auf Endometrium-, Haut- oder Prostatakrebs. Bei letzterem Bericht sind ausschließlich Proben aus einer radikalen Prostatektomie (Prostata-Entfernung) berücksichtigt.

Bei der Frage nach dem Vorhandensein erforderlicher Inhalte gibt es ausschließlich die Antwortmöglichkeiten *vorhanden* oder *nicht vorhanden*. Es erfolgt keine Plausibilitätsprüfung der angegebenen Daten.

2 Vorbetrachtung

Dieses Kapitel umfasst im ersten Abschnitt die Aufzählung der Gründe für die Entwicklung der geplanten Software. Anschließend wird beschrieben, welche Grundvoraussetzungen für das aktuelle Vorhaben bereits gegeben sind und ob sich dafür existierende Lösungsansätze adaptieren lassen. Aus den Gründen leitet sich die Definition der Programmanforderungen ab, auf deren Basis später dessen Konzeption erfolgt.

2.1 Gründe für das Softwaretool

Ein besonders wichtiger Grundgedanke, der bei der Entstehung dieser Arbeit verfolgt wurde, ist der einer Vereinheitlichung der Struktur aller Befundberichte im pathologischen Bereich. Unter der Prämisse, dass sich eine geeignete Form solcher Dokumente global standardisieren lässt, können zukünftig Berichte mit immer gleichem Aufbau verfasst werden. Das hat den Vorteil, dass diese einheitlich, computergesteuert verarbeitet werden können. Wie einleitend im Abschnitt 1.1 erwähnt, ist eine manuelle Bearbeitung aller Daten und Dokumente im heutigen klinischen Umfeld längst nicht mehr möglich. Im Zeitalter der digitalen Patientenakte werden administrative Prozesse in großem Umfang automatisiert getriggert und abgearbeitet. Das Erstellen einiger medizinischer Dokumente wie beispielsweise eines Arztbriefs mit enthaltener Diagnose erfolgt bereits seit einigen Jahren digital, jedoch zum Teil nach wie vor in Schriftstücken uneinheitlicher Struktur, abhängig vom Land, medizinischer Einrichtung und verwendetem System, in dem es erstellt wird. Die vorliegende Arbeit soll für den pathologischen Bereich einen Ansatz liefern, nach dem ein Befundbericht standardisiert erzeugt werden kann. Für einheitlich formatierte Berichte lassen sich unterstützende Prozesse zu deren Generierung entwickeln. Die damit einhergehenden Vorteile sind sowohl Zeit- und Kosteneinsparung als auch die Steigerung der maschinell abbildbaren Arbeitsschritte und der Übersichtlichkeit innerhalb der Dokumentenlandschaft. Gerade in Bezug auf große Datenmengen und Datensammelstellen wie Krebsregister und Organzentren ist zu erwarten, dass dies die Dokumentationsarbeit des Pathologen ebenso erleichtert wie den administrativen Aufwand der Datenintegration in verschiedenste Krankenhausinformations- und Subsysteme.

Wenn außerdem zu jeder Berichtsklasse Konsens über eine formale Festlegung der im Einzelbericht erforderlichen Inhalte herrscht, ist ein Automatismus denkbar, der ein Dokument auch inhaltlich auf Vollständigkeit überprüft. Für jedes pathologische Krankheitsbild kann festgelegt werden, wie die Beschreibung der Untersuchungsergebnisse zu strukturieren und zu codieren ist. Ein solcher Standard kann den Pathologen dabei unterstützen, die Ergebnisse seiner Beobachtungen schematisch und lückenlos zu dokumentieren.

Durch Verknüpfung mit formal definierten Wissensbasen lassen sich die Berichtsinhalte für Maschinen semantisch verständlich repräsentieren. Dies erleichtert den elektronischen Austausch und die Übertragung der Informationen, welche im medizinischen Bereich unter Verwendung definierter Standards (HL7[1], DICOM[2]) realisiert und unterstützt ist. Beispiele für solche Wissensbasen sind Terminologien wie ICD-10[3], LOINC[4] und SNOMED CT[5].

Auch in Bezug auf Suchanfragen ist die Erstellung maschinenverständlicher Befundberichte ein erstrebenswertes Ziel. So können die Dokumente dann im Nachhinein nicht nur stringbasiert nach speziellen Textpassagen durchsucht werden, sondern auch nach den definierten Codes aus den einzelnen Terminologien. Auf diese Weise wird nicht länger nur nach den konkreten Worten aus der Suchanfrage gesucht, sondern nach allen Worten, die das gesuchte Konzept beschreiben. Das bedeutet, Synonyme, Abkürzungen und unterschiedliche Schreibweisen beziehungsweise Sprachen des gesuchten Wortes sind über eine solche Suche ebenfalls identifizierbar. Somit können dann beispielsweise gespeicherte Befunde nach bestimmten Untersuchungskriterien oder Diagnosen durchsucht werden, was unter anderem für klinische Studien interessant ist.

Im Endeffekt profitiert nicht zuletzt auch der betroffene Patient davon, dass seine Daten durch gesteigerte Produktivität und Qualitätssicherung schneller und transparenter verarbeitet werden können.

[1] Health Level Seven: http://www.hl7.org
[2] Digital Imaging and Communications in Medicine: http://medical.nema.org
[3] International Statistical Classification of Diseases and Related Health Problems 10th Revision: http://apps.who.int/classifications/icd10/browse/2015/en
[4] Logical Observation Identifiers Names and Codes: https://loinc.org
[5] Systematized Nomenclature Of Medicine Clinical Terms: http://www.ihtsdo.org/snomed-ct

2.2 Analyse des Ist-Zustands

Ein Befundbericht aus der Domäne der anatomischen Pathologie dokumentiert die Ergebnisse einer pathologischen Untersuchung von Patienten zuvor entnommenem Gewebe. Das Dokument enthält medizinische Daten, die sowohl wichtig für die Prognose sind, als auch wertvoll für Forschung, Epidemiologie und zu (Fort-)Bildungszwecken. Obwohl pathologische Befundberichte typischerweise einen verglichen mit Laborbefunden kleinen Anteil der Gesamtdatenmenge in einer Patientenakte ausmachen, haben sie großen Einfluss auf die Patientenversorgung. [DM11, S. 15]

Die Berichte sind häufig in Form von frei formuliertem Fließtext verfasst, da sich dieser gut mit einem Diktiergerät aufnehmen und später ins Reine schreiben lässt. Unstrukturierter Text erschwert jedoch erheblich das Extrahieren der für die Versorgung wichtigen Kerninformationen sowie das automatisierte Aufteilen der Daten zwischen verschiedenen klinischen und verwaltungstechnischen Anwendungen. [DM11, S. 8]

2.2.1 Strukturierung pathologischer Befundberichte: APSR

Das aus einer Zusammenarbeit von IHE Anatomic Pathology[6] und HL7 Anatomic Pathology[7] entstandene Dokument „IHE Anatomic Pathology Technical Framework Supplement - Anatomic Pathology Structured Reports (APSR)" beschreibt ein Inhaltsprofil für strukturierte Befundberichte im Bereich der anatomischen Pathologie (nachfolgend mit APSR abgekürzt). Ziel dieser Ausarbeitung ist, eine Art grundlegende Standard-Struktur für solche Reporte vorzuschlagen, um eine international vereinheitlichte, exakte Datenerhebung und deren effiziente Nutzung zu ermöglichen. Darüber hinaus wurde im Zuge dieser Initiative auch ein Implementation Guide veröffentlicht, welcher die Umsetzung von APSRs unter Verwendung der Clinical Document Architecture (CDA) nach Dolin et al. [DAB+06] beschreibt. [DM11, S. 7]

Das Inhaltsprofil definiert folgende Bereiche (Sections) innerhalb eines APSR: [DM11, S. 7-8]

- **Klinische Informationen**
  Dieser Inhalt wird in der Regel von der Person geliefert, welche die durch den APSR beschriebene pathologische Untersuchung angefordert hat. (zum Beispiel Anamnese, Grund für die pathologische Untersuchung, Probeentnahmeverfahren und Region der entnommenen Probe)

[6] IHE Anatomic Pathology: http://www.ihe.net/Anatomic_Pathology/
[7] HL7 Anatomic Pathology: http://www.hl7.org/Special/committees/anatomicpath/

- **Intraoperative Beobachtungen** (sofern welche veranlasst wurden)
- **Makroskopische Beobachtungen**
- **Mikroskopische Beobachtungen**
- **Diagnose**
- **Schrittfolge der durchgeführten Prozeduren**
  Hier kann nachvollzogen werden, welche pathologischen Verfahren an der/den entsprechenden Probe/n vorgenommen wurden. Das schließt nicht aus, dass einige, der hier beschriebenen Schritte, auch in anderen Bereichen - wie zum Beispiel in dem der makroskopischen Beobachtungen - enthalten sind.

Bezüglich des Inhalts der einzelnen Bereiche des APSR ist Folgendes festgelegt: [DM11, S. 8]

- Jeder Bereich muss einen Text enthalten, der den Inhalt für einen Menschen lesbar darstellt. Wie genau dieser Text aufgebaut ist, wird nicht spezifiziert. Demzufolge ist hier Fließtext ebenso zulässig wie eventuell durch regionale oder fachspezifische Richtlinien vorgeschriebenes Vokabular.
- Der Bereich der Diagnose muss mindestens ein sogenanntes Entry-Element enthalten, welches die Diagnose inhaltlich in Form von maschinenverständlichen Daten repräsentiert.
- Die anderen Bereiche sollten - müssen aber nicht - solche Entry-Elemente enthalten.
- Der Bereich der klinischen Informationen kann weitere Sections beinhalten.

Gemäß den vorangegangenen Spezifikationen, enthält ein APSR sowohl menschen- als auch maschinenverständliche Informationen. Aus diesem Grund müssen Anwendungen, um einen APSR korrekt erzeugen zu können, mit beiden dieser Informationsarten umgehen können und dabei gleichzeitig eine Benutzerschnittstelle anbieten, welche Verwechslungen ausschließt. Für den Anwender muss zu jeder Zeit die Trennung zwischen den zwei Beschreibungswelten klar ersichtlich sein und somit auch, welche Information für welchen Interpreter gedacht ist. Dies könnte beispielsweise durch das Bereitstellen von Formularen mit entsprechenden Textfeldern für die Eingabe von frei formulierbarem Text und speziell gekennzeichneten Kontrollkästchen (Checkboxes), Listen-, Kombinations- oder Optionsfeldern für die Auswahl codierter Entry-Elemente realisiert werden. [DM11, S. 8]

Eine Software, die das Erzeugen eines APSR und dessen Übertragung ermöglicht, wird im Inhaltsprofil als **Content Creator** bezeichnet [DM11, S. 14]. Benutzer einer solchen Software ist beispielsweise ein Pathologe, der Vorgehen und Ergebnis(se) seiner pathologischen Untersuchung(en) in einem APSR beschreibt.

Ein **Content Consumer** empfängt die vom Content Creator erstellten APSRs und ist dafür verantwortlich, dass diese importiert und für den Anwender korrekt angezeigt werden können [DM11, S. 15]. Diese Software ist optimalerweise für eine Bedienung durch einen Pathologe oder einen medizinischen Assistenten geeignet, um beispielsweise Berichte aus der Vergangenheit erneut betrachten und durchsuchen zu können. Auch ein Arzt einer anderen medizinischen Einrichtung, der einen APSR von einem Pathologen als Ergebnis einer zuvor angeforderten Untersuchung erhält, kann demnach einen Content Consumer nutzen, um Einsicht in den Bericht zu erhalten.

Bei den spezifizierten APSRs handelt es sich um Dokumente, die nach dem Standard CDA R2 (Clinical Data Architecture Release 2.0) formatiert sind. Ein solches Dokument besteht grundlegend aus einem Dokumentenkopf (Header) und einem Dokumentenkörper (Body).

Im **Header** stehen Informationen über den Kontext des pathologischen Behandlungsauftrages wie beispielsweise die Patientenstammdaten, die medizinische Einrichtung, die die Untersuchung in Auftragt gegeben hat, und der beurteilende Pathologe. [DM11, S. 8]

Der **Body** enthält die einzelnen hierarchisch gegliederten Bereiche (Sections). Nach oben definiertem Vorbild beschreibt jeder Bereich seinen Inhalt in Form von menschenlesbarem Text, welcher gegebenenfalls durch das Einbinden von kleinen Grafiken illustriert ist. Um den Speicherbedarf eines APSR möglichst gering zu halten, ist es möglich, Bildmaterial wie Whole-Slide-Images mit dem Text zu verlinken. Die Bilddaten verbleiben dadurch in ihrer separaten Speicherstruktur und sind mit Hilfe der Elemente *observationMedia* oder *regionOfInterest* über den DICOM-Standard erreichbar. [DM11, S. 9]

Darüber hinaus enthalten einige Bereiche Entry-Elemente, welche die Informationen aus dem Text-Element in maschinell semantisch interpretierbare Daten umwandeln. Entry-Elemente stehen innerhalb einer Section nach dem Text-Element. Jedes Entry-Element referenziert über eine URI-Adresse ein bestimmtes Konzept, welches innerhalb einer Terminologie semantisch beschrieben ist. Dabei besitzt jedes Konzept einen eindeutigen URI (Uniform Resource Identifier). Auf die Terminologie, in welcher das gewünschte Konzept zu finden ist, wird ebenfalls durch einen URI verwiesen. Dieser bildet den Adressraum - auch Namespace genannt - der Terminologie. Die Adresse jedes Konzepts beginnt mit dem URI der Terminologie, in der es modelliert ist. Die Entry-Elemente des nach CDA R2-Standard formatierten APSRs können auf verschiedene Terminologien verweisen. Im APSR-Anwendungsprofil werden hier beispielhaft SNOMED CT, ICD-O-3, LOINC und ADICAP benannt [DM11, S. 8]. Auch die von der IHE Anatomic Pathology mit entwickelte Terminologie PathLex wird in drei Code-Beispielen mit eingebunden [DM11, S. 47 und S. 79].

Zu beachten ist, dass das APSR-Dokument keine automatisierte Erkennung von Konzepten innerhalb eines Fließtextes beschreibt. Die Konzepte sollen viel mehr vom Kliniker selbst bei der Erstellung des Berichts unter Verwendung von Hilfsmitteln wie beispielsweise Listen-, Kombinations- und Optionsfeldern ausgewählt werden. Dazu ist ein zusätzliches Softwaretool notwendig, das dann als Content Creator agiert.

Das APSR-Profil beinhaltet 22 HL7 CDA R2 Templates für pathologische Befundberichte (einen allgemeinen APSR, einen allgemeinen APSR für Krebs und 20 organspezifische APSRs für die strukturierte Dokumentation der jeweiligen Krebsart) [DM11, S. 19, 32 und 70].

Thematisch befasst sich das APSR-Inhaltsprofil derzeit mit dem operativen Gebiet der anatomischen Pathologie (Krebserkrankungen, benigne (gutartige) Neoplasmen und nicht entartetes Material) sowie der Zytopathologie. Auch Untersuchungsmethoden der „traditionellen“ Pathologie wie zum Beispiel unter Verwendung von Lichtmikroskopie (einschließlich Immunhistochemie) werden berücksichtigt. Spezielle Techniken wie die Durchflusszytometrie, die Zytogenetik und die Elektronenmikroskopie sowie der Bereich der forensischen Pathologie (Autopsie, Toxikologie) seien für die Zukunft geplant. [DM11, S. 15]

2.2.2 Inhaltliche Vorlagen für pathologische Befundberichte

Während es in heutigen pathologischen Befundberichten bereits anerkannter Standard und gelebte Praxis ist, gewisse Angaben zur Ergebnisbeschreibung einer Untersuchung wie zum Beispiel Tumorgröße, Randstatus, Grading und Stadium festzuhalten, gibt es doch eine Vielzahl an unterschiedlichen Vorlagen zu deren Dokumentation und nicht alle verlangen nach denselben Informationen in derselben Struktur. Im Allgemeinen sollen solche Vorlagen den Pathologen dabei unterstützen, Fehler bei der Berichterstattung zu vermeiden. Dazu gehören sowohl Falschangaben als auch fehlende Informationen.

Mehrere internationale Initiativen haben bereits damit begonnen, strukturierte Vorlagen für bestimmte Arten von pathologischen Befundberichten zu definieren. Das CAP (College of American Pathologists) hat in der Krebs-Domäne 80 Protokollvorlagen veröffentlicht, um die Erstellung von vollständigen Reports zu erleichtern[8]. In Frankreich haben die SFP (Société Française de Pathologie)[9] und das INCa (Institut National du Cancer)[10] 23 APSR-Vorlagen veröffentlicht, die festlegen, welche Daten zur Beschreibung eines Primärtumors erforderlich sind. In Australien entwickeln das RCPA (Royal College of Pathologists Australasia) und

[8] Krebs-Protokoll-Templates des CAP und Hintergrundinformationen: http://www.cap.org/web/oracle/webcenter/portalapp/pagehierarchy/cancer_protocol_templates.jspx
[9] Französische Gesellschaft der Pathologie: www.sfpathol.org
[10] Nationales Krebsinstitut Frankreichs: www.e-cancer.fr

die nationale Behörde Cancer Australia in Zusammenarbeit mit nationalen Medizinern und Pathologie-Organisationen unter anderem Protokollvorlagen[11] für Befundberichte von Lungen-, Haut-, Brust-, Darm-, Lymphdrüsen- und Prostatakrebs sowie ein Rahmenwerk, um die Entwicklung dieser Protokolle zu lenken. [DM11, S. 15]

Eine internationale Zusammenarbeit von Experten aus den USA, Großbritannien, Kanada und Austalien hat unter dem Namen ICCR (International Collaboration on Cancer Reporting) acht Datensätze[12] veröffentlicht, die festgelegen, welche Angaben innerhalb eines Befundberichts der jeweiligen Krebsart erforderlich und welche darüber hinaus als empfohlen zu betrachten sind. [EDH+13]

2.2.3 PathLex als pathologische Terminologie

Das Projekt PathLex wurde gemeinsam von den Arbeitsgruppen IHE Anatomic Pathology und HL7 Anatomic Pathology in Zusammenarbeit mit anderen im Bereich der anatomischen Pathologie tätigen Organisationen - einschließlich CAP[13], SFP[14] und SEAP[15] - ins Leben gerufen [Dan10]. PathLex bildet eine Terminologie ab, die inhaltlich die Beobachtungen und Probeentnahmeverfahren bei Untersuchungen im Bereich der anatomischen Pathologie umfasst [DM11, S. 30]. Sie wurde mit dem Ziel entwickelt, das Erstellen und Bearbeiten von Befundberichten sowohl für Anbieter als auch Nutzer von pathologischen Informationssystemen zu vereinfachen. Über die Definition eines gemeinsamen Grundvokabulars hinaus soll mit Hilfe von PathLex im Bereich der anatomischen Pathologie eine semantische Konsistenz der Standard-Nachrichten und Dokumentstrukturen innerhalb und zwischen den einzelnen Standards (HL7 v2.5, HL7 v3, DICOM) erreicht werden. Der Idee des APSR-Profils bezüglich einer einheitlichen Strukturierung von Befundberichten folgend, soll PathLex auch dabei helfen, klinische Angaben zu codieren und damit deren Bedeutung sowohl für Mensch als auch für Maschine verständlich zu repräsentieren. Dazu werden einerseits bestehende Terminologien wie SNOMED CT und ICD-O genutzt und andererseits fehlendes Wissen durch neu modellierte Konzepte ergänzt. [Dan10]

PathLex wird innerhalb der APSR-Vorlage durch die Ontologie-ID 1.3.6.1.4.1.19376.1.8.2.1 referenziert und steht in der aktuellen Version 1.2 auf der BioPortal-Website kostenlos zum Download zur Verfügung[16]. Auf der Website der internationalen Organisation HL7 ist das

[11] Protokolle von RCPA und Cancer Australia: https://www.rcpa.edu.au/Library/Practising-Pathology/Structured-Pathology-Reporting-of-Cancer/Cancer-Protocols
[12] ICCR-Datensätze: http://www.iccr-cancer.org/datasets
[13] College of American Pathologists: www.cap.org
[14] Société Française de Pathologie (Französische Gesellschaft der Pathologie): www.sfpathol.org
[15] Sociedad Española de Anatomía Patológica (Spanische Gesellschaft der Pathologie): www.seap.es
[16] Kostenloser Download von PathLex unter http://bioportal.bioontology.org/ontologies/PATHLEX

Projekt offiziell als externes Kodierungssystem gelistet[17]. Genau genommen handelt es sich bei PathLex um eine sogenannte Interfaceterminologie (auch Displayterminologie genannt). Im klinischen Umfeld unterstützt ein solches Vokabular einen Mediziner bei der Eingabe von Informationen in ein Computerprogramm, indem es eine systematische Sammlung von klinisch orientierten Phrasen bereitstellt. Auf dem umgekehrten Weg erleichtern Interfaceterminologien die Darstellung von elektronisch gespeicherten, maschinenverständlichen Patienteninformationen als für den Kliniker einfach lesbaren Text. Somit stellen Interfaceterminologien eine Schnittstelle zwischen den uneingeschränkten, umgangssprachlichen Konzeptualisierungen eines Klinikers und den stärker strukturierten, codierten internen Datenelementen, welche durch spezifische Software verwendet werden, dar. [RMJ+06, S. 277] Mit Hilfe von umgangssprachlichen, im klinischen Alltag geläufigen Begriffen und Synonymen wird dem Anwender die Interaktion mit Konzepten erleichtert. Das ist vor allem bei klinischen Dokumentationssystemen von Vorteil. Ein weiteres Beispiel für eine Interfaceterminologie ist LOINC.

Vergleichend zu den eher am Sprachgebrauch der Benutzer orientierten Interfaceterminologien gehen Referenzterminologien einen Schritt weiter in die Richtung der formalen, expliziten Wissensrepräsentation. Solche Terminologien werden in der Regel entwickelt, um eine genaue und vollständige Darstellung des Wissens einer bestimmten Domäne, einschließlich ihrer Klassen und deren Beziehungen untereinander, zu schaffen. [RMJ+06, S. 277] Somit können Referenzterminologien beispielsweise dabei unterstützen, klinischen Daten zu klassifizieren, strukturiert zu speichern und für Abfragen wiederauffindbar zu machen. Das Speicherformat der Daten wird dabei von deren interner Systemdarstellung bestimmt, auf welche Interfaceterminologien dann mit Hilfe von Mapping verweisen können. [RMJ+06, S. 278]

Demnach stellt die Konzeptualisierung von PathLex eine Palette an flexiblen, "Pathologenfreundlichen" Phrasen bereit, erhebt jedoch keinen Anspruch auf eine vollständige, allumfassende semantische Repräsentation der enthaltenen Konzepte in Bezug auf den realen Wissensbereich der Medizin. Als Interfaceterminologie besteht die Strategie von PathLex zur semantischen Interoperabilität darin, aus den von Pathologen verwendeten Phrasen Konzepte abzuleiten und diese dann auf standardisierte Referenzterminologien zu verlinken. Damit ein solches Mapping erfolgreich durchgeführt werden kann, müssen neu definierte Konzepte, die bisher in keiner Referenzterminologie vorkommen mit Hilfe bekannter Konzepte und Relationen beschrieben werden. Die bekannten Konzepte können anschließend mit ihrer Repräsentation in bestehenden Referenzterminologien verknüpft werden. Auf diese Weise kann PathLex die Wissensbasis im Anwendungsbereich der anatomischen Pathologie umfassend

[17] PathLex gelistet bei HL7: http://www.hl7.org/oid/index.cfm?Comp_OID=1.3.6.1.4.1.19376.1.8.2.1

repräsentieren und somit als Hilfe zur semantischen Strukturierung bei der Erstellung von pathologischen Dokumentationen und Befundberichten dienen.

Die umfassendste Referenzterminologie im medizinischen Bereich ist SNOMED CT. [SNOb] Zur Erweiterung der semantischen Aussagekraft werden Konzepte von PathLex nach Möglichkeit auf Konzeptbeschreibungen innerhalb von SNOMED CT verlinkt. Für einige pathologische Beobachtungen ist dieses Mapping bereits weit vorrangeschritten (zum Beispiel im Bereich der histologischen Beobachtungen), für andere stehen in SNOMED CT keine vordefinierten Konzepte mit der entsprechenden Bedeutung zur Verfügung (so wie beispielsweise für die TNM-Klassifikation von Tumoren). [Dan10] Insgesamt verweisen bisher laut BioPortal 340 der 1786 PathLex-Konzepte auf Konzepte von SNOMED CT[18]. Weitere Terminologien, die PathLex referenziert, sind zum Beispiel SNOMED International V3.5 (96 Verweise), RadLex (153 Verweise), OpenGALEN (55 Verweise), ICD-O und ADICAP.

Aus ontologischer Sicht besteht PathLex aus 1786 Klassen, welche in einer vierstufigen Hierarchie angeordnet sind. Es sind weder Instanzen noch Relationen (Properties) modelliert. Einzige Ausnahme ist die Unterklassenbeziehung (*is-a*-Relation), welche eine Klasse als Teil einer übergeordneten Klasse spezifiziert. Bis auf die erste Hierarchie-Ebene enthält jede Klasse eine Beschreibung und obwohl diese theoretisch frei formuliert werden können, sind alle nach demselben Muster aufgebaut: Sie beginnen mit der anatomischen Region, welche bei der Untersuchung eine Rolle spielt (entweder allgemein [„Generic“] oder dem Organ zugeordnet [„Breast“, „Liver“, „Lung“]) und konkretisieren, jeweils durch einen Bindestrich getrennt, den inhaltlichen Schwerpunkt der Klasse. So hat beispielsweise das Konzept namens „Specimen weight“ mit der Pathlex-ID 462 die Beschreibung „Thyroid-Infiltrating malignant neoplasm-Specimen weight“, da das Konzept das Gewicht einer Probe umfasst, welche aus einem in die Schilddrüse eindringenden Malignom entnommen wurde. Diese Beschreibung gibt jedoch keine Auskunft über den tatsächlichen Aufbau der hierarchischen Struktur von PathLex. Da etwa 85 % (n = 1522) der vorhandenen Klassen nicht hierarchisch eingeordnet sind, ist PathLex nicht zur Klassifizierung geeignet. Das deckt sich mit der Beschreibung und dem Ziel von Interfaceterminologien nach Rosenbloom et al. [RMJ+06, S. 277]. PathLex liefert für alle in Form von Klassen modellierten Konzepte aus dem Bereich der pathologischen Befundberichterstellung eine eindeutige Identifikationsnummer mittels URI (Uniform Resource Identifier). Diese URI-Adressen können verwendet werden, um Inhalte von pathologischen Berichten dokumentenübergreifend eindeutig zu referenzieren.

[18] Mapping-Tabelle auf der BioPortal-Website:
http://bioportal.bioontology.org/ontologies/PATHLEX?p=mappings

2.3 Anforderungen

Auf Basis der in Abschnitt 2.1 beschriebenen Entwicklungsgründe soll das zu erstellende Softwaretool folgende Möglichkeiten bieten:

- Einheitliche Strukturierung der erstellten Befundberichte und einheitliches Speicherformat
- Maschinenverständliche Repräsentation der Berichtsinhalte
  - Globale dokumentenunabhängige Codierung durch eindeutige Identifikationsnummer, z. B. durch URI-Adressen
  - Automatisierte Überprüfung der inhaltlichen Vollständigkeit von Befundberichten gemäß Vorgaben internationaler Experten
  - Steigerung der internationalen interdisziplinären Austauschbarkeit durch Mapping der Berichtsinhalte zu Komponenten aus anerkannten medizinischen Wissensbasen, z. B. SNOMED CT

Der erste Punkt der Anforderungsliste zielt darauf ab, eine Strukturierung und ein Speicherformat für pathologische Befundberichte zu finden, welche zukünftig als Standard in diesem Bereich dienen können. Hierzu wird sich an bereits vorhandenen Spezifikationen und technischen Umsetzungen orientiert, um Schwierigkeiten hinsichtlich der breiten Akzeptanz eines möglichen Standards zu minimieren.

Die Modellierung von maschinenverständlichen Axiomen bedingt eine eindeutige Identifikation für jede Art von Konzept und für jedes einzelne Objekt (Individuum). Die Klassen von Fragen und Antworten innerhalb eines spezifischen Report-Typs müssen ebenso identifizierbar sein wie die gestellten Fragen und gegebenen Antworten eines konkreten Einzeldokuments. Auch die wechselseitigen Beziehungen müssen als Mengen von Objektverknüpfungen referenziert werden können. Dementsprechend benötigt jede Relation eine eindeutige Adresse, um sie auf immer gleiche Weise ansprechen zu können. Die genannten Voraussetzungen sollen durch die Modellierung des Diskursbereichs als Ontologie und unter Verwendung ontologiebasierter Methoden erfüllt werden. Eine maschinell gesteuerte Vollständigkeits-Überprüfung auf Basis von logischen Schlussfolgerungen ist dadurch ebenfalls denkbar. Zusätzlich lässt sich durch die Verknüpfung der modellierten Klassen von Berichtsinhalten zu anerkannten internationalen Terminologien deren Bedeutung vereinheitlichen. Dadurch soll die Zusammenarbeit von Pathologen auf globaler Ebene erleichtert werden. Eindeutig zu referenzierende Berichtsinhalte sollen Sprachbarrieren abbauen und ein international einheitliches Verständnis von der jeweils referenzierten Entität erzeugen und fördern.

Grundsätzlich wird die geplante Software mit dem Ziel entwickelt, den Pathologen dabei zu unterstützen, seine Befunde korrekt und vollständig zu erstellen. Zu diesem Zweck sollen die Möglichkeiten zur Beantwortung einer bestimmten Fragestellung durch das Bereitstellen vordefinierter Antwortvarianten eingegrenzt werden. Ein späterer dokumentenübergreifender Vergleich der gegebenen Antworten ist dadurch ebenfalls begünstigt. Die Beschreibung des geplanten Workflows zur Programmlaufzeit wird in Abschnitt 4.1 gegeben.

3 Grundlagen: Ontologiebasierte Methoden

Ausgehend von der Definition von Thomas Gruber als „explizite Spezifikation einer Konzeptualisierung" [Gru93, S. 1] und unter Berücksichtigung der von Borst beschriebenen Erweiterung durch die Begriffe „formal" und „gemeinsam" [Bor97, S. 12], lässt sich eine Ontologie zusammengefasst als „formale, explizite Spezifikation einer gemeinsamen Konzeptualisierung" definieren.

Der Begriff ist angelehnt an die gleichnamige Disziplin der theoretischen Philosophie, die sich mit der Lehre des Seins beschäftigt [Sta16]. In der Informatik geht es dabei letztendlich darum, ein Modell zu beschreiben und zu repräsentieren. Getreu dem Motto „Alles, was existiert, kann auch repräsentiert werden" ist eine *Konzeptualisierung* als ein Modell eines bestimmten Diskursbereichs zu verstehen, welches all seine relevanten Konzepte und Beziehungen enthält.

Explizit und *formal* bedeuten in diesem Zusammenhang, dass alle verwendeten Konzepte definiert und in maschinenverständlicher Form codiert sein müssen. Ohne formale Semantik, kann eine Maschine den textuell modellierten Diskursbereich zwar lesen, jedoch nicht verstehen.

Für eine erfolgreiche Kommunikation ist es zudem wichtig, dass die Kommunikationspartner dieselbe Konzeptualisierung und auch dasselbe Verständnis der ihr zugrundeliegenden Festlegungen haben. Die Konzeptualisierung muss allen Kommunikationspartnern *gemeinsam* sein, das bedeutet, von ihnen allen geteilt werden.

Eine Ontologie ist als eine Art Wissensbasis zu verstehen. Die Wissensbasis ist dabei grundlegend in zwei Komponenten aufgeteilt: Die T-Box, in der die allgemeinen Zusammenhänge eines Anwendungsbereichs definiert sind (terminologisches Wissen), und die A-Box, in der das Wissen über einzelne Objekte mit Hilfe des in der T-Box gegebenen Grundvokabulars beschrieben ist (assertorisches Wissen). [FKM$^+$06, S. 1]

Für die Repräsentation des Wissens einer bestimmten Domäne ist die Beschreibung ihrer Konzepte, ihrer Instanzen und deren wechselseitiger Beziehungen untereinander notwendig. Konzepte werden in einer Ontologie innerhalb der T-Box spezifiziert. Sie können mit Hilfe der Beschreibungslogik formal beschrieben werden. Sogenannte primitive - oder auch

atomare - Konzepte werden dabei durch Klassen abgebildet, welche eine Abstraktion des real existierenden Konzepts darstellen. Definiert werden diese Konzepte über Klassifikation (Mann $\sqsubseteq$ Person) oder Äquivalenz (Mann $\equiv$ männlicherErwachsener) [FKM$^+$06, S. 1]. Die Interpretation von Klassen einer Ontologie sind Objektmengen in der realen Welt.

Beziehungen werden in Ontologien über Relationen realisiert. Innerhalb der T-Box geht es ausschließlich um Beziehungen der Konzepte untereinander. Diese können in der Realität als binäre Objektverknüpfungen, also als Menge von Paaren betrachtet werden. Das bedeutet, dass eine Relation zwischen zwei Klassen zur Folge hat, dass diese Relation auch zwischen allen Instanzen dieser beiden Klassen besteht.

Instanzen sowie deren Relationen untereinander und zu bestimmten Konzepten werden in der A-Box einer Ontologie beschrieben. Sie enthält das fallspezifische Wissen. Eine Instanz ist als Individuum oder auch als Objekt der realen Welt zu verstehen. Ein Objekt kann ein Konzept der T-Box instanziieren (rdf:type) und Beziehungen (rdf:Property) zu anderen Objekten der A-Box besitzen. Die Gesamtheit aller Relationen einer Ontologie wird auch als R-Box bezeichnet [FKM$^+$06, S. 1].

Um eine solche Wissensbasis in maschinenverständlicher Form beschreiben zu können, sind formale Sprachen notwendig. Das World Wide Web Consortium (kurz W3C) hat zu diesem Zweck Standardisierungen entsprechender Techniken im World Wide Web definiert. Die Web Ontology Language (OWL) basiert technisch auf der Syntax des Resource Description Frameworks (RDF) und geht über die Ausdrucksmächtigkeit des Vokabulars von RDF-Schema (RDFS) hinaus. Die Sprache lässt sich in ihrer zweiten Version OWL 2 auf die Beschreibungslogik SROIQ(D) abbilden. Um den Anforderungen verschiedener Nutzergruppen gerecht werden zu können, existieren unterschiedliche Ausprägungen der Sprache, welche sich bezüglich ihrer semantischen Ausdrucksstärke hierarchisch wie folgt einteilen lassen:

$$\text{OWL 2 EL, OWL 2 QL, OWL 2 RL} \sqsubseteq \text{OWL 2 DL} \sqsubseteq \text{OWL 2 Full}$$

Die Profile OWL 2 EL, OWL 2 QL und OWL 2 RL stellen jeweils eine Untermenge der in OWL 2 DL verwendbaren Sprachkonstrukte dar. Da sie erst seit dem Update auf OWL 2 existieren, werden sie nachfolgend vereinfacht durch OWL EL, QL und RL bezeichnet. Die Wahl zwischen diesen Profilen ist davon abhängig, welche Aussagekraft zur Beschreibung der Ontologie erforderlich ist, welche Rolle das Reasoning dabei spielt, wie groß die zu untersuchende Datenmenge ist und wie viel Wert auf deren Skalierbarkeit gelegt wird. OWL EL eignet sich besonders für große Ontologien mit geringer semantischer Komplexität wie beispielsweise die medizinische Terminologie SNOMED CT, da durch die Restriktion der verwendbaren Sprachkonstrukte ein besonders schnelles Reasoning innerhalb der T-Box möglich ist [OWL12,

2.4 Profiles]. OWL QL und RL sind eher in Verbindung mit schlichter gehaltenen Ontologien - sogenannten „Lightweight Ontologies" - zu verwenden, welche mit großen Datenmengen arbeiten müssen. *Schlicht* meint in diesem Zusammenhang, dass diese Ontologien keine komplexen Relationen in der T-Box enthalten. Während OWL QL bevorzugt in Verbindung mit relationalen Datenbanken genutzt werden soll, wobei der Datenzugriff beispielsweise über SQL realisiert werden kann, eignet sich OWL RL eher für einen direkten Zugriff auf Daten in Form von RDF-Tripel [OWL12, 2.4 Profiles]. OWL 2 Full ist die mächtigste Ausprägung der Web Ontology Language und bietet den maximal möglichen Umfang der Sprache. Durch die Möglichkeit der Verwendung aller definierten Sprachkonstrukte - wie etwa auch dem der Reifikation - ist OWL 2 Full in Bezug auf das Reasoning nicht entscheidbar. Das bedeutet, dass es durchaus sein kann, dass der Reasoning-Prozess unendlich lange dauert und zu keinem Ergebnis führt.

Bezüglich der Syntax von Ontologien gibt es verschiedene Serialisierungen für eine solche Wissensbasis. In der vom W3C erstellten Standardisierung werden hierzu die Varianten RDF/XML, OWL/XML, Functional Syntax, Turtle und die seit dem Update auf OWL 2 definierte Manchester Syntax aufgeführt [OWL12, 2.2 Syntaxes]. Im Verlauf dieser Arbeit werden alle Code-Beispiele in der Turtle-Syntax dargestellt, da deren Struktur für das menschliche Auge gut zu überblicken ist und die Notation von Sprachkonstrukten verglichen mit den anderen Syntax-Varianten am platzsparendsten ist.

Im Folgenden werden die für das geplante Softwaretool benötigten ontologiebasierten Methoden vorgestellt.

3.1 Reasoning

Der große Vorteil von Ontologien ist, dass sie in Form einer formal definierten Sprache beschrieben sind, die der Technologie eine semantische Aussagekraft verleiht. Mit Ausnahme von OWL 2 Full basieren alle Ausprägungen der Web Ontology Language auf einer entscheidbaren Beschreibungslogik. Das ermöglicht das Ziehen von logischen Schlussfolgerungen, welche als gültig und schlüssig (englisch: valid and sound) eingestuft werden. Ein Schluss ist logisch gültig, wenn es auf Grund seiner Form unmöglich ist, dass die Prämissen wahr sind, jedoch die Konklusion falsch ist. Das folgende Beispiel entspricht einer gültigen Argumentation:

„Jeder Astrologe ist auch ein Pathologe." ∧
„Pathologen haben Medizin studiert."
———————————————————
→ „Astrologen haben Medizin studiert."

„Ein Argument heißt schlüssig genau dann, wenn es gültig ist und alle seine Prämissen wahr sind." [Bec03, S. 22] In einigen Fällen wird die im Englischen als soundness bezeichnete Eigenschaft an Stelle von Schlüssigkeit auch mit dem Wort Korrektheit übersetzt [Feh13, S. 53]. Das obere Beispiel ist demzufolge zwar gültig, jedoch nicht schlüssig (nicht korrekt), da die Prämisse „Jeder Astrologe ist auch ein Pathologe." nicht wahr ist. Das nachfolgende Beispiel ist sowohl gültig als auch schlüssig:

„Pathologen sind Ärzte." ∧
„Ärzte haben Medizin studiert."
———————————————————
→ „Pathologen haben Medizin studiert."

Durch die Möglichkeit des Schlussfolgerns enthält eine Ontologie nicht nur das Wissen, welches in Form von Axiomen explizit formuliert ist, sondern darüber hinaus auch das durch diese implizit ableitbare Wissen. Dieses Schlussfolgern von Informationen auf Basis des vorhandenen Wissens wird im Kontext von Ontologien als Reasoning bezeichnet. Die Komponente, welche das Reasoning technisch umsetzt, wird Reasoner genannt.

Dabei ist zu beachten, dass das Ziehen von logischen Schlussfolgerungen in OWL allgemein auf dem Konzept der so genannten Open World Assumption (kurz OWA) basiert. Diese „Offene-Welt-Annahme" besagt, dass ein Reasoner erst dann etwas als nicht existent bezeichnet, wenn explizit in der Wissensbasis definiert ist, dass es nicht existiert. Vereinfacht ausgedrückt, heißt das, dass nur weil ein Fakt nicht in der Wissensbasis steht, ein Reasoner nicht annimmt, dass dieser Fakt falsch ist. Es wird angenommen, dass dieser Fakt durchaus existieren kann und lediglich noch nicht zur aktuellen Wissensbasis hinzugefügt worden ist.

Da Ontologien auf einer formal definierten Semantik beruhen, lässt sich der Reasoning-Prozess, bei dem neues Wissen aus einer gegebenen Klassen-Hierarchie gefolgert wird, auch mathematisch ausdrücken:

$$\mathbf{c_1, c_2: T(i, rdf{:}type, c_1) \wedge T(c_1, rdfs{:}subClassOf, c_2) \rightarrow T(i, rdf{:}type, c_2)}$$

[Sac16, S. 5]

Allgemein besagt diese Formel, dass für zwei Kassen c_1 und c_2 gilt, dass wenn das Tripel T(i, rdf:type, c_1) und das Tripel T(c_1, rdfs:subClassOf, c_2) in einer Wissensbasis existieren, ein Reasoner automatisch das Tripel T(i, rdf:type, c_2) ableiten kann. Ontologisch formuliert bedeutet das: Wenn i ein Individuum der Klasse c_1 ist und c_1 eine Unterklasse der Klasse c_2, dann ist i auch ein Individuum der Klasse c_2.

Das folgende in Turtle-Syntax notierte Beispiel veranschaulicht, wie so etwas innerhalb einer Ontologie aussehen kann. Die Buchstaben vor dem Doppelpunkt kennzeichnen jeweils den Namensraum, in dem die darauffolgende Ressource definiert ist, wobei `bsp:` hier lediglich für „Beispiel" steht und somit keinen real existierenden Namensraum benennt. `rdf:` steht dagegen für das Resource Description Framework und `rdfs:` für das RDF-Schema. Eine Ressource kann sowohl eine Klasse als auch ein Individuum oder eine Relation (Property) sein. Jedes in Turtle-Syntax beschriebene Axiom besteht aus einer Sequenz von Subjekt, Prädikat und Objekt, welche durch einen Punkt terminiert ist. Die Reihenfolge dieser drei jeweils durch ein Leerzeichen voneinander getrennten Bestandteile ist dabei stets dieselbe. Axiome, welche Aussagen über Individuen treffen, werden oft auch als Fakten bezeichnet.

Wenn die beiden Axiome

```
bsp:AloisAlzheimer rdf:type bsp:Neuropathologe.
```

und

```
bsp:Neuropathologe rdfs:subClassOf bsp:Pathologe.
```

bekannt sind, dann schlussfolgert ein Reasoner automatisch, dass

```
bsp:AloisAlzheimer rdf:type bsp:Pathologe.
```

In natürlicher Sprache formuliert: Wenn Alois Alzheimer ein Neuropathologe ist und jeder Neuropathologe auch ein Pathologe, dann ist Alois Alzheimer ebenfalls ein Pathologe.

Der mathematische Ausdruck für eine Folgerung von Klassenzugehörigkeiten von Entitäten anhand von Domain- und Range-Restriktionen einer Property sieht wie folgt aus:

$$\begin{aligned}\forall\ \mathbf{i_1, i_2, c_1, c_2, p:}\quad & \mathbf{T(i_1, p, i_2)} \wedge \\ & \mathbf{T(p, rdfs{:}domain, c_1)} \wedge \mathbf{T(p, rdfs{:}range, c_2)} \\ & \rightarrow \mathbf{T(i_1, rdf{:}type, c_1)} \wedge \mathbf{T(i_2, rdf{:}type, c_2)}\end{aligned}$$

[Sac16, S. 7]

Wenn i_1 und i_2 durch Property p miteinander verbunden sind und für p die Klasse c_1 als Domain und die Klasse c_2 als Range definiert sind, dann ist i_1 ein Individuum der Klasse c_1 und i_2 ein Individuum der Klasse c_2.

In Turtle lässt sich auf diese Weise mit Hilfe eines Reasoners feststellen, dass, wenn die drei Axiome

`bsp:ConradMurray bsp:istLeibarztVon bsp:MichaelJackson.` und

`bsp:istLeibarztVon rdfs:domain bsp:Arzt.` und

`bsp:istLeibarztVon rdfs:range bsp:Patient.`

existieren, dann automatisch die Informationen

`bsp:ConradMurray rdf:type bsp:Arzt.` und

`bsp:MichaelJackson rdf:type bsp:Patient.` hergeleitet werden können.

Ebenso, wie anhand von Klassenhierarchien, lassen sich auch Rückschlüsse aus Relationshierarchien ziehen:

$$\forall\ \mathbf{i_1, i_2, p_1, p_2:}\quad \mathbf{T(i_1, p_1, i_2)} \wedge \mathbf{T(p_1, rdfs{:}subPropertyOf, p_2)} \rightarrow \mathbf{T(i_1, p_2, i_2)}$$

[Sac16, S. 10]

Wenn i_1 und i_2 durch Property p_1 miteinander verbunden sind und für p_1 SubProperty der Property p_2 ist, dann sind i_1 und i_2 auch durch Property p_2 miteinander verbunden.

Wenn beispielsweise in einer Wissensbasis gilt, dass

`bsp:PieroDaVinci bsp:istVaterVon bsp:LeonardoDaVinci.` und

`bsp:istVaterVon rdfs:subPropertyOf bsp:istVorfahrVon.`,

dann gilt implizit auch

`bsp:PieroDaVinci bsp:istVorfahrVon bsp:LeonardoDaVinci.`

3.2 Ontology Learning

Es gibt verschiedene Ansätze, die bei der Erstellung einer Ontologie verfolgt werden können. Noy und McGuinness [NM01] beschreiben beispielsweise eine Design-Methode namens „Ontology Development 101", bei der die Wissensbasis nach der initialen Festlegung der zu modellierenden Domäne durch sechs iterativ ablaufende Prozesse geformt wird. Zu Beginn werden sogenannte Kompetenzfragen formuliert, welche später durch die Ontologie beantwortet werden sollen und somit den Umfang der Domäne beschreiben. Anschließend wird geprüft, ob sich diese Domäne durch eine oder mehrere bereits vorhandene Ontologien beschreiben lässt. Je nachdem, wie aufwändig das Anpassen an die eigenen Zwecke erscheint, wird die gewünschte Wissensbasis zunächst aus Teilen anderer zusammengesetzt oder von Grund auf neu erstellt. Im weiteren Verlauf werden die noch fehlenden benötigten Klassen, Relationen, Restriktionen und Instanzen modelliert. Durch den iterativen Charakter dieser Methode kann jederzeit von einem Design-Schritt zu einem vorherigen gesprungen werden, um Elemente zu ergänzen.

Letztendlich gibt es keinen vorgeschriebenen Standard, nach welcher Design-Methode eine Ontologie zu erstellen ist. „There is no one correct way to model a domain - there are always viable alternatives." [NM01, S. 4]

Unabhängig vom Design ist jedoch eines allen Wissensmodellierungen gemein: Das Erstellen einer Ontologie erfordert Expertenwissen über die zu modellierende Domäne. Experten stellen im Allgemeinen eine knappe, begehrte Ressource dar. Da somit die Kreationsphase einer Wissensbasis je nach Größe und Anwendungsgebiet sehr zeit- und kostenintensiv sein kann, besteht großes Interesse daran, diesen Schritt vollständig oder zumindest teilweise zu automatisieren. Ontology Learning lässt sich ins Deutsche etwa mit „Ontologie-Lernen" übersetzen und bezeichnet den Prozess, bei dem eine Ontologie durch teil- oder vollautomatische Verfahren erstellt, verändert oder erweitert wird. Erstmals geprägt von Maedche und Staab [MS01] werden für den Lernprozess mittlerweile auch die Bezeichnungen Ontology Generation, Ontology Mining und Ontology Extraction verwendet [Sac15a, S. 2].

Die Aufgabe besteht dabei nicht zwangsläufig darin, eine Ontologie von Grund auf neu zu erstellen. Es ist ebenfalls denkbar, Elemente eines bestehenden Modells zu verändern (Ontology Update) oder ohne deren Veränderung das Konstrukt zu erweitern beziehungsweise zu spezifizieren (Ontology Enrichment oder Tuning). Auch ein halb oder vollständig automatisiertes Übersetzen einer Wissensbasis in eine andere Sprache - zum Beispiel von FOL[1] in OWL 2 DL - gilt als Ontology-Learning-Prozess. [Sac15a, S. 16-17]

[1] First Order Logic, deutsch: Prädikatenlogik erster Stufe

Im Zuge dieser Aufgaben können unterschiedliche Methoden zum Einsatz kommen. Die Modellierung der Wissensbasis unter Einsatz von Techniken aus den Bereichen Text-Mining und Natural-Language-Processing (NLP, deutsch: Automatische Sprachverarbeitung) ist dabei ebenso möglich, wie die Unterstützung durch Verfahren aus dem sogenannten Linked Data Mining, bei denen mittels statistischer Betrachtungen nach Mustern innerhalb von RDF-Graphen gesucht wird. Ein weiteres Hilfsmittel stellt die aus dem Bereich des maschinellen Lernens stammende Induktive Logische Programmierung (ILP) dar. Hierbei wird versucht, aus bestehenden Daten - wie etwa einer existierenden Ontologie - allgemeingültige Schemainformationen für die zu modellierende Ontologie zu finden. Ein vierter Ansatz namens Crowdsourcing zielt darauf ab, die Schnelligkeit von Computersystemen mit der Genauigkeit von Menschen zu vereinen. So können zum Beispiel große Taxonomien, welche zwar sehr viele Elemente enthalten, jedoch kein komplexes Fachwissen voraussetzen, teilweise durch Jobs auf Plattformen wie Amazon Mechanical Turk [2] und durch Nutzer-Interaktionen in Online-Spielen mit Hilfe fremder menschlicher Ressourcen erstellt werden. [LV14]

Die Wahl einer bestimmten Ontology-Learning-Methode ist abhängig vom Strukturierungsgrad der zur Verfügung stehenden Informationen. Die Einteilung der Input-Daten für die Wissensmodellierung erfolgt in drei Gruppen:

- Strukturierte Daten (z. B. aus Datenbanken) [VOS03]
- Semi-strukturierte Daten (z. B. aus HTML- oder XML-Files)
- Unstrukturierte Daten (Freitext aus z. B. Text-Dokumenten, TXT-Files)

Während sich strukturierte Daten durch Machine-Learning-Verfahren wie der induktiven logischen Programmierung analysieren lassen, sind für halb- und unstrukturierte Input-Daten zunächst Vorverarbeitungen mittels NLP-Techniken notwendig, um deren Inhalt auf eine strukturiertere und damit maschinenverständliche Abstraktionsebene zu heben. [CMSV09, S. 246]

Das Identifizieren von Termen, Konzepten, Relationen und auch komplexen Axiomen aus textuellen Informationen und das anschließende Konstruieren einer Ontologie wird im Englischen als „Ontology Learning from Text" bezeichnet [Sac15a, S. 4], [Cim06]. Die dabei angewandten Techniken der automatischen Sprachverarbeitung (NLP) reichen von der initialen Tokenisierung[3], welche den Fließtext in Einheiten auf Wortebene segmentiert, über das sogenannte Part-of-speech-Tagging[4], bei dem Wörter und Satzzeichen des Textes zu

[2] MTurk: https://www.mturk.com
[3] Demonstration von Tokenisierung, implementiert im Python-Natural-Language-Toolkit (NLTK): http://text-processing.com/demo/tokenize/
[4] Demonstration von Part-of-Speech-Tagging: http://cogcomp.cs.illinois.edu/page/demo_view/POS

Wortarten zugeordnet werden, bis hin zur Syntaxanalyse eines Satzes[5] durch geeignete Parser, die beispielsweise Personen, Orte, Zahlen und Zeitangaben aus dem Text herausfiltern.

Die Datenquelle, die dem Ontology-Learning-Prozess als Input dient, besteht zumeist nicht nur aus einem einzelnen, sondern aus mehreren Dokumenten, welche in ihrer Gesamtheit als Korpus bezeichnet werden. Durch das Identifizieren und Extrahieren der für diesen Korpus relevanten Terme entsteht eine Terminologie, welche den Wortschatz der betrachteten Dokumentensammlung widerspiegelt. Aus dieser Terminologie werden durch Konzeptualisierung die Konzepte für die zu erstellende Wissensbasis gebildet. [Sac15a, S. 5]

3.2.1 Ontology Learning Layer Cake

Die einzelnen Teilaufgaben beim Ontology Learning sowie die zwischen ihnen existierenden konzeptionellen Abhängigkeiten lassen sich in einem Schichtdiagramm visualisieren. Cimiano [Cim06] beschreibt diese Grafik im Jahr 2006 unter der Bezeichnung „Ontology Learning Layer Cake", was auf deutsch übersetzt so viel heißt wie „Schichttorte des Ontologie-Lernens". Im Fokus steht dabei das Erlernen der T-Box der Wissensbasis. Die Modellierung des assertorischen Wissens der A-Box wird im nachstehenden Unterabschnitt 3.2.3 unter dem Titel „Ontology Population" beschrieben. Abb. 3.1 zeigt das Schichtdiagramm nach Vorbild von Cimiano [Cim06, S. 23]. Links von den einzelnen Schichten stehen konkrete Beispiele aus dem medizinischen Bereich, um den Output der jeweiligen Schicht zu veranschaulichen.

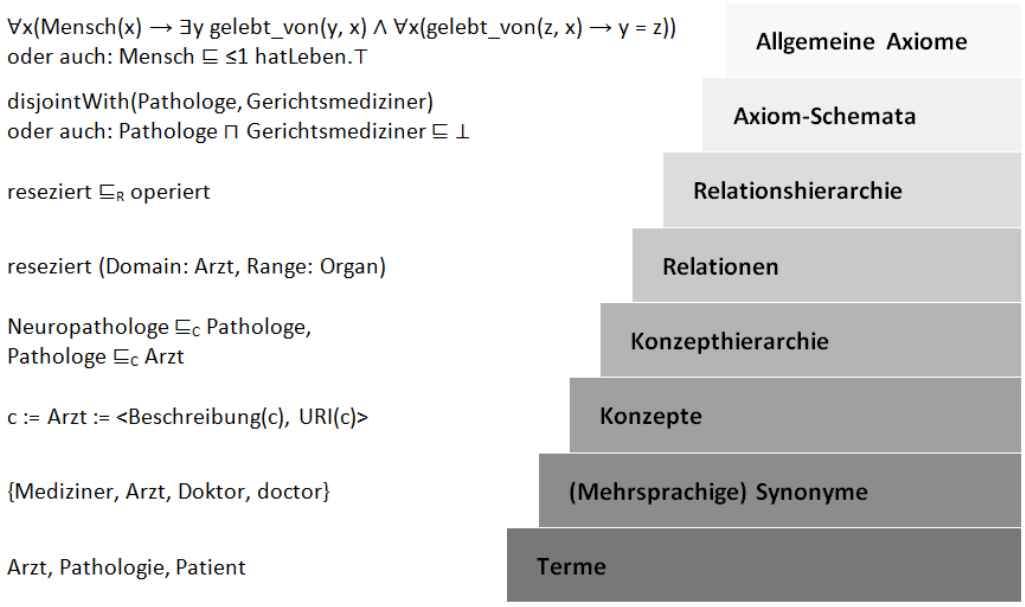

Abbildung 3.1: Ontology Learning Layer Cake nach Cimiano [Cim06, S. 23]

[5] Demonstration von Syntaxanalyse: http://nlpviz.bpodgursky.com

Die einzelnen Schichten des Diagramms bauen aufeinander auf, das bedeutet, dass das Ergebnis einer unteren Schicht als Input für die höheren dient. So macht es zum Beispiel erst Sinn, sich mit den Relationen zwischen Konzepten zu befassen, wenn die Konzepte und deren hierarchische Gliederung bekannt sind. Andernfalls können Domain- und Range-Restriktionen der Beziehungen nicht korrekt der Hierarchie entsprechend bestimmt werden. Es ist jedoch nicht zwingend notwendig, beim Erlernen einer Ontologie die einzelnen Schichten für den gesamten zu betrachtenden Korpus streng nacheinander abzuarbeiten. Es gibt durchaus Algorithmen wie etwa die Formale Konzeptanalyse, in deren Verlauf versucht wird, Konzepte und deren hierarchische Einteilung zur gleichen Zeit zu entdecken. Die Betrachtung der oberen beiden Schichten ist ebenso simultan möglich (siehe 3.2.1.7 Axiom-Schemata und allgemeine Axiome). [Cim06, S. 22]

Die beiden unteren Schichten des Schichtdiagramms entsprechen der lexikalischen Ebene des Ontology Learning. Hier besteht die Aufgabe darin, zunächst die relevanten Terme zu identifizieren, welche später in ihrer Gesamtheit die Terminologie für das weitere Vorgehen bilden. Außerdem werden erkannte Synonyme zusammengefasst, damit später für jedes real existierende Konzept auch nur ein Konzept in der Wissensbasis modelliert wird und nicht mehrere Konzepte mit derselben semantischen Bedeutung existieren. Die extrahierten Terme und Synonymgruppen dienen als Grundlage für die Erstellung der Konzepte. Konzepte unterscheiden sich von Termen dadurch, dass sie ontologische Einheiten (Entities) sind und damit Abstraktionen menschlichen Denkens im Sinne von Ganter und Wille [GW99]. Die nächsthöheren Ebenen des Diagramms entsprechen in aufsteigender Reihenfolge den Aufgaben des Lernens der Konzepthierarchie, der Relationen sowie deren Hierarchie. Die oberen beiden Schichten, welche das Erkennen standardisierter Axiome und die Ableitung beliebiger Regeln repräsentieren, zeigen besonders anspruchsvolle Aufgaben. So gibt es für die Komplexität des letztgenannten Bereichs keine strikte, standardisierte Begrenzung. In der Praxis jedoch werden die erlaubten Axiome durch die Wahl der Wissensrepräsentationssprache wie zum Beispiel OWL DL eingeschränkt. Im Folgenden sind die einzelnen Aufgaben Schicht für Schicht beschrieben. [CMSV09, S. 250-251]

3.2.1.1 Terme

In der untersten Schicht des Diagramms besteht die Aufgabe darin, die Begriffe aus den zur Verfügung stehenden Input-Daten zu extrahieren, welche für die Beschreibung der Domäne relevant sind und somit im späteren Verlauf beispielsweise als Konzept oder Relation modelliert werden. Das Identifizieren und Extrahieren dieser Terme kann daher als Grundvoraussetzung für alle weiteren Teilaufgaben des Ontology Learning betrachtet werden [Cim06, S. 23].

Ein Verfahren, das hierbei zum Einsatz kommen kann, ist beispielsweise die Zählung der Frequenz, mit der ein Begriff in einem gegebenen Satz von linguistisch vorverarbeiteten Dokumenten vorkommt. Die Methode basiert auf der Annahme, dass ein innerhalb eines fachspezifischen Korpus häufig vorkommender Begriff ein für diese Domäne relevantes Konzept identifiziert. Forschungen im Bereich des Information Retrieval (IR) haben gezeigt, dass es effektivere Methoden für die Gewichtung von Begriffen gibt als das simple Zählen von Frequenzen. So ist hier beispielsweise auch die Anwendung des Tf-idf-Maßes[6] denkbar. Im Bereich der Computerlinguistik gibt es darüber hinaus eine Reihe von komplexeren Techniken wie etwa die Bestimmung des C-Werts und des NC-Werts, welche von Frantzi et al. [FAM00] beschrieben wird. Diese Messung berücksichtigt nicht nur die Häufigkeit der Begriffe, sondern auch die Tatsache, dass diese ineinander verschachtelt vorkommen können. So stecken beispielsweise im Term *Ontology-Learning-Prozess* die Begriffe *Ontology Learning* und *Prozess*. Im Englischen werden diese verschachtelten Terme „nested terms" genannt und treten durch die sprachspezifische Bildung von zusammengesetzten Substantiven durch simples Aneinanderreihen der Einzelterme relativ häufig auf. [CMSV09, S. 251]

Obwohl es scheint, als sei das Feld der Term-Extraktion dank der Vielzahl an untersuchten Methoden ausgereift, gibt es unter den Experten kein allgemein anerkanntes Verständnis davon, welche Techniken für welchen Zweck am besten geeignet sind. Bestimmte Domänen wie beispielsweise die Medizin benötigen entsprechende Anpassungen der Verfahren in Bezug auf ihre spezifische Charakteristik. [CMSV09, S. 251]

3.2.1.2 Synonyme

Um Synonyme zu extrahieren, setzen die meisten Ansätze auf die „Distributionelle Hypothese" von Harris [Har54], nach der Wörter in dem Maße semantisch ähnlich sind, in dem sie syntaktische Kontexte teilen. Das bedeutet, es besteht ein Zusammenhang zwischen der Bedeutungsähnlichkeit von Wörtern und der Ähnlichkeit ihrer Verteilung über bestimmte sprachliche Kontexte, die aus empirischen Korpora gewonnen werden. Somit lässt sich beispielsweise erkennen, dass die Wörter *Arzt* und *Mediziner* von der Bedeutung her eine gewisse Ähnlichkeit aufweisen. Beide lassen sich unter anderem in den Kontext der Wörter *untersuchen*, *Diagnose* und *behandeln* bringen. Die Wörter *Operation* und *Station* teilen sich dagegen einen weitaus geringeren sprachlichen Kontext, tauchen jedoch in einigen Fällen beide eventuell in Verbindung mit den Wörtern *Krankenhaus* und *leiten* auf. Je nach Größe und Inhalt eines Korpus lassen sich auf diese Weise domänenspezifische Synonyme erkennen.

[6] Tf-idf: Term Frequency (Vorkommenshäufigkeit) und Inverse Document Frequency (inverse Dokumenthäufigkeit), siehe [BYRN99, S. 29-30]

Diese Hypothese steht im Einklang mit der bekannten Aussage von John R. Firth „You shall know a word by the company it keeps" [Fir57], welche übersetzt etwa so zu verstehen ist, dass sich die Bedeutung eines Wortes anhand seiner Begleitung erkennen lässt. Firth nennt das gemeinsame Vorkommen von Wörtern „Kollokation". Ein wichtiger Teil der Bedeutung eines Wortes seien seine Kollokate, also die Wörter, mit denen es in Sätzen und Texten zusammen auftritt. In dem Wort *Nacht* steckt somit beispielsweise auch die Bedeutung seiner Kollokate *dunkel*, *still*, *Mond* und *Sterne* [Fil12, S. 116].

Die Bedeutung eines Wortes w wird mittels eines Vektors v_w repräsentiert, welcher die statistische Verteilung des Wortes über relevante sprachliche Kontexte darstellt. Berechnet wird die semantische Ähnlichkeit zwischen einzelnen Wörtern beispielsweise über den Jaccard-Koeffizienten [Jac01]. Dabei wird die Anzahl der gemeinsamen Elemente (Schnittmenge) durch die Größe der Vereinigungsmenge geteilt:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Die Ähnlichkeit im Vektorraum zwischen verschiedenen Wortvektoren kann berechnet und besonders ähnliche Wörter können unter Berücksichtigung eines festzulegenden Schwellwerts als Synonyme betrachtet werden.

Andere Techniken für das Erkennen von Synonymen verwenden statistische Methoden in Verbindung mit Online-Suchmaschinen [WPT+09] oder bedienen sich der Vielzahl von bereits vordefinierten Synonymen aus der Welt des Onlinelexikons Wikipedia [SCE15]. Ein weiterer Ansatz ist die Berechnung der semantischen Verwandtschaft auf Basis einer Taxonomie oder eines semantischen Netzes wie WordNet[7] [PPM04]. WordNet gliedert Worte mit synonymer Bedeutung in sogenannte Synsets und liefert lexikalische Beziehungen zwischen diesen. Jedes Synset entspricht demzufolge einem Konzept und kann beispielsweise über die Relationen der Art Hyperonymie/Hyponymie („ist eine Art von") sowie Holonymie/Meronymie („ist Teil von") mit anderen Synsets verbunden sein. So ist zum Beispiel das Synset mit der Bedeutung *Rechteck* als Hyperonym des Synsets *Quadrat* definiert, da ein Quadrat eine spezielle Art eines Rechtecks ist. Der Ausdruck *Computer* ist ein Holonym zu *CPU*, *Arbeitsspeicher*, *Tastatur* und anderen Synsets, die Bestandteile eines Computers beschreiben. Darüber hinaus liefert das semantische Netz zu jedem Synset eine natürlichsprachliche Definition - eine sogenannte Glosse. [CMSV09, S. 253]

Zur Steigerung der Sicherheit und Zuverlässigkeit bei der Identifizierung von Synonymen ist auch die Kombination von Methoden, welche auf der Distributionellen Hypothese beruhen, mit WordNet-basierten Verfahren möglich [AAH+09].

[7] WordNet: [Fel98], https://wordnet.princeton.edu/

3.2.1.3 Konzepte

Das Erlernen von Konzepten lässt sich nach Cimiano et al. [CMSV09, S. 251] in drei Paradigmen gliedern:

- Konzeptionelle Clusterbildung
- Linguistische Analyse
- Induktive Methoden

Ansätze der **Konzeptionellen Clusterbildung** wie die Formale Konzeptanalyse nach Ganter und Wille [GW99] werden eingesetzt, um Konzepte zu bilden und diese gleichzeitig hierarchisch zu sortieren. Demzufolge laufen die in Abb. 3.1 dargestellten Aufgaben der Schichten 3 und 4 nicht nacheinander sondern zeitgleich ab. Ansätze dieser Art erzeugen eine intentionale Beschreibung jedes Konzepts in Bezug auf die Attribute, die es mit anderen Konzepten teilt sowie auf diejenigen, die es von anderen Konzepten unterscheidet.

Bei Anwendung von **Linguistischen Analysetechniken** wird zu jedem Begriff eine intentionale Beschreibung in Form einer natürlichsprachlichen Definition erstellt. Die Vorgehensweise von Velardi et al. [VNCN06] nutzt WordNet, um einen zusammengesetzten Term zu interpretieren und als Nebenprodukt eine Beschreibung auf Grundlage der WordNet-Glosse der einzelnen Terme zu erstellen. So ist zum Beispiel die Definition des Terms *knowledge management practice* aus den Glossen von *knowledge management* und *practice* zusammengesetzt [VNCN06, S. 12]. Damit dieser Prozess die jeweils richtigen Definitionen zusammenschließt, ist die Disambiguierung (Vereindeutlichung) eines jeden Wortes erforderlich. Das bedeutet, dass es bei mehrdeutigen Begriffen für jede Bedeutung eine eindeutige Repräsentation in der lexikalischen Datenbank geben muss. Aus diesem Grund gibt es beispielsweise für das Wort *cat* in WordNet acht verschiedene Repräsentationen mit unterschiedlichen Bedeutungen von der Hauskatze, über die Großkatze bis hin zur Abkürzung für die Computertomographie (CAT = Computed Axial Tomography). Neben der Disambiguierung sind Regeln notwendig, nach denen die Generierung der zusammengesetzten Definitionen erfolgt.

Bei vorhandenen Instanzen in der Wissensbasis können **Induktive Methoden** wie etwa die Induktive Logische Programmierung zur Erstellung von Regeln verwendet werden, die eine Gruppe von Instanzen intentional beschreiben. Ein solcher Ansatz kann zum Beispiel dazu dienen, eine Taxonomie zu reorganisieren oder Lücken in Konzept-Definitionen zu entdecken. [CMSV09, S. 254]

3.2.1.4 Konzepthierarchie

Es gibt verschiedene Methoden, taxonomische Beziehungen zwischen Konzepten aus Texten zu erlernen. Nachfolgend werden mit der Analyse von lexiko-syntaktischen Mustern, der Cluster- und der Phrasenanalyse sowie der Verfeinerung einer bestehenden Hierarchie vier Ansätze erläutert.

Analyse lexiko-syntaktischer Muster: Einer der ältesten Ansätze untersucht maschinenlesbare Wörterbücher auf Regelmäßigkeiten bezüglich der Formulierung oder Repräsentation der einzelnen Einträge. Unter der Annahme, dass gleiche semantische Beziehungen innerhalb der entsprechenden Wörterbucheinträge ähnlich zum Ausdruck gebracht werden, zielt diese Methode darauf ab, Muster zu definieren, anhand derer sich beispielsweise Hyponymie-Relationen („ist eine Art von", englisch: is-a relation) automatisiert erkennen und extrahieren lassen [Als87], [Cal84]. Hearst [Hea92] definiert eine Sammlung von lexiko-syntaktischen Mustern, die diese Hyponymie-Beziehungen nicht aus maschinenlesbaren Wörterbüchern, sondern aus frei formulierten Fließtexten lernen können. [CMSV09, S. 254]

Clusteranalyse: Dieser Prozess kann als Organisation von Objekten in Gruppen definiert werden, wobei sich Mitglieder einer Gruppe hinsichtlich einer bestimmten Repräsentation (z. B. als Vektor) eine oder mehrere gleiche Eigenschaften teilen. Im Folgenden werden gemäß Cimiano [CMSV09, S. 255] drei Arten von Clusterverfahren vorgestellt:

Die **divisiven Clusterverfahren** werden auch als Top-down-Verfahren bezeichnet. In der Initialisierungsphase stellt die Menge aller Terme zunächst einen einzigen großen Cluster dar. Im Verlauf des Verfahrens werden in der sogenannten Verfeinerungsphase durch das Aufspalten des größten beziehungsweise des am wenigsten homogenen Clusters schrittweise kleinere Cluster erzeugt. Der Verfeinerungsprozess endet, wenn ein zuvor definiertes Anhaltekriterium erreicht ist oder läuft soweit, bis letztlich jedes Cluster nur noch aus einem einzigen Objekt besteht.

Die **agglomerativen Clusterverfahren** folgen dem entgegengesetzten sogenannten Bottom-up-Ansatz. In der Initialisierungsphase wird jeder Term als eigenes Cluster definiert. Anschließend werden in der Wachstumsphase durch die Verschmelzung der ähnlichsten beziehungsweise am wenigsten ungleichen Cluster immer größere Cluster zusammengeschlossen. Ist hier kein Anhaltekriterien definiert, terminiert der Prozess, wenn alle Terme zu einem einzigen großen Cluster gehören.

Sowohl die divisiven als auch die agglomerativen Verfahren gehören zu den Techniken der hierarchischen Clusteranalyse und können demzufolge zur Herstellung einer hierarchischen Struktur von Termen verwendet werden. Beide Verfahrensarten beruhen auf der Betrachtung von (Un-)Ähnlichkeiten, für die eine Reihe von Messverfahren existieren wie zum Beispiel der Euklidische Abstand, die L1-Norm, der Kosinus oder der Jaccard-Koeffizient, zusammengefasst in [Lee99, S. 27]. Nachteil dieser Verfahren ist, dass einmal gebildete Cluster nicht mehr verändert werden können. An der entstandenen Hierarchie ist nicht mehr erkennbar, wie sie berechnet wurde. Darüber hinaus liefert die hierarchische Clusteranalyse keine Clustermodelle. Je nach verwendeten Messverfahren können Ketteneffekte entstehen, die aus Ausreißern oft kleine Cluster aus wenigen Termen werden lassen. Diese kleinen Cluster müssen dann nachträglich analysiert werden, um für sie ein allgemeines Modell zu beschreiben.

Bei den **konzeptionellen Clusterverfahren**, welche bereits als eines der drei Paradigmen für das Erlernen von Konzepten im vorherigen Abschnitt 3.2.1.3 aufgeführt wurden, werden zu Konzepten zusammengefasste Terme durch Knotenpunkte in einer Gitter- oder Baumstruktur dargestellt. Algorithmen, die das konzeptionelle Clusterverfahren anwenden, sind zum Beispiel COBWEB [Fis87] und SUBDUE [JACH02]. Da das Erlernen von Konzepten und deren hierarchische Gliederung bei konzeptionellen Clusterverfahren nicht nacheinander erfolgen muss, können jederzeit neue Konzepte in die vorhandene Hierarchie integriert werden. Dabei ist auch eine Veränderung bestehender Strukturen (Cluster) möglich, was ein Vorteil gegenüber den divisiven und agglomerativen Verfahren ist. Nicht zuletzt auf Grund dieser Flexibilität führen konzeptionelle Clusterverfahren oft zu einem besseren Ergebnis als Varianten der anderen beiden Arten [CHS05, S. 332].

Phrasenanalyse: Einige Ansätze beruhen auf der Tatsache, dass die interne Struktur von Nominalphrasen (NP) genutzt werden kann, um taxonomische Relationen zu entdecken. Im Wesentlichen bauen diese Methoden auf der Heuristik auf, dass Adjektive oder auch Substantive, die als Begleiter des Kern-Substantivs der Nominalphrase auftreten, typischerweise eine Unterklasse der durch den Kern angegebenen Klasse definieren. Die Begleiter können dem Kern-Substantiv dabei sowohl voran- als auch nachgestellt sein. Das bedeutet zum Beispiel, dass *fokale Epilepsie* als Unterklasse von *Epilepsie* interpretiert wird. Diese und weitere Regeln für ein Mapping zwischen linguistischen Annotationen und Klassen sind beispielsweise in dem von Buitelaar et al. [BOS04] implementierten Protégé-Plug-in OntoLT enthalten. Voraussetzung für diesen Prozess ist, dass die einzelnen Bestandteile der Nominalphrasen mit den entsprechenden linguistischen Annotationen versehen sind - wie zum Beispiel head für die Kern-Substantive und mod für die Begleiter [BOS04, Kapitel 3: Linguistic Annotation].

Verfeinerung einer bestehenden Klassenhierarchie: Wenn bereits eine ausreichend umfangreiche Klassenhierarchie gegeben ist - beispielsweise durch Basis-Kategorien aus einer breit aufgestellten, allgemeinen Quelle wie WordNet - ist es möglich, durch neue Terme bezeichnete Konzepte als Unterklassen zu vorhandenen Klassen zu modellieren. Dadurch sind diese automatisch in die gegebene Hierarchie integriert. Demnach wird die hierarchische Gliederung hierbei nicht umstrukturiert, sondern lediglich durch hinzukommende Unterklassenbeziehungen verfeinert.

Die Verteilungsdarstellung eines Wortes durch einen Vektor (v_w) wird dann verwendet, um aus einem Trainingskorpus und den vordefinierten Konzepten mit deren lexikalischen Einträgen einen Klassifikator zu erlernen. Anschließend lässt sich die Vektordarstellung von relevanten, noch nicht klassifizierten Termen konstruieren und der gelernte Klassifikator wird genutzt, um einen Vorschlag zur Integration der aus dem neuen Begriff definierten Klasse in die betrachtete Hierarchie zu erhalten. Große lexikalische Datenbanken wie WordNet eignen sich besonders zum Testen solcher Algorithmen [Wid03].

Ausschlaggebend für die Zuordnung dieser Methode zu den Lernverfahren der T-Box ist, dass die neuen Terme jeweils Konzepte bezeichnen, welche anschließend als Klassen modelliert werden. Der Prozess, bei dem aus gefundenen Begriffen Instanzen generiert werden, heißt Klassenzuordnung und wird im Unterabschnitt 3.2.3 Ontology Population thematisiert.

3.2.1.5 Relationen

Für das Entdecken von Beziehungen zwischen Worten finden beim Ontology Learning verschiedene Techniken aus den Bereichen des maschinellen Lernens (ML) und der automatischen Sprachverarbeitung (NLP) Anwendung. Das gemeinsame Auftreten von Worten innerhalb eines bestimmten Bereichs, das heißt zum Beispiel innerhalb eines Satz oder eines Absatzes, kann Identifikator für Assoziationen zwischen diesen Worten sein. Häufige Kookkurrenzen von Begriffen sprechen für eine grammatikalische oder semantische Abhängigkeit dieser voneinander. Maedche und Staab [MS00] verwenden einen Algorithmus zur Assoziationsanalyse und stellen Kookkurrenzen von Worten innerhalb eines Satzes als Transaktionen dar. Diese Repräsentation erlaubt es, Support und Zuversicht (englisch: confidence) für binäre Transaktionen zu berechnen und damit anonyme binäre Assoziationen zwischen Worten zu detektieren. So fanden sie dadurch beispielsweise heraus, dass zwischen den Worten *Raum* und *Ausstattung*, dargestellt durch die binäre Transaktion (room furnishing), eine signifikantere anonyme Assoziation besteht als zwischen den Worten *Raum* und *Fernsehen* (room television) [MS00, S. 10].

In der Linguistik wird die Aufgabe der Entdeckung starker Zusammenhänge zwischen Worten als Ermittlung von Kollokationen (nach Firth [Fir57]) bezeichnet. Im Wesentlichen, geht es darum, Wörter zu entdecken, die unter Ausschluss des Zufalls mit statistisch signifikanter Häufigkeit zusammen auftreten. Die statistische Signifikanz wird dabei mit Hilfe eines Tests wie zum Beispiel des Chi-Quadrat-Tests (χ^2-Test) oder durch den t-Test von Gosset alias Student [Wil08] überprüft [CMSV09, S. 257].

Ein weiterer Ansatz ist das Erlernen von Relationen anhand linguistischer Annotationen. In Verbindung mit NLP-Techniken wie etwa einem Parser werden dabei insbesondere die als Prädikat eines Satzes eingesetzten Verben betrachtet [SB05], [CHR06].

3.2.1.6 Relationshierarchie

Zentrale Herausforderung beim Erlernen von Beziehungen ist es, die richtige Abstraktionsebene für die Domain- und Range-Festlegungen der betrachteten Relation in Bezug auf die Konzepthierarchie zu finden. Für diese Aufgabe gibt es verschiedene Herangehensweisen. Während Maedche et al. [MS00] die Konzepthierarchie in den Assoziationsanalyseprozess mit einbringen, benutzen Cimiano et al. [CHR06, S. 192-193] berechnete Werte wie beispielsweise die bedingte Wahrscheinlichkeit, um die Hierarchie-Ebene für Domain und Range zu ermitteln. Dabei wird das Konzept gewählt, welches mit maximal berechneter statistischer Signifikanz entlang der Hierarchie das spezifischste - das heißt, das am weitesten unten stehende - ist. So wird versucht, die Relationen auf genau der Hierarchie-Ebene zu schließen, die statistisch so passend wie möglich und in Bezug auf die Konzeptbeschreibung so allgemein wie nötig ist.

3.2.1.7 Axiom-Schemata und allgemeine Axiome

Die obersten beiden Schichten des Ontology Learning Layer Cake (Abb. 3.1) beschäftigen sich mit dem Erkennen und Erlernen von Axiom-Schemata sowie komplexen Regeln für Ontologien. Während es in den unteren Schichten darum geht, durch eher einfache Klassen- und Relationshierarchien ein Grundgerüst für die Wissensbasis zu konstruieren, zielt dieser Schritt darauf ab, ausdrucksstärkerer, spezifischere Axiome zu modellieren.

Dazu gehören beispielsweise Eigenschaften wie die Disjunktheit von Konzepten oder die Symmetrie sowie die Transitivität von Relationen. OWL besitzt für solche Zwecke vordefinierte Axiome, welche bereits in der formalen Sprachdefinition enthalten sind (siehe [OWL04, Abschnitt 5.3 Disjoint Classes und 3.3 Property Characteristics]). Diese vordefinierten Axiome

werden auch **Axiom-Schemata** genannt. So gibt es zum Beispiel halbautomatisierte Verfahren aus dem Bereich des überwachten Lernens, die auf die Identifizierung von Disjunktheit zwischen Konzepten abzielen [VVSH07].

Auch das Erlernen von nicht innerhalb der OWL-Sprachdefinition vorgefertigten, **allgemeinen Axiomen** nach Vorbild der logischen Implikation ist möglich. Völker et al. [VHC07] beschreiben einen Ansatz zur Erzeugung von formalen Klassenbeschreibungen aus natürlichsprachlichen Definitionen, welche zum Beispiel aus Online-Glossaren und Enzyklopädien wie Wikipedia extrahiert sind. Dabei wird versucht, die Syntax von natürlichsprachlichen Definitionen basierend auf lexikalisch-syntaktischen Mustern in OWL DL-Axiome umzuwandeln. So wird beispielshaft der Satz „Enzyme sind Proteine, die chemische Reaktionen katalysieren." in den beschreibungslogischen Ausdruck Enzym $\equiv$ (Protein $\sqcap$ $\exists$katalysieren.(Chemisch $\sqcap$ Reaktion)) übersetzt [VHC07, S. 670-671].

3.2.2 Anpassung bestehender Ontologien

Ein vergleichsweise einfacher Ansatz für die Erzeugung einer Domänenontologie unter Gegebenheit eines Korpus ist, eine vorhandene allgemeine Ontologie anzupassen. Buitelaar et al. [BS01] beschreiben ein Verfahren, mit dem WordNet-Synsets je nach Relevanz in Bezug auf den betrachteten Korpus auf Grundlage eines frequenzbasierten Messverfahrens gerankt werden können. Diese Technik zur Anpassung von WordNet an eine bestimmte Domäne durch Reduzierung der gegebenen Synsets wird auch Pruning genannt (englisch: to prune = beschneiden, reduzieren) und kann ebenfalls dazu verwendet werden, eine existierende Ontologie zu reduzieren.

Auch Kietz et al. [KVM00] verwenden als Ausgangslage auf der einen Seite eine bestehende, allgemein gehaltene Ontologie oder ein semantisches Netz mit Wörterbuch-Charakter, wie etwa WordNet oder GermaNet[8], und auf der anderen Seite einen domänenspezifischen Korpus. Darüber hinaus wird ein zweiter Korpus benötigt, der wie die semantische Quelle thematisch allgemein gehalten ist - zum Beispiel der Artikelbestand einer Tageszeitung. Nachdem die beiden erst genannten Quellen zur Erstellung der Wissensbasis genutzt wurden, wird mit Hilfe des unspezifischen Korpus versucht, die für die zu beschreibende Domäne irrelevanten Konzepte zu entfernen. Ein Konzept gilt dann als relevant, wenn es um einen bestimmten Faktor c mal öfter im domänenspezifischen als im allgemeinen Korpus vorkommt. Hierbei ist c eine vom Benutzer angegebene Konstante und zur Bestimmung der Relevanz wird das Tf-idf-Maß verwendet.

[8] GermaNet: [HF97], http://www.sfs.uni-tuebingen.de/GermaNet/

3.2.3 Ontology Population

Als Ontology Population wird der Prozess bezeichnet, bei dem durch Klassen repräsentierte Konzepte aus der T-Box einer Ontologie durch konkrete Individuen in der A-Box instanziiert werden. Anhand des semiotischen Dreiecks nach Sowa [Sow00] erklärt, bedeutet das, dass den Konzepten dazugehörige Objekte zugeordnet werden. Dieser Vorgang entspricht aus Sicht der objektorientierten Programmierung (OOP) der Instanziierung von Klassen durch Objekte. Mit Objekt oder Instanz ist im Kontext von Ontologien die Repräsentation eines Individuums in der Wissensbasis gemeint (owl:NamedIndividual). Im Zuge der Ontology Population wird demnach das assertorische Wissen der A-Box einer Ontologie modelliert. In der Literatur wird oft auch von einer Instanziierung von Relationen gesprochen. Damit sind in OWL formulierte Objekt- und Datentyp-Relationen gemeint, welche nicht zwischen Klassen untereinander (owl:ObjectProperty) oder Klassen und Datentypen (owl:DatatypeProperty) bestehen, sondern konkrete Individuen miteinander beziehungsweise mit definierten Datenwerten verbinden.

Hinsichtlich der Instanziierung von Klassen ist der im Zuge der Ontology Population ablaufende Prozess im Allgemeinen als **Klassifizierung**, beziehungsweise genauer noch als **Klassenzuordnung** oder auch **Klassierung**, bekannt. Die Gesamtheit aller Klassen innerhalb der Ontologie bildet demnach eine Klassifikation. [Wik16, Kapitel 1 Begriffsabgrenzung]

Wenn die T-Box einer Wissensbasis alle erforderlichen Konzepte in Form von Klassen modelliert, das heißt, wenn bereits eine ausreichend umfassende Klassifikation gegeben ist, dann können aus einem Text extrahierte Begriffe durch entsprechende Klassenzuordnungen voll- oder halbautomatisiert als Individuen der dazugehörigen Klassen modelliert werden.

Wie bereits beim Erlernen der T-Box, können auch für die Modellierung der A-Box je nach Strukturierungsgrad der Inputdaten Methoden aus unterschiedlichen Bereichen verwendet werden. So können als Basis für die Klassifizierung beispielsweise der k-Nearest-Neighbor-Algorithmus (k-NN) [dFE08] oder Support Vector Machines (SVM) [XLLL08] genutzt werden.

Witte et al. [WKR10] nutzen NLP-Techniken, um Fließtexte zu annotieren. Diese Annotationen identifizieren im Text erwähnte *Personen* oder *Firmen*. So kann jeder als *Person* annotierte Begriff in Form eines Individuums der entsprechenden OWL-Klasse zugeordnet werden. Auf gleiche Weise werden Instanzen der Relation *arbeitet_für* erkannt und verbinden die *einzelnen Personen* mit den als dazugehörig erkannten *instanziierten Firmen* [WKR10, S. 3847].

Geleijnse und Korst [GK05] beschreiben eine Methode, welche Instanzen von Klassen und Relationen mittels Google-Suchanfragen erlernen. So wird zum Beispiel für das modellierte Individuum *Johan Cruijff* die Suchanfrage „Johan Cruijff was born in" generiert, um einerseits generell Instanzen von Orten zu finden und andererseits den gesuchten Geburtsort *Amsterdam*. Dieser kann wiederum für die Suche „wurde geboren in Amsterdam" verwendet werden, um weitere dort geborene Personen zu finden. [GK05, Kapitel 2 Global Approach]

Boer et al. [BSW06] konzentrieren sich in ihrer Arbeit auf das Instanziieren von Relationen und stellen dazu ein Verfahren vor, welches auf Informationen aus heterogenen Internetressourcen basiert und dabei unabhängig von Strukturierung, Domäne und Sprache der gegebenen Dokumente agiert. Die verglichen mit korpus- oder domänenspezifischen Verfahren geringere Präzision der verwendeten allgemeinen Methoden soll dabei durch die Redundanz der Informationen, die sich aus der Kombination vieler verschiedener Quellen ergibt, kompensiert werden. [BSW06, Abschnitt 3.1 Approach]

3.2.4 Herausforderungen

Ontology Learning kann als eine Art Reverse-Engineering-Prozess betrachtet werden. Der Autor eines Textes oder einer Datei hat beim Schreiben eine Welt im Kopf, die er bis zu einem gewissen Maß mit anderen Domainexperten teilt. Diese Welt stellt ein implizites Modell der betrachteten Domäne dar und hat ebenso Einfluss auf den erstellten Text wie die Intention, mit der er verfasst wurde. Die Rekonstruktion dieser Welt ist Aufgabe des Ontology Learning und zumindest aus zwei Gründen komplex und herausfordernd: Zum einen enthält der betrachtete Text zumeist nur einen Teil des Domänenwissens seines Verfassers und kann demzufolge auch nur einen Teil zur Rekonstruktion der zu modellierenden Welt beitragen. Zum anderen ist das Wissen über grundlegende Fakten als Voraussetzung dazu, die Welt verständlich beschreiben zu können, nur selten explizit formuliert - sofern es sich beim untersuchten Text nicht um ein Lehr- oder Wörterbuch handelt [Cim06, S. 20]. So behaupten Brewster et al. [BCW03] beispielsweise, dass das Schreiben und Lesen von Texten im Grunde genommen eine Art Pflege eines bestimmten Wissensstandes ist, das bedeutet, dass für das Verständnis eines Textes meist ein gewisses Hintergrundwissen vorausgesetzt wird und nur das eigentliche Thema einen Wissenszuwachs durch explizit formulierte Fakten darstellt. Demzufolge sind die Kerninformationen der meisten Texte zwar modellierbar, jedoch ohne Hintergrundwissen nicht verständlich interpretierbar. Eine Herausforderung beim Erlernen einer Ontologie mittels Ontology Learning ist somit, festzulegen, welche Dateien und Dokumente kombiniert werden müssen, damit die Gesamtheit der Daten eine ausreichend umfassende Modellierung der Domäne ermöglicht. Informationen, die nicht im

gewählten Input-Korpus enthalten sind, können im Verlauf des Ontology Learning nicht automatisiert modelliert werden. Lassen sich diese Informationen auch durch Reasoning nicht herleiten, fehlen sie in der Wissensbasis. Unvollständiges Wissen kann dazu führen, dass die erstellte Ontologie die Aufgaben, zu deren Zweck sie entwickelt wurde, entweder gar nicht, nicht vollständig oder nicht korrekt bearbeiten kann. [Cim06, S. 20]

Eine Herausforderung während der Ontology Population ist das Erkennen von Instanzen, wenn die dazugehörigen Klassen nicht definiert sind. Im Kontext des semiotischen Dreiecks nach Ogden und Richards [OR23] sowie nach Sowa [Sow00] gesprochen: Es ist eine Herausforderung, ein Wort als Symbol zu erkennen, welches ein Objekt bezeichnet, wenn es dafür noch kein Konzept in der Wissensbasis gibt. Wird ein Symbol nicht als solches erkannt, wird kein entsprechendes Individuum in der A-Box modelliert. Erkannte Symbole, für die in der T-Box keine passende Klasse definiert ist, können zwar formal als Individuum erstellt, jedoch nicht durch Klassierung (rdf:type) der entsprechenden Klasse zugeordnet werden. [Cim06, S. 21]

Lehmann und Völker [LV14, S. XI] beschreiben für das Ontology Learning sechs grundsätzliche Herausforderungen: Heterogenität, Unsicherheit, Reasoning, Skalierbarkeit, Qualität und Interaktivität.

Heterogenität

Daten, die im Internet zur Verfügung stehen, können sich in vielerlei Hinsicht unterscheiden. Beispielkriterien für einen solchen Vergleich sind das Format, die Sprache, die beschriebene Domäne und die Qualität der Daten. Buitelaar et al. [BCF+08] beschreiben einen Ansatz, nach dem eine Wissensbasis aus heterogen formatierten Daten wie zum Beispiel Fließtext, Tabellen und Bildbeschriftungen gewonnen werden kann. Jedoch gibt es in der Praxis bisher weder für die Integration einer solchen Methode noch für die Homogenisierung von Daten einen unter Ontology-Learning-Experten anerkannten und verbreiteten Standard. [LV14, S. XI]

Unsicherheit

Unstrukturierte Daten, die nur schwer für Maschinen interpretierbar und damit ungeeignet für das Modellieren von Ontologien sind, führen nur mit geringer Wahrscheinlichkeit zu einem korrekten und vollständigen Ergebnis. Hauptgrund dafür ist, dass die Vorverarbeitung der Daten beispielsweise durch NLP-Methoden nicht für jeden Fließtext gleichermaßen qualitativ

hochwertige Entitäten herauskristallisiert. Deshalb sind viele Ontology-Learning-Methoden so gestaltet, dass sie jedes einzelne Ergebnis mit einem Sicherheitswert assoziieren, der das Vertrauen der jeweiligen Methode in Bezug auf die Richtigkeit der erzielten Ergebnisse widerspiegelt. Solche Werte zur Beschreibung von (Un-)Sicherheiten zusammen mit Informationen bezüglich der Integrität (Zeitstempel) und Authentizität (Datenherkunft) sind besonders wichtig, wenn es darum geht, erlernten Ontologien manuell oder automatisch zu debuggen. [LV14, S. XII]

Reasoning

Häufig werden Ontologien für Anwendungen erstellt, die vom Schlussfolgern neuer Fakten durch logische Inferenz (Reasoning) Gebrauch machen. Für den Reasoning-Prozess müssen die Ontologien logisch konsistent sein, was wiederum bedeutet, dass das Ontology-Learning-Verfahren dazu in der Lage sein müssen, jederzeit konsistente und kohärente Ontologien zu erstellen. Daher beruhen nicht nur Methoden zum Herleiten von Konzepten in Beschreibungslogiken, sondern auch solche zum Erlernen von Ontologien auf logischem Schlussfolgern. Das von Lehmann und Bühmann [LB10] präsentierte Tool ORE[9] realisiert eine enge Kopplung zwischen dem Ontology Learning und dem Ontologie-Debugging, das heißt der Inkonsistenz-Diagnose mit gegebenenfalls anschließender Reparatur. [LV14, S. XII]

Skalierbarkeit

Extrahieren von Wissen aus den wachsenden Datenmengen im Internet - unstrukturierte, textuelle Daten einerseits und strukturierte Daten wie Datenbanken, Linked Data oder Ontologien auf der anderen Seite - erfordert skalierbare und effiziente Ansätze. Speziell wenn es darum geht, aus verteilten, lose miteinander verbundenen Daten zu lernen oder Wissen aus mehreren Quellen zu integrieren, stehen Ontology-Learning-Methoden vor großen Herausforderungen. Um diesen zu begegnen, werden verschiedene Strategien verfolgt, wie etwa verteilte Berechnungen für die horizontale Skalierung des Ontology Learning, inkrementelle Lernansätze zur Wiederverwendung vorhandenen Wissens oder Sampling [HLA09] und Modularisierung zur Verbesserung der Effizienz von Ontologie-Lernalgorithmen. [LV14, S. XII]

[9] ORE: Ontology Repair and Enrichment, http://dl-learner.org/community/ore-tool/

Qualität

Entscheidend für die Beurteilung des Erfolgs oder Misserfolgs eines Ontology-Learning-Prozesses ist die Antwort auf die Frage, wie gut das entstandene Modell für den gewünschten Zweck geeignet ist. Grundsätzlich lässt sich die Qualität einer automatisch generierten Ontologie ebenso messen wie die Qualität einer jeden anderen. Jedoch ist die Evaluation einer Wissensbasis keine leichte Aufgabe. Formale Korrektheit, Vollständigkeit und Konsistenz sind nur einige von vielen möglichen Kriterien für die Beurteilung der Qualität einer Ontologie. Brank et al. [BGM05] und Vrandečić [Vra09] geben einen Einblick in die Vielzahl an Möglichkeiten zur Ontologie-Evaluation. Letztendlich gibt der Anwendungskontext, für den die Ontologie erstellt wurde, die Wahl der Evaluationskriterien und den geforderten Qualitätsstandard vor. Deshalb ist es von Vorteil, jeden Schritt des Ontology Learning in Hinblick auf die jeweilige Domäne oder gewünschte Anwendung zu optimieren. Dazu gehören bereits die Wahl der Inputdaten und deren Vorverarbeitungen. [LV14, S. XII]

Interaktivität

In der Praxis hängt die Qualität einer gelernten Ontologie oft vom Grad der Automatisierung ab. Je geringer das Ausmaß, in dem der Mensch an dem Prozess der Ontologie-Generierung beteiligt ist, desto geringer die zu erwartende Qualität. Eine Ontologie, die zwar vollautomatisch erzeugt wurde, jedoch letztlich für die beabsichtigte Anwendung qualitativ nicht ausreichend ist, erfordert zeitaufwendige Nachbearbeitungsschritte. Obwohl die Nachbearbeitung nicht generell Part der automatischen Ontologie-Generierung ist, kann sie durch Ontology-Learning-Methoden unterstützt werden. So liefern einige Verfahren beispielsweise wertvolle Konfidenz- und Relevanzwerte für einzelne Axiome oder Zeitstempel und Schlüsselinformationen über das eingesetzte Lernverfahren. Die in der Einleitung dieses Kapitels angesprochenen Möglichkeiten zur Einbeziehung von Nicht-Experten in den Ontologie-Erstellungsprozess mittels Crowdsourcing oder interaktiven Spielen ist zwar auf der einen Seite entlastend für Domänen- und Ontologie-Experten, bringt allerdings auch einige Risiken, zum Beispiel bezüglich der Genauigkeit der erstellten Wissensbasis mit sich. Darüber hinaus ist auch das Übersetzten von Spieler-Interaktionen in Ontologie-Modellierungsentscheidungen keinesfalls ein triviales Problem. [LV14, S. XII-XIII]

3.3 Mapping

Das Thema Mapping beschäftigt sich mit der Verknüpfung von Entitäten unterschiedlicher Wissensbasen. Bei der Entwicklung einer Ontologie für einen bestimmten Diskursbereichs kann es vorkommen, dass Konzepte repräsentiert werden, die ebenfalls in anderen Ontologien dargestellt sind. So gibt es auch in der im Zuge dieser Arbeit erstellten Report-Ontologien Inhalte, die sich in bereits existierenden formalen Wissensbasen wiederfinden lassen. Durch die Verbindung dieser Elemente wird das Wissen über eine modellierte Entität in Wissensbasis w_1 mit dem Wissen über das korrespondierende Element aus einer anderen Wissensbasis w_2 angereichert, ohne dass dieses dabei explizit in w_1 beschrieben werden muss.

Der dem Mapping vorausgehende Prozess, bei dem Korrespondenzen zwischen zwei oder mehreren Ontologien gesucht werden, wird als Matching oder Alignment bezeichnet [PFS+05, S. 7]. Der Schwellwert, ab dem eine Ähnlichkeit oder eine Verwandtschaft von Konzepten zu einer Korrespondenz der entsprechenden Klassen führt, wird dabei über geeignete, im Vorfeld definierte Parameter bestimmt. Auch zusätzliche Ressourcen, wie zum Beispiel Thesauri, formal definierte Regeln oder das Allgemeinwissen des Ontologie-Modellierers, können beim Finden von Korrespondenzen helfen.

H. Sack [Sac15b] beschreibt eine Korrespondenz als Ergebnis eines Matchings formal als ein Quintupel der Art

$$\langle \mathbf{id}, \mathbf{e_1}, \mathbf{e_2}, \mathbf{r}, \mathbf{n} \rangle.$$

Die **id** repräsentiert hierbei einen eindeutigen Identifikator der jeweiligen Korrespondenz, während die beiden zu verlinkenden Entitäten durch $\mathbf{e_1}$ und $\mathbf{e_2}$ bestimmt sind. Der Parameter **r** gibt die Relation an, durch die $\mathbf{e_1}$ und $\mathbf{e_2}$ miteinander verknüpft werden. Dabei sind sowohl Äquivalenz ($=$), Generalisierung ($\sqsupseteq$, $\geq$), Spezialisierung ($\sqsubseteq$, $\leq$) und Disjunktheit ($\perp$) als auch jede sonstige in einer der ansprechbaren Ontologien definierte Relation möglich. Mit **n** wird ein Vertrauenswert für die beschriebene Korrespondenz angegeben, der üblicherweise im Wertebereich von 0 bis 1 liegt.

Durch ein solches Quintupel können sowohl einfache Korrespondenzen der Art

- *http://dbpedia.org/resource/Tim_Berners-Lee = http://www.freebase.com/m/07d5b*
- *Vorfahr* $\geq_{1,0}$ *Mutter*
- *rdfs:label* $\geq_{0,9}$ *skos:prefLabel*

als auch komplexere Exemplare, wie zum Beispiel

- $Stunde = Sekunde \times 3600$
- $Buch(x) \wedge authorIst(x,y) \wedge Author(y) \Rightarrow_{0,85} geschreibenVon(x,\ concat(y.Vorname,\ y.Nachname))$

formal notiert werden. [Sac15b]

Bei den Techniken, die für das Matching von Ontologien eingesetzt werden, gibt es elementbasierte und strukturbasierte Ansätze. Zu jeder der zwei Kategorien gibt es unterschiedliche Vorgehensvarianten. Abb. A.1 zeigt eine Einteilung der verschiedenen Ontologie-Matching-Techniken nach Euzenat und Shvaiko [ES07].

Zu der Kategorie der elementbasierten Techniken gehören beispielsweise der stringbasierte und der sprachbasierte Ansatz. Das stringbasierte Verfahren durchsucht die Input-Ontologien nach Entitäten mit gleichem Namen oder gleicher Beschreibung. Je nach Festlegung eines Schwellwertes können hier auch Korrespondenzen zwischen Entitäten mit ähnlich geschriebenem Namen oder Beschreibung vorgeschlagen werden. Ein sprachbasierter Algorithmus verwendet ein Lexikon oder einen Thesaurus, um die Entitäten auf Homonyme, Synonyme und Partonyme zu untersuchen [Sac15b].

Das graphbasierte Verfahren stellt eine strukturbasierte Matching-Technik dar. Hierbei werden Ontologien als benannte Graphen betrachtet. Die Suche nach Korrespondenzen erfolgt unter der Annahme, dass gleiche Knoten auch gleiche Nachbarknoten haben. Nach demselben Muster arbeitet auch ein auf Taxonomie basierter Matching-Algorithmus, der sich dabei allerdings auf das Finden von Generalisierungen ($\sqsupseteq$, $\geq$) und Spezialisierungen ($\sqsubseteq$, $\leq$) beschränkt. [Sac15b]

Für das Verbinden von zwei oder mehreren Ontologien nach dem Matching gibt es zwei unterschiedliche Möglichkeiten: Das Merging (deutsch: Zusammenführung) und das Mapping (deutsch: Zuordnung). Während beim Merging aus den zwei oder mehreren auf Korrespondenzen untersuchten Ontologien eine neue Ontologie erstellt wird, welche dann die verglichenen zusammenfasst und ersetzt, werden beim Mapping die Verbindungen der betroffenen Entitäten zwischen den Input-Ontologien beschrieben, wobei diese (bis auf die hinzugekommenen Verbindungsbeschreibungen) unverändert bestehen bleiben. [PFS+05, S. 6-7]

4 Das Softwaretool

Während Abschnitt 4.1 die Konzeption des geplanten Programms inklusive der vorzubereitenden Importdaten und genutzten Schnittstellen darlegt, enthält Abschnitt 4.2 die konkrete Beschreibung von dessen Aufbau und Funktionalität.

4.1 Konzeption

Das Softwaretool ist für den nachfolgend beschriebenen Arbeitsablauf zur dynamischen Erzeugung eines pathologischen Befundberichts, zu dessen Speicherung sowie dessen späterer Bearbeitung konzipiert. Als inhaltliche Vorlage für die zu generierenden Befundberichtsvorlagen werden die im Unterabschnitt 4.1.1 beschriebenen Quellen als Input verwendet.

Zu Beginn der Erstellung eines neuen Berichts werden dem Pathologen auf einer grafischen Benutzeroberfläche alle Fragen angezeigt, die generell für eine vollständige Bearbeitung der gewählten Berichtsart beantwortet werden müssen. Die Auswahl bestimmter Antwortmöglichkeiten kann weitere Fragen mit sich führen, welche dann ebenfalls an entsprechender Stelle visualisiert werden. Wenn der Pathologe beispielsweise angibt, für die Beurteilung einer Probe aus einer radikalen Prostatektomie Lymphknoten entnommenen zu haben, ergibt sich daraus die Frage nach deren Lage. Andernfalls ist diese Frage überflüssig. Daher erscheint sie erst dann, wenn das Vorhandensein von Lymphknoten entsprechend konstatiert wurde.

Im Verlauf der Berichterstellung prüft das Programm im Hintergrund nach jeder gegebenen Antwort, ob der Report vollständig ist. Das Wissen darüber, welche Elemente in einem vollständigen Bericht enthalten sein müssen, erlangt die Software aus der Ontologie des entsprechenden Report-Typs, welche im Zuge eines Ontology-Learning-Verfahrens aus einer international zusammengestellten Vorlage erstellt worden ist. Die Einzelheiten dazu sind in den folgenden Abschnitten näher erläutert.

Um den Pathologen nicht zu stark in seiner Arbeitsweise zu limitieren, ist es jederzeit möglich, den aktuellen Report zu speichern und zu einem späteren Zeitpunkt wieder zu öffnen.

4.1.1 Inputs

Für die Implementierung der Software sind maßgeblich zwei Arten von Inputdaten vorgesehen. Zum einen entspricht die Repräsentation eines Reports einer in OWL formulierten Wissensbasis, zum anderen wird das Expertenwissen zur Generierung eines neuen Report-Typs sowie zur Beurteilung von dessen Vollständigkeit aus einer Excel-Datei importiert. Da der Grundaufbau des pathologischen Befundberichts bei Programstart intern modelliert wird, verlangt das Erstellen eines neuen spezifischen Report-Typs keine gesonderte OWL-Datei als Input. Dabei wird der internen Wissensbasis das Wissen aus der zu selektierenden Excel-Datei hinzugefügt. Wenn jedoch eine bereits erstellte Report-Vorlage oder sogar ein in der Vergangenheit abgespeicherter Befundbericht geladen werden soll, muss der Import einer OWL-Datei möglich sein. Nachfolgend werden der Grundaufbau der Report-Ontologie und die zu dessen Erweiterung genutzten Excel-Dateien beschrieben.

4.1.1.1 Basis der Report-Ontologie

Die Ontologie, die zur Generierung des Befundberichts genutzt wird, enthält dessen grundlegende Struktur in der T-Box und die von der zu untersuchenden Krebsart abhängigen Elemente sowie die fallspezifischen Angaben in der A-Box. Sie wurde initial mit Hilfe der Open-Source-Software Protégé erstellt und ist auf der beiliegenden Programm-CD in der von Beckett et al. [BBLPC14] definierten Turtle-Syntax der Sprache OWL unter dem Namen *APRBasis.owl* gespeichert.

Vor der Ontology Population, dem Hinzufügen der konkreten Fragen und Antwort-Elemente, besteht die Ontologie ausschließlich aus terminologischem Wissen, welches die Konzepte der Reportelemente in Form von Klassen und deren Relationen untereinander modelliert. Die Struktur des Berichts ist mit der Struktur eines Fragebogens vergleichbar. Aus diesem Grund ist die T-Box an ein Modellierungsbeispiel von Allemang und Hendler aus dem Buch „Semantic Web for the Working Ontologist" angelehnt [AH08, S. 179-196, Kapitel 9].

Der Befundbericht besteht im Wesentlichen aus einer Anzahl von Fragen (`:Question`), denen jeweils eine Auswahl möglicher Antworten (`:Answer`) zugewiesen ist. Die Beziehung zwischen einer Frage und einer ihr zugeordneten Antwortmöglichkeit wird mit `:hasOption` bezeichnet und stellt eine *owl:ObjectProperty* dar. Die inverse Beziehung dazu heißt `:isOptionOf`. Mit Hilfe der Relation `:hasSelectedOption` kann gespeichert werden, welche Antwortmöglichkeiten zu einer Frage ausgewählt wurden. Da `:hasSelectedOption` eine untergeordnete Beziehung (*rdfs:subPropertyOf*) von `:hasOption` ist, gelten hier dieselben Bestimmungen für Domain und Range. Der Grundaufbau des Befundberichts lässt sich wie in Listing

4.1 formulieren. Sofern nicht anders angegeben, sind alle in dieser Arbeit aufgeführten Ontologie-Code-Auszüge in Turtle-Syntax dargestellt.

Listing 4.1: Grundaufbau eines Befundberichts als Fragebogen

```
:Answer rdf:type owl:Class ;
        rdfs:label "Answer"@en ,
                   "Antwort"@de .

:Question rdf:type owl:Class ;
          rdfs:label "Frage"@de ,
                     "Question"@en .

:hasOption rdf:type owl:ObjectProperty ;
           rdfs:domain :Question ;

           rdfs:range :Answer ;
           owl:inverseOf :isOptionOf ;
           rdfs:label "has option"@en ,
                      "hat Option"@de .

:isOptionOf rdf:type owl:ObjectProperty ;
            rdfs:label "is option of"@en ,
                       "ist Option von"@de .

:hasSelectedOption rdf:type owl:ObjectProperty ;
                   rdfs:subPropertyOf :hasOption ;
                   rdfs:label "has selected option"@en ,
                              "hat ausgewaehlte Option"@de .
```

Jeweils vor dem Doppelpunkt ist der Namensraum (englisch: Namespace) der entsprechenden Entität angegeben. Die im Anhang befindliche Tabelle A.2 zeigt die für diese Arbeit relevanten Namensräume. So ist zum Beispiel die Relation `rdf:type` im Namespace `rdf` des Resource Description Frameworks definiert. Die formale Beschreibung dieser Beziehung, das bedeutet, die dahinterliegende semantische Aussage, ist nicht innerhalb der hier beschriebenen OWL-Datei enthalten, sondern gemäß Tabelle A.2 innerhalb der Datei *22-rdf-syntax-ns*, welche unter der URI *http://www.w3.org/1999/02/22-rdf-syntax-ns* zu finden ist. Nachfolgend wird die Relation `rdf:type` durch ein `a` abgekürzt, wie es in Turtle-Syntax üblich ist. Steht vor dem Doppelpunkt kein Namensraum, so ist die neu entwickelte Ontologie gemeint. Da Sie innerhalb der Datei *APRBasis.owl* beschrieben wird, gilt Sie hier als Basis und braucht keinen speziell definierten Namensraum.

Für in Abb. 4.4 gezeigten Auszug aus dem ICCR-Datensatz für einen Prostatektomie-Befundbericht werden im Zuge des Ontology-Population-Prozesses die in Listing A.1 dargestellten Fragen und Antwortmöglichkeiten modelliert (Z. 1-59).

Das zu entwickelnde Programm soll dabei helfen, einen inhaltlich vollständigen Befundbericht zu erstellen, indem es alle in der Ontologie definierten Fragen an den Pathologen richtet

und dessen Antworten fortwährend abspeichert. Dabei wird zu jeder Instanz der Klasse `:Question` der durch die Relation `rdfs:label` zugeordnete Text sowie die zu dieser Frage möglichen Antwort-Instanzen der Klasse `:Answer` und deren Label auf dem Bildschirm angezeigt. Wenn es am Ende des Programmdurchlaufs keine offenen Fragen gibt, wurde der Pathologe an alle wichtigen Inhalte seines spezifischen Befundberichts erinnert und der Report kann als inhaltlich vollständig identifiziert werden. Fragen, für die schon eine Antwort gespeichert wurde, müssen deshalb folgerichtig als beantwortet gelten. Bisher kann beispielsweise bei der Beantwortung der ersten Frage mit der zweiten Antwortmöglichkeit das Tripel `:indQuestion01 :hasSelectedOption :indAnswer01.02` in die Wissensbasis gespeichert werden. Die erste Antwortmöglichkeit stellt einen Sonderfall dar, da bei ihrer Auswahl ein zusätzliches Textfeld erscheint, das die Eingabe eines konkreten Wertes fordert. Diese Sonderfälle sind durch den Teilstring „(specify)" im Report zu erkennen. Die im zusätzlichen Textfeld eingegebene Antwort wird über die Datentyp-Relation `:hasAnswerText` in die Wissensbasis gespeichert.

Damit bereits beantwortete Fragen auch als solche erkannt werden können, ist eine Klasse `:AnsweredQuestion` wie in Listing 4.2 definiert.

Listing 4.2: Ontologie-Erweiterung durch eine Klasse für beantwortete Fragen

```
1 :AnsweredQuestion a owl:Class ;
2         owl:equivalentClass [ a owl:Restriction ;
3                               owl:onProperty :hasSelectedOption ;
4                               owl:someValuesFrom :Answer
5                             ] ;
6         rdfs:label "Answered Question"@en ,
7                    "Beantwortete Frage"@de .
```

Die Klassenbeschreibung enthält eine someValuesFrom-Restriktion (Z. 2-5). In OWL können neue Klassen mit Hilfe von bereits definierten Klassen, Eigenschaften oder auch Instanzen beschrieben werden. So kann mit Hilfe einer Restriktion (*owl:Restriction*) eine Klasse durch die Beschreibung der Individuen, die sie enthält, modelliert werden [AH08, S. 180]. Die someValuesFrom-Restriktion wird verwendet, um Einschränkungen für Individuen zu beschreiben, welche über eine Relation p mit einer Klasse c verbunden sind [AH08, S. 184]. Das syntaktische Element *owl:onProperty* gibt die zu betrachtende Beziehung p an. Da es sich bei c um eine Klasse (*owl:Class*) handelt, muss p eine *owl:ObjectProperty* sein. Die eckigen Klammern um die Restriktion weisen darauf hin, dass es sich hierbei um eine anonyme Klasse (auch Blank Node oder bnode) handelt, welche alle Individuen enthält, für die diese Einschränkung gilt.

Da die Wissensbasis zum Tripel `:indQuestion01 :hasSelectedOption :indAnswer01.01` zusätzlich die Instanziierung `:indAnswer01.01 a :Answer` enthält, kann durch die in Listing 4.2 dargestellte Äquivalenzbeziehung zwischen der Klasse `:AnsweredQuestion` und der OWL-Restriktion das Tripel `:indQuestion01 a :AnsweredQuestion` geschlussfolgert werden. Das Besondere dabei ist, dass letztgenanntes Tripel nicht explizit in der Wissensbasis existieren muss, da es durch Reasoning gefolgert wurde und auch jederzeit erneut gefolgert werden kann. Für den Ablauf des geplanten Hilfe-Tools für den Pathologen bedeutet diese Erweiterung der Ontologie, dass das Programm nun so implementiert werden kann, dass es automatisch einen Bericht als inhaltlich vollständig erkennt, sobald alle existierenden Instanzen der Klasse `:Question` gleichzeitig auch Instanzen der Klasse `:AnsweredQuestion` sind. Voraussetzung dafür ist, dass nach jeder Beantwortung einer Frage der Reasoner aktiv wird. Ohne Inferenz werden beantwortete Fragen nicht der Klasse `:AnsweredQuestion` zuordnet und das Programm kann den Bericht nicht als vollständig identifizieren.

Die Auswahl bestimmter Antwortmöglichkeiten hat Auswirkungen auf nachfolgend zu stellende Fragen. So ergibt sich, wie in 4.1 einleitend erwähnt, bei der Beantwortung der Frage nach der Lymphknoten-Präsenz mit „präsent" (present) die Frage nach deren Lage. Um diesen Mechanismus in der Ontologie abzubilden, wird der Grundaufbau des allgemeinen Befundberichts aus Listing 4.1 so erweitert, dass die Möglichkeit besteht, spezielle Fragen nur zu stellen, wenn im Vorfeld dazu passende Antworten gegeben wurden. Listing 4.3 zeigt die dazu modellierten Entitäten.

Listing 4.3: Ontologie-Erweiterung für die Aktivierung von Fragen

```
1  :hasSelectedOption rdfs:range :SelectedAnswer .
2
3  :EnabledQuestion a owl:Class .
4
5  :enablesCandidate a owl:ObjectProperty ;
6                    rdfs:domain f:Antwort ;
7                    rdfs:range f:Frage .
8
9  :SelectedAnswer a owl:Class ;
10         rdfs:subClassOf :Answer ,
11                         [ a owl:Restriction ;
12                           owl:onProperty :enablesCandidate ;
13                           owl:allValuesFrom :EnabledQuestion
14                         ] ;
15         rdfs:label "Ausgewaehlte Antwort"@de ,
16                    "Selected Answer"@en .
```

Es wird zunächst definiert, dass die Relation `:hasSelectedOption` als Range eine Klasse namens `:SelectedAnswer` verlangt (Z. 1). Wenn der Pathologe auf die Frage nach der Lymphknoten-Präsenz (`:indQuestion05`) mit präsent (`:indAnswer05.02`) antwortet

und folglich das Tripel `:indQuestion05 :hasSelectedOption :indAnswer05.02` in die Wissensbasis gespeichert wird, dann kann `:indAnswer05.02 a :SelectedAnswer` geschlussfolgert werden. Des Weiteren zeigt Listing 4.3 die Definitionen einer neuen Klasse namens `:EnabledQuestion` (Z. 3) und einer neuen Relation `:enablesCandidate` (Z. 5-7).

Unter genannten Voraussetzungen kann mit Hilfe einer allValuesFrom-Restriktion die anonyme Klasse beschrieben werden, welche alle Individuen zusammenfasst, deren Werte für die Relation `:enablesCandidate` alle von der Klasse `:EnabledQuestion` kommen müssen (Z. 9-16). Durch die *rdfs:subClassOf*-Beziehung in Z. 10 muss nun jedes Individuum der Klasse `:SelectedAnswer` zwingend auch diese Restriktion erfüllen. Im vorliegenden Beispiel bedeutet das für den Fall, dass die Antwort `:indAnswer05.02` zu einem Individuum der Klasse `:SelectedAnswer` wird, sie gleichzeitig auch ein Individuum der in Z. 12-15 beschriebenen Restriktion ist. Da die Wissensbasis außerdem das Tripel `:indAnswer05.02 enablesCandidate :indQuestion06` enthält (Listing A.1, Z. 47), kann vom Reasoner `:indQuestion06 a :EnabledQuestion` gefolgert werden. Unter der Voraussetzung, dass das Hilfe-Tool zur Erstellung eines Befundberichts nun ausschließlich Fragen der Klasse `:EnabledQuestion` an den Pathologen richtet und zu Beginn des Programmablaufs die Frage `:indQuestion06` nicht vom Typ der Klasse `:EnabledQuestion` ist, wird die Frage nach der Lage der Lymphknoten nur gestellt, wenn die vorher gestellte Frage nach der Lymphknoten-Präsenz mit präsent (`:indAnswer05.02`) beantwortet ist.

Damit dieser Ablauf realisiert werden kann, muss definiert sein, welche Fragen bereits zu Beginn der Erstellung eines neuen Befundberichts aktiviert sind. In Listing A.1 ist demonstriert, wie diese Zuordnung für die in Abb. 4.4 gezeigten Fragen realisiert ist (Z. 61-73). Ob die Beantwortung einer Frage als erforderlich oder empfohlen gilt, wird über die Beziehungen `:enablesCandidate` beziehungsweise `:enablesRequiredCandidate` modelliert. Einzige Ausnahme hierbei bildet die Frage nach dem spezifischen Report-Typ, welche in diesem Fall gleich zu Beginn der Berichterstellung mit Prostatakrebs-Report beantwortet wurde (Z.64 66). Durch die Zuordnung des Individuums `:prostateCancerReport` zur Klasse `:SelectedAnswer` wird durch zuvor beschriebenen Reasoning-Prozess geschlussfolgert, dass alle durch die Relation `:enablesCandidate` mit diesem Individuum in Verbindung stehenden Fragen gleichzeitig auch aktivierte Fragen sind. Aktivierte Fragen (`:EnabledQuestion`) sind grundsätzlich diejenigen, deren Beantwortung als empfohlen gilt (z. B. Frage 3 in Abb. 4.4, Z. 71 in Listing A.1). Aktivierte erforderliche Fragen (`:EnabledRequiredQuestion`) sind über die Relation `:enablesRequiredCandidate` mit dem Individuum des Report-Typs verknüpft (z. B. Fragen 1, 2, 4 und 5 in Abb. 4.4, Z. 72-73 in Listing A.1). Allgemein gilt, dass `:EnabledRequiredQuestion` eine Unterklasse von `:EnabledQuestion` ist. Die Unter-

scheidung der aktivierten Fragen in erforderliche und normale, nicht erforderliche (optionale) ist für den späteren Vollständigkeits-Check des Berichts notwendig (siehe Abschnitt 4.2.2 unter Funktion des Vollständigkeits-Checks).

Eine vollständige Repräsentation des modellierten Report-Grundgerüsts ist grafisch in Abb. 4.1 dargestellt. Für die spätere Implementierung der einzelnen Antwort-Elemente auf einer grafischen Benutzeroberfläche sind diese in Text-, Number-, Checkbox- und Radio-ButtonAnswer gegliedert. Die Datentyp-Relation `:hasAnswerText` ist nicht in Abb. 4.1 enthalten, da solche Beziehungen nicht zwischen zwei Klassen, sondern zwischen einer Klasse als Subjekt und einem konkreten Literal als Objekt bestehen. Das Subjekt ist in diesem Fall die Klasse `:Answer`.

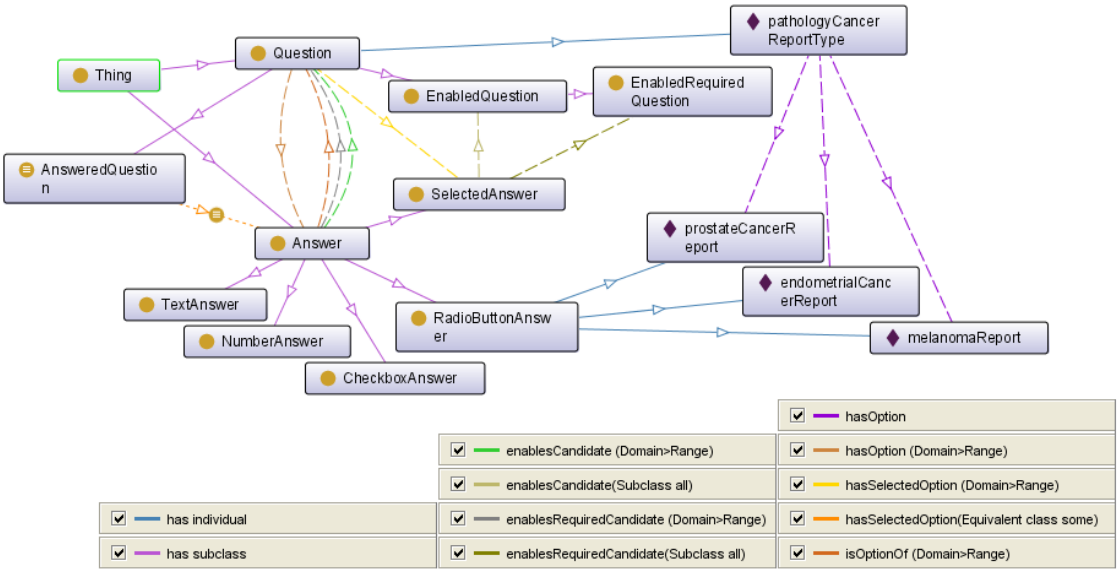

Abbildung 4.1: Grundgerüst einer Report-Ontologie (APRBasis.owl)

Aus der genannten Modellierung ergibt sich für die Ontologie der Report-Basis die in Abb. 4.2 dargestellte Metrik.

Metrics

Axiom	84
Logical axiom count	33
Class count	10
Object property count	5
Data property count	1
Individual count	4
DL expressivity	ALCHI(D)

Abbildung 4.2: Metrik der Report-Basis-Ontologie

Die Ontologie enthält insgesamt 84 Axiome, von denen 33 sogenannte AnnotationAssertion-Axiome sind. Diese beschreiben Verknüpfungen von Entitäten zu einer bestimmten Annotation. Im vorliegenden Fall handelt es sich um die Relation `rdfs:label`, durch welche die Klassen, Relationen und Individuen mit einem Label verbunden sind. Des Weiteren enthält die Wissensbasis Beschreibungen von zehn Klassen, vier Individuen sowie fünf Objekt- und einer Datentyp-Relation. Die verbleibenden Axiome verteilen sich beispielsweise auf SubClass- und SubProperty-Axiome sowie auf Beschreibungen von äquivalenten und disjunkten Klassen, inversen Relationen und den Klassen-Zuweisungen der drei Individuen zur Modellierung der Report-Typen zu den entsprechenden über die Relation `rdf:type`. Durch die verwendeten Axiome erhält die Ontologie eine semantische Aussagekraft der Beschreibungslogik $\mathcal{ALCHI}(\mathcal{D})$. Tabelle 4.1 zeigt die Bedeutung der einzelnen Symbole bezüglich der verwendeten Axiome.

Tabelle 4.1: Beschreibungslogik $\mathcal{ALCHI}(\mathcal{D})$

Symbol	Bedeutung	
$\mathcal{ALC}$	Attributive Language with Complements:	
	Basis-Beschreibungslogik, welche folgende atomare Typen	
	und Konstruktoren umfasst:	
	Atomares Konzept: A	(owl:Class)
	Universelles Konzept (Top): $\top$	(owl:Thing)
	Leeres Konzept (Bottom): $\bot$	(owl:Nothing)
	Rolle (Relation): R	(rdf:Property)
	Komplement (Negation): $\neg C$	(owl:complementOf)
	Schnitt (Konjunktion): $C \sqcap D$	(owl:intersectionOf)
	Wert-Restriktion: $\forall R.C$	(owl:allValuesFrom)
	Beschränkte Existenz-Restriktion: $\exists R.\top$	(owl:someValuesFrom)
$\mathcal{H}$	Relationshierarchie	(rdfs:subPropertyOf)
$\mathcal{I}$	Inverse Relationen	(owl:inverseOf)
$(\mathcal{D})$	Datentyp-Relationen	(owl:DatatypeProperty)
	Datentypen	(z. B. xsd:string)

Semantisch entsprechen die Konzeptbeschreibungen den in Abb. 4.3 formal dargestellten Interpretationen.

$$
\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\
\bot^{\mathcal{I}} &= \emptyset \\
(\neg A)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus A^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(\forall R.C)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} | \forall b : (a,b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\} \\
(\exists R.\top)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} | \exists b : (a,b) \in R^{\mathcal{I}}\}
\end{aligned}
$$

Abbildung 4.3: Semantik von Konzeptbeschreibungen

4.1.1.2 Datenherkunft für die Vorlagen der drei Report-Typen

Das Expertenwissen darüber, welche Inhalte in einem pathologischen Befundbericht empfohlen oder gar erforderlich sind, wird aus den Datensätzen der ICCR (International Collaboration on Cancer Reporting) entnommen (Unterabschnitt 2.2.2: Inhaltliche Vorlagen für pathologische Befundberichte). Speziell werden hierbei die Spezifikationen im Excel-Dateiformat (*.xlsx) für die Berichte der Krebsarten Endometrium-, Haut- und Prostatakrebs genutzt. Im Zuge des Ontology Learning werden die Konzepte der laut ICCR-Vorlage empfohlenen oder erforderlichen Angaben als Klassen (Ontology Enrichment) und Individuen (Ontology Population) in maschinenlesbarer Form modelliert. Mit der länderübergreifenden Zusammenarbeit beim Erstellen dieser Vorlagen geht die Initiative einen großen Schritt in Richtung globaler Standardisierung von pathologischen Berichten. Das erhöht nicht nur die Sicherheit einer korrekten Wissensgrundlage sondern steigert sogleich die globale Akzeptanz der definierten Vorlagen.

Die ICCR-Datensätze stehen online in den Dateiformaten PDF, DOC und XLSX zur Verfügung. Letztere Variante enthält die wichtigsten Informationen zur Befundberichtvorlage in Tabellenform (Microsoft Excel-Dateiformat). Abb. 4.4 zeigt einen Ausschnitt aus der Excel-Datei, welche die Spezifikation eines Prostatektomie-Berichts beschreibt.

	A	B	C	D	E
1	**Required/ Recommended**	**Element name**	**Values**	**Commentary**	**Implementation notes**
2	Required	Pre-biopsy serum PSA	Numeric: ___ ng/mL OR Not available	Pre-biopsy serum PSA is essential for stage grouping in the 7th Edition of the [...]	
3	Required	Specimen weight (ie Prostate without seminal vesicles)	Numeric: ____g		
4	Recommended	Specimen dimensions (prostate)	Numeric: __x__x__mm		
5	Required	Seminal vesicles	Single selection value list: • Absent • Present (partially or completely resected)		
6	Required	Lymph nodes	Single selection value list: • Absent • Present		
7	Recommended	Laterality	Single selection value list: • Left • Right • Bilateral		Record only if Lymph nodes is present.

Abbildung 4.4: Auszug aus dem ICCR-Datensatz eines Prostatektomie-Berichts

Die erste Spalte der Tabelle gibt Auskunft darüber, ob das in Spalte B benannte Element in einem solchen Bericht erforderlich (required) oder optional empfohlen (recommended) ist.

Mit Ausnahme der Spaltenüberschriften in Zeile 1 spiegelt jede folgende Zeile ein Element des Reports wider und ist als Frage in Bezug auf das zu beschreibende Untersuchungsergebnis zu verstehen.

Spalte C enthält Informationen über die Struktur der Antwort. Ist ein frei formulierbarer Text zulässig, steht hier lediglich „Text". Wird jedoch eine Auswahl an Antwortmöglichkeiten vorgegeben, aus denen eine oder mehrere ausgewählt werden können, ist dies durch andere Schlagworte zu identifizieren. Es gibt vier verschiedene Arten von Antwortelementen:

- Auswahlliste, bei der ausschließlich ein einziges Element selektiert werden kann (Optionsfelder, Radiobuttons oder auch ToggleGroup)
- Auswahlliste, bei der mehrere Elemente selektiert werden können (Kontrollkästchen, Checkboxes)
- Einzeiliges Eingabefeld für kurze Angaben in Textform (Textfeld, Textfield)
- Mehrzeiliges Eingabefeld für Fließtext (Textbereich, Textarea)

Bezüglich der vom Benutzer möglichen Angaben sind die Antwortelemente dieser Liste in absteigender Reihenfolge mit immer weniger Restriktionen belegt. Während in einem Optionsfeld lediglich eine der vorgegebenen Optionen gewählt werden kann, können in Textbereichen frei formulierte Fließtexte stehen. In Hinblick auf die beabsichtigte Standardisierung von pathologischen Befundberichten sind Antwortelemente des oberen Listenbereichs besser für eine einheitliche Modellierung von Berichten geeignet als darunterliegende Elemente.

Eine Einleitung in Spalte C mit dem String „Single selection value list" lässt erkennen, dass es sich um eine Frage handelt, zu deren Beantwortung eine der vorgegebenen Antwortmöglichkeiten auszuwählen ist. Die einzelnen Antworten werden anschließend innerhalb der Tabellenzelle jeweils durch einen Punkt (Unicode-Zeichen: U+2022) eingeleitet. Diese Antwortstruktur erfordert im späteren Implementierungsprozess das Erstellen einer Gruppe von Radiobutton (ToggleGroup, Optionsfeld).

Wenn zur Fragenbeantwortung mehrere Auswahlmöglichkeiten gewählt werden können, wird dies durch den Ausdruck „Multi selection value list" innerhalb der entsprechenden Tabellenzelle gekennzeichnet und die daraus resultierende Antwortstruktur gleicht einer Gruppe von Kontrollkästchen (Checkboxes), welche unabhängig voneinander ausgewählt werden können. Zur Beantwortung der Frage sind in diesem Fall mindestens eine, bei Bedarf aber auch mehrere Antwortmöglichkeiten auszuwählen.

Neben der Antwort als Fließtext, welche durch das Wort „Text" erkannt wird, deutet die Einleitung „Numeric" auf einen Zahlenwert als Antwort hin. In letzterem Fall wird im Verlauf

der softwaregesteuerten Berichtserstellung ein Textfeld erstellt, welches durch Zahlen zu füllen ist.

Zeile D enthält Kommentare zur jeweiligen Frage. Diese sind an den Pathologen gerichtet und dienen als Hintergrundinformationen zum Verständnis und zur korrekten Beantwortung der Frage. Sie werden im zu erstellenden Softwaretool bei der Betrachtung der jeweiligen Frage in einem gesonderten Textbereich angezeigt.

Die Notizen zur Implementierung aus der fünften Spalte E geben Auskunft über die Voraussetzungen, die erfüllt sein müssen, damit die betreffende Frage im Bericht angezeigt wird. So enthält diese Spalte beispielsweise für die Frage nach der Lage der entnommenen Lymphknoten (*Laterality*) die Notiz, dass diese nur an den Pathologen gerichtet wird, wenn die Frage nach dem Vorhandensein von Lymphknoten (*Lymph nodes*) mit ja (*Present*) beantwortet wurde. Diese Art der Abhängigkeiten von einzelnen Fragen voneinander gilt es ebenfalls, durch geeignete Relationen innerhalb der zu erstellenden Ontologie abzubilden.

Damit der Ontology-Learning-Prozess fehlerfrei ablaufen kann, müssen die ICCR-Datensätze zunächst in Microsoft Excel betrachtet und gegebenenfalls angepasst werden. Dazu gehört die Kontrolle der Implementierungsnotizen in Spalte E (Abb. 4.4) bezüglich der korrekten Syntax. Hier ist es wichtig, dass Restriktionen nach dem Muster [„Record only if ” + *Titel der aktivierenden Frage* + „ is ” + *Titel der entsprechenden Antwortmöglichkeit*] formuliert sind. Andernfalls werden diese während des Ontology Learning nicht korrekt modelliert. Ein Beispiel für eine solche Restriktion ist in Abb. 4.4, Zeile 7, Spalte 5 zu sehen. Die Frage nach der Lage der entnommenen Lymphknoten (*Laterality*) ist nur zu stellen, wenn die Frage nach dem Vorhandensein von Lymphknoten (*Lymph nodes*, Zeile 6) mit ja (*present*) beantwortet wurde. Für den Fall, dass die Beantwortung einer Frage unabhängig von der gegebenen Antwort zur Aktivierung einer weiteren Frage führen soll - wie es zum Beispiel bei Textfeldern der Fall ist - kann anstelle der Antwortmöglichkeit der String „given” in Spalte E stehen, sodass sich das Muster [„Record only if ” + *Titel der aktivierenden Frage* + „ is given”] ergibt.

Des Weiteren wird die Darstellung der TNM-Klassifikation in den Excel-Tabellen nachträglich als Kombination aus Optionsfeldern beschrieben. Die originalen ICCR-Vorlagen verweisen an dieser Stelle lediglich auf die entsprechenden Quellen, während in den PDF-Varianten der Report-Vorlage noch die jeweils spezifische TNM-Klassifikation tabellarisch aufgeführt ist. Anhand dieser Dokumente wird die Spalte C zu den entsprechenden Zeilen der Excel-Tabellen ergänzt.

Eine Besonderheit für die Beschreibung von Antwortmöglichkeiten stellt zum Beispiel die

Frage nach dem PSA-Wert vor der Biopsie dar. Wie anhand der zweiten Spalte zu dieser Frage in Abb. 4.4 zu erkennen ist, kann als Antwort ein konkreter Zahlenwert stehen oder die Option „Not available" ausgewählt werden. Damit dies während des Ontology-Learning-Prozesses korrekt modelliert werden kann, ist hier eine Anpassung notwendig. Der entsprechende Zelleninhalt wird als „Single selection value list" umformuliert und die Antwortmöglichkeit „Numeric: _____ ng/mL" erhält den Suffix „(specify)". Durch diesen wird während des Ontology Learning erkannt, dass bei Auswahl der entsprechenden Option ein konkreter Wert als Antwort verlangt ist. Dies bedingt die Modellierung der entsprechenden Datentyp-Relation (`:hasAnswerText`) in der Wissensbasis für diese Antwortmöglichkeit.

4.1.2 Schnittstellen

Im Folgenden werden die grundlegenden verwendeten Schnittstellen der geplanten Software erläutert. Dabei wird darauf geachtet, dass alle zu verwendenden Komponenten mit derselben Programmiersprache angesprochen werden können. Aufgrund der weiten Verbreitung, Plattformunabhängigkeit und der Verfügbarkeit einer Vielzahl von Bibliotheken wurde sich für die objektorientierte Programmierung in Java entschieden.

POI API für den Excel-Datenimport

Für den Import der Excel-Dateien in die Umgebung des Java-Programms wird die Programmierschnittstelle (englisch: application programming interface, abgekürzt: API) POI[1] von Apache eingesetzt, welche das Einlesen von Microsoft-Dokumenten wie Word- und Excel-Dateien erlaubt.

OWL API zum Erstellen, bearbeiten und speichern der Report-Ontologie

Um Ontologien in ihrer formalen Beschreibungssprache OWL innerhalb eines Java-Programms erstellen und bearbeiten zu können, ist eine weitere Programmierschnittstelle notwendig. Hierzu wird die OWL API von Horridge und Bechhofer [HB11] genutzt, welche unter LGPL (GNU Lesser General Public License) lizenziert kostenlos als Open-Source-Software zur Verfügung steht und in Java über die Erstellung eines Maven-Projekts mit entsprechenden Anhängigkeiten (englisch: dependencies) eingebunden werden kann[2].

Die OWL API ist mit den aktuellen Strukturvorgabe der W3C-Spezifikation OWL 2 abgestimmt und unterstützt sowohl Parsing (Syntaxanalyse) als auch Rendering (Darstellung,

[1] Apache POI: https://poi.apache.org/
[2] OWL API als Download oder als Maven dependency: https://github.com/owlcs/owlapi/wiki/Documentation

Übersetzung) für die dort definierten Syntax-Varianten (RDF/XML, OWL/XML, Turtle, Functional Syntax und Manchester Syntax). Neben der Manipulation von existierenden ontologischen Strukturen, ist mit Hilfe der API auch die Erstellung einer komplett neuen Ontologie im Java-Umfeld möglich. Die Schnittstelle selbst ist dabei ebenfalls in Java implementiert und beinhaltet Validatoren für die in OWL 2 definierten Profile (OWL 2 QL, OWL 2 EL und OWL 2 RL). [HB11]

Durch die Integration verschiedener Reasoner lässt sich über die OWL API auch der Reasoning-Prozess in der Sprache Java steuern. Damit ermöglicht diese Schnittstelle die inferenzgesteuerte Erweiterung einer zuvor in Protégé modellierten Ontologie auch von außerhalb dieses Entwicklungstools.

OWL-Reasoner HermiT

Das Softwaretool, welches innerhalb einer Ontologie aus gegebenen Informationen Rückschlüsse mittels Inferenz (englisch: Reasoning) zieht, wird allgemein als Reasoner bezeichnet. Die zuvor beschriebene OWL API von Horridge und Bechhofer [HB11] stellt Schnittstellen für das Arbeiten mit den Reasonern FaCT++, HermiT, Pellet und Racer bereit. Die Reasoner FaCT++ und HermiT sind ebenfalls in der verwendeten Protégé-Version 4.3.0 auswählbar, weshalb sich im weiteren Verlauf für letzteren entschieden wurde.

HermiT[3] ist ein auf dem Hypertableau-Algorithmus basierendes Schlussfolgerungstool für OWL-2-Ontologien, welches alle Features von OWL 2 DL unterstützt. Entwickelt wurde der Reasoner initial an der University of Oxford und später in Zusammenarbeit mit der Universität Ulm. Das Tools ist kostenlos als Open-Source-Projekt unter der LGPL-Lizenz verfügbar und läuft auf allen Betriebssystemen, die Java 1.5 oder höher unterstützen. Neben den Möglichkeiten des Aufrufs mittels OWL API oder innerhalb von Protégé, lässt sich HermiT auch über die Kommandozeile benutzen. [GHM+14]

4.1.3 Output

Die im Zuge des Programmverlaufs erstellte und modifizierte Ontologie kann jederzeit als OWL-Datei in der Turtle-Syntax gespeichert werden. Die Datei enthält je nach Fortschritt der Berichterstellung das in Abschnitt 4.1.1.1 beschriebene Grundgerüst mit den Erweiterungen des jeweiligen Report-Typs durch die Prozesse des Ontology Enrichment und Ontology Population. Darüber hinaus sind alle bis zum Zeitpunkt des Speicherns gegebenen Antworten

[3] HermiT 1.3.8: http://www.hermit-reasoner.com

nach Vorgaben der Struktur des Grundgerüsts in der OWL-Datei modelliert. Das bedeutet, dass alle entsprechenden Objekt-Relationen zwischen Fragen und gegebenen Antworten sowie die Datentyp-Relationen zwischen Antworten und jeweiligem Literal in der OWL-Datei enthalten sind.

4.2 Implementierung

Für die geplante dynamische Erstellung von Befundberichten nach in Abschnitt 4.1 einleitend beschriebener Arbeitsweise und deren Modellierung gemäß des in 4.1.1.1 erläuterten Grundaufbaus, sind die nachfolgend beschriebenen Programmbestandteile realisiert. Die Software ist in der Programmiersprache Java, Version 8, Update 60, unter Verwendung der Entwicklungsumgebung Eclipse, Version 4.6 (Neon), implementiert worden. Die Funktionalität der Software wird unter 4.2.2 erläutert und enthält eine Übersicht darüber, welche Benutzeroberflächen-Elemente für welchen Aufgabenbereich zur Verfügung stehen.

4.2.1 Aufbau des Programms

Abb. A.2 zeigt das Klassendiagramm der entwickelten Software und dient im Folgenden als Leitfaden für die Beschreibung des Programmaufbaus. Die neun dargestellten Klassen gliedern das Tool je nach Aufgabenbereich, wobei sich inklusive der Main-Klasse sieben Aufgabenpakete abstecken lassen.

Main

Die Main-Klasse stellt den Einstiegspunkt beim Start des Programms dar. Hier wird die grafische Benutzeroberfläche (englisch: Graphical user interface, abgekürzt: GUI) geladen, welche die plattformübergreifende Java-Applikation unter Einsatz des JavaFX-Frameworks visualisiert (Abb. 4.5).

MainWindowController

Diese Klasse dient der Steuerung der Benutzeroberfläche und enthält die dazu notwendigen Methoden. Die Elemente der GUI, die von hier aus angesprochen werden können, sind als private Variablen festgelegt, welche jeweils die in der FXML-Datei definierte grafische Repräsentation des entsprechenden Elements referenzieren. Die Variable btStart verweist

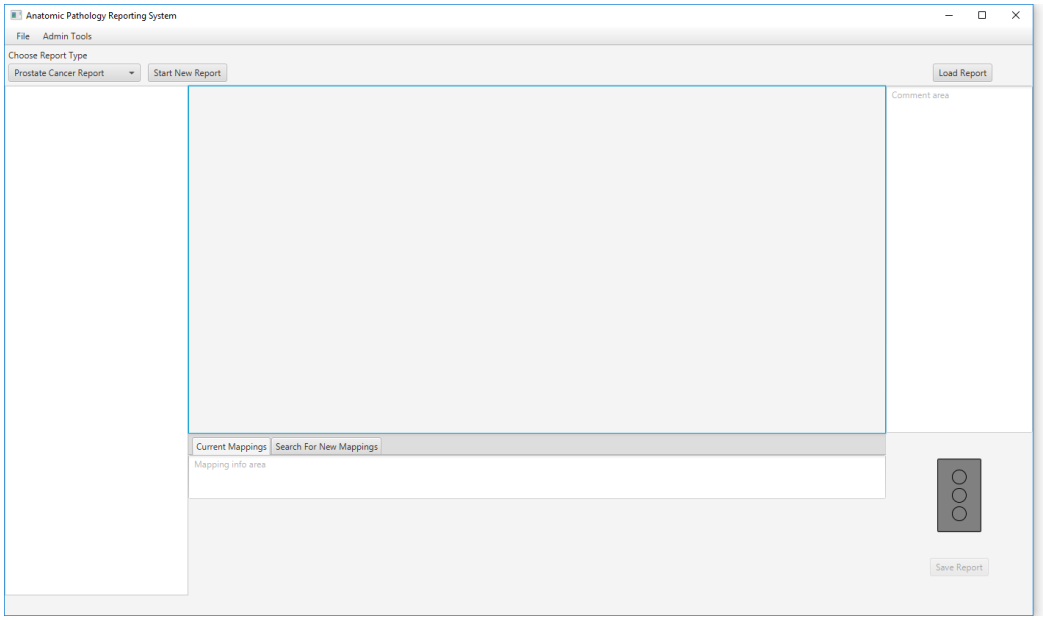

Abbildung 4.5: Benutzeroberfläche bei Programmstart

beispielsweise auf die Schaltfläche zum Erstellen eines neuen Befundberichts. In der FXML-Datei lässt sich ebenfalls festlegen, welche Funktion beim Klick auf diese Schaltfläche aufgerufen werden soll. Hier ist die Methode startNewReport() hinterlegt, welche im Abschnitt 4.2.2 beschrieben ist. Die innere Klasse QuestionNodeComparator ermöglicht die Sortierung der dargestellten Fragen in der Liste auf der rechten Seite und im zentralen Bereich der Benutzeroberfläche (vgl. Abb. A.4).

Ontology

Diese Klasse repräsentiert das Konzept eines pathologischen Reports (APR) im OWL-Format. Folglich entspricht jeder mit dieser Software erstellte Befundbericht einer Instanz dieser Klasse. Der Standard-Konstruktor wird bei jedem Programmstart ausgeführt und modelliert das Grundgerüst eines solchen Reports, welches im Gliederungspunkt 4.1.1.1 unter dem Namen *APRBasis.owl* beschrieben ist. Zum Sortieren von Instanzen der Klasse Frage in programminternen Listen ist die innere Klasse QuestionComparator implementiert.

Question

Das Konzept einer konkreten Frage innerhalb eines Befundberichts, welches in der APR-Basis-Ontologie durch ein Individuum der Klasse `:Question` (vgl. 4.1.1.1 Basis der Report-Ontologie) modelliert ist, wird im Programm durch die gleichnamige Klasse Question

beschrieben. Im Kontext der Softwareprogrammierung ist diese als innere Klasse der Ontology-Klasse definiert und implementiert verschiedene Methoden zu ihrer Repräsentation. Die Funktion getQuestionLabel(String language) liefert beispielsweise das innerhalb der Ontologie durch `rdfs:label` festgeschriebene Label des Individuums in der gewünschten Sprache (String language="en"). getIndQuestion() hingegen gibt die Repräsentation dieser Frage als Instanz der Java-Klasse OWLNamedIndividual zurück, welche dem über die OWL API von Horridge und Bechhofer [HB11] referenzierten Individuum innerhalb der Report-Ontologie entspricht.

Answer

Analog zur Java-Klasse Question entspricht die Klasse Answer der Repräsentation einer Antwort, die innerhalb der APR-Basis-Ontologie durch ein Individuum der Klasse `:Answer` (vgl. 4.1.1.1 Basis der Report-Ontologie) modelliert ist. Auch diese ist innerhalb der Java-Klasse Ontology implementiert und stellt Methoden zum Erstellen, Identifizieren und Bearbeiten eines entsprechenden Vertreters auf der Benutzeroberfläche bereit.

ExcelDataImport

Diese Klasse realisiert unter Verwendung der POI API von Apache den Import der Daten aus einer Excel-Datei (vgl. Abschnitt „POI API" in 4.1.2 Schnittstellen). Im Programmkontext werden über diese Klasse im Zuge des Ontology-Learning-Prozesses die Informationen aus der ICCR-Vorlage geladen, welche anschließend zur Erstellung der spezifischen Report-Typ-Ontologie genutzt werden (vgl. Methode enrichAndPopulateOntology() in 4.2.2 Programmfunktionen). Bei der Erstellung der Wissensbasis für einen Prostatakrebsbericht wird hier beispielsweise der Inhalt der Excel-Datei „ICCRProstateCancerDataset.xlsx" importiert.

SPARQLQuery

Die Suche nach Mapping-Vorschlägen von Entitäten aus der Report-Ontologie zu Entitäten externer Wissensbasen erfordert eine entsprechend definierte Abfrage. RDF-basierte Wissensbasen lassen sich über die SQL-ähnliche Abfragesprache SPARQL durchsuchen. Die Klasse SPARQLQuery realisiert unter Verwendung der Bibliothek Jena ARQ[4] eine solche Anfrage an einen SPARQL-Endpoint. Voraussetzung für das Erhalten von Mapping-Vorschlägen ist, dass die zu befragenden Wissensbasis einen SPARQL-Endpoint bereitstellt. Die Website

[4] ARQ - SPARQL-Anfrage-Engine des Jena Frameworks für semantische Netze: https://jena.apache.org/documentation/query/

BioPortal liefert einen solchen Zugang[5] zu allen hier veröffentlichten Wissensbasen, darunter auch SNOMED CT[6], NCIT[7] (National Cancer Institute Thesaurus) und PathLex[8]. DBpedia kann über den Virtuoso SPARQL-Endpoint[9] befragt werden.

4.2.2 Funktionalität und Beschreibung der dazugehörigen Methoden

Dieser Abschnitt enthält die Beschreibung der wichtigsten Funktionen aus den in Abb. A.2 dargestellten Klassen. Eine Übersicht darüber, welche Methode zur Bearbeitung welcher Aufgabe aufgerufen wird, ist in Tabelle 4.2 ersichtlich. Dabei handelt es sich um Funktionen, die aus der Benutzeroberfläche heraus aufgerufen werden können und demzufolge zunächst ihren Einstiegspunkt in der Klasse MainWindowController haben (vgl. Abb. 4.5). Anschließend werden diese Methoden näher erläutert. Um den Anhang dieser Arbeit nicht zu groß werden zu lassen, wurde darauf verzichtet, jede einzelne Funktion als Listing in dieses Dokument zu integrieren. Die programmatische Umsetzung der beschriebenen Funktionen kann sowohl im Quellcode der Software als auch in der dazugehörigen Dokumentation (Javadoc) auf der beiliegenden CD eingesehen werden.

MainWindowController:initialize()

Diese Methode wird automatisch beim Laden der Klasse MainWindowController aus der Main-Klasse heraus aufgerufen und wird somit bei jedem Programmstart automatisch ausgeführt. Durch das Erzeugen einer Instanz der Klasse Ontology durch den Aufruf des Standard-Konstruktors Ontology() wird das Grundgerüst der Report-Ontologie modelliert und innerhalb der Klasse MainWindowController an eine Variable vom Typ Ontology gebunden. Somit ist die erstellte Instanz im weiteren Programmverlauf über diese Variable ansprechbar. Des Weiteren wird das Auswahlfeld für den Report-Typ initialisiert, indem über die Funktion getReportTypes() aus der Java-Ontology-Klasse eine entsprechende Liste der vorhandenen Individuen aus der eingangs modellierten Report-Basis-Ontology verlangt wird. Standardmäßig sind hier die drei Report-Typen zur Befundung von Endometrium-, Haut- und Prostatakrebs modelliert und programmintern als OWL-Dateien verfügbar.

[5]SPARQL-Endpoint von BioPortal: http://sparql.bioontology.org/
[6]SNOMED CT auf BioPortal: http://bioportal.bioontology.org/ontologies/SNOMEDCT
[7]NCIT auf BioPortal: http://bioportal.bioontology.org/ontologies/NCIT
[8]PathLex auf BioPortal: http://bioportal.bioontology.org/ontologies/PATHLEX
[9]SPARQL-Endpoint von DBpedia: http://dbpedia.org/sparql

Tabelle 4.2: Funktionalität der Software

Aufgabenbereich	Trigger-Element auf der Benutzeroberfläche	Zuständige Methode als Einstieg innerhalb der Software
Modellieren des Grundgerüsts der Report-Ontologie	Bei Programmstart	initialize()
Erstellen einer neuen Report-Typ-Vorlage	Menüpunkt *Create New Report Type* unter *Admin Tools*	createNewReportType()
Erstellen eines neuen Reports unter Laden einer vorhandenen Report-Typ-Vorlage	Schaltfläche *Start New Report* oder Menüpunkt *New Report* unter *File*	startNewReport()
Laden eines Reports	Schaltfläche oder Menüpunkt *Load Report*	loadReport()
Anzeigen aktivierter Fragen und dazugehöriger Antwortmöglichkeiten	Bei Programmstart und bei Auswahl einer Frage oder einer Antwort	updateEnabledQuestions(Pane questions, Ontology ont)
Überprüfen des Reports auf Vollständigkeit	Bei Auswahl einer Antwort	updateTrafficLights(Ontology ont)
Anzeigen vorhandener Mappings	Bei Auswahl einer Frage oder einer Antwort, erscheint unter Tab *Current Mappings*	updateCurrentMappings(Object obj)
Suchen nach Mapping-vorschlägen und Bearbeiten vorhandener Mappings	Schaltfläche *Search Mappings* unter Tab Search For New Mappings	searchMappings()
Speichern eines Reports	Schaltfläche oder Menüpunkt *Save Report*	saveReport()

createNewReportType(): Ontology Enrichment & Ontology Population

Die Methode zum Erstellen der Ontologie für einen Report-Typ beinhaltet den Aufruf der Funktion enrichAndPopulateOntology(String strReportType, File excelFilePath), welche auszugsweise in Listing A.2 dargestellt ist. Die Funktion ist in der Klasse Ontology implementiert und bildet den Kern des Ontology-Learning-Prozesses unter Verwendung der Daten aus den spezifischen ICCR-Vorlagen. Es werden der zu modellierende Report-Typ als String aus dem entsprechenden Auswahlfeld (vgl. Abb. 4.5) und die Excel-Datei als Input für diese Prozedur erwartet. Bei Fehlern während des späteren Einlesens der Datei wird eine Fehlermeldung erzeugt.

Zu Beginn werden die hierbei benötigten OWL-Klassen und Objekt-Relationen an Variablen gebunden. (Listing A.2, Z. 8-18). Anschließend erfolgt der Datenimport aus der gewählten Excel-Datei durch die Instanziierung der Klasse ExcelDataImport und den Aufruf der Import-Funktion (Z. 21-22), welcher die entsprechende Excel-Datei übergeben wird.

Für jede aus der Excel-Datei gelesene Zeile - mit Ausnahme der Spaltenüberschriften - wird im Folgenden eine Frage modelliert. Grundsätzlich werden diese Fragen als Individuen innerhalb des Programms betrachtet. Für das spätere Mapping ist es jedoch notwendig, zu jeder Frage auch eine entsprechende Klassen zu modellieren, da es andernfalls beim Mapping von Individuen zu Konflikten mit der Spezifikation der verwendeten Mapping-Relation `skos:closeMatch` kommt. Da beim Erstellen von Klassen die T-Box der Wissensbasis modifiziert wird, handelt es sich bei diesem Lernprozess um ein Ontology Enrichment (vgl. Abschnitt 3.2 Ontology Learning). Das Modellieren der Fragen als Individuen entspricht in seiner Gesamtheit laut Definition in Unterabschnitt 3.2.3 einem Ontology-Population-Verfahren.

Die Prozedur createQuestionAsNamedIndividual() zum Erzeugen eines Individuums steht beispielhaft für die generelle Modellierung einer Entität mit Hilfe der OWL API und ist in Z. 79-93 dargestellt. Entscheidend sind hier die Zeilen 87, 89 und 91, in denen das Individuum zuerst benannt, dann in eine Klassenzuweisung zur Klasse `:Question` und letztendlich als Teil dieser Klassenzuweisung über den Aufruf addAxiom() in die Wissensbasis geschrieben wird. Durch das Hinzufügen des Axioms der Klassenzuweisung wird das Individuum der entsprechenden Frage automatisch als `owl:NamedIndividual` modelliert.

Nachdem die Frage als Klasse und als Individuum erzeugt wurde, bekommen beide Repräsentationsformen sowohl das aus Spalte B der Excel-Datei ausgelesene Label (`rdfs:label`) als auch den aus Spalte D importierten Kommentar (`rdfs:comment`) zugewiesen. Die Zuweisung eines Labels ist durch die Funktion createLabel() (Z. 94-106) realisiert. In Z. 99 wird die vordefinierte Annotationsrelation über RDFS_LABEL.getIRI() in eine lokale Variable gespeichert. Mit der in Z. 101 definierten Annotation als Objekt wird das Axiom der Annotations-Zuweisung erstellt und der Ontologie hinzugefügt (Z. 103-105). Die Zuordnung eines Kommentars erfolgt insofern ähnlich, dass sich nur die Annotationsrelation ändert.

In Z. 47-61 ist die Zuordnung der Frage als Objekt der entsprechenden Objekt-Relation `:enablesCandidate` oder `:enablesRequiredCandidate` realisiert. Diese ist abhängig von der aus Spalte A des Excel-Datei importierten Festlegung, ob es sich hierbei um eine erforderliche (required) oder empfohlene (recommended) Fragestellung handelt. Subjekt dieser Relation ist stets der zu modellierende Report-Typ.

Die Funktion createAnswers() (Z. 64) modelliert je nach Inhalt der importierten Spalte C die Antwortmöglichkeit(en) zur vorliegenden Frage. Je nach Spalteninhalt werden Individuen der Klassen Text-, Number-, Checkbox- oder RadioButtonAnswer erzeugt.

Im letzten Schritt wird die ICCR-Spezifikation aus der Spalte E modelliert. Stimmt diese mit dem in Abschnitt 4.1.1.2 beschriebenen Muster überein, erfolgt eine automatische Zuweisung der Ausgangsfrage zur aktivierenden Frage über die Relation `:enablesCandidate`.

startNewReport() und loadReport(): Import von OWL-Dateien

Über den Aufruf des Konstruktors Ontology(InputStream ontologyInputStream) wird die vom Benutzer im Report-Typ-Auswahlfeld selektierte Ontology über einen Input-Stream eingelesen. Ist die entsprechende Vorlage geladen, kann mit der Erstellung eines neuen Befundberichts begonnen werden. Da sich sowohl beim Laden einer Report-Vorlage als auch beim Laden eines konkreten Berichts dieselben Aufgaben zum Import der Daten aus der OWL-Datei und zur Visualisierung der eingelesenen Entitäten ergeben, sind diese Funktionen sehr ähnlich aufgebaut. Einer der Unterschiede ist hier, dass bei Neuerstellung eines Reports die selektierte Report-Typ programmintern geladen wird, während beim Laden eines zuvor gespeicherten Befundberichts der Benutzer die einzulesende OWL-Datei angeben muss.

updateTrafficLights(Ontology ont): Vollständigkeits-Check

Diese in Listing A.3 gezeigte Funktion prüft über Aufrufe von Methoden aus der Klasse Ontology mit Hilfe des HermiT-Reasoners, ob die vorhandenen Axiome innerhalb der mit Methodenaufruf übergebenen Wissensbasis einen „ausreichenden" oder gar „vollständigen" Bericht beschreiben.

Zu Beginn werden über die Funktionsaufrufe in Z. 9-11 die aktuell in der Report-Wissensbasis vorhandenen Individuen der Fragen-Klassen `:AnsweredQuestion`, `:EnabledQuestion` sowie `:EnabledRequiredQuestion` in separate Listen gespeichert. Jeder dieser Funktionsaufrufe verwendet für die eigentliche Abfrage der Ontologie die Funktion getInstancesOf Class(OWLClass cls), wobei sich lediglich der übergebene Input-Parameter ändert.

Die in der Ontology-Klasse implementierte Methode ist in Listing A.4 ersichtlich. Die Funktion reasoner.getInstances() in Z. 7 ist für den Aufruf des Reasoning-Prozesses verantwortlich. Dieser Prozess sorgt dafür, dass für nachfolgende Abfrage nicht nur alle derzeit explizit modellierten Axiome zur Verfügung stehen, sondern auch alle daraus ableitbaren. Im darauffolgenden Schritt werden die gefundenen Instanzen der gesuchten Klasse in eine Liste geschrieben, welche der Funktion updateTrafficLights() Rückgabewert übergeben wird.

Auf Grundlage der Informationen aus dem Schlussfolgerungsprozess, werden nun die drei Listen durchlaufen und so jeweils miteinander verglichen. In Listing A.3, Z. 15-24 wird die Liste der aktivierten Frage betrachtet. Jedes Listenelement wird einer temporären Liste lstOpenQuestions hinzugefügt und im direkten Anschluss nur wieder entfernt, wenn die entsprechende Frage auch in der Liste der beantworteten Fragen steht. Ist die temporäre Liste lstOpenQuestions dann wieder leer, bedeutet das, das alle aktivierten Fragen gleichzeitig auch beantwortete Fragen sind. Daraus folgt, dass der Bericht vollständig ist. Dem Benutzer wird dies links unten auf der Benutzeroberfläche durch die Umschaltung der Ampel auf Grün signalisiert. (Z. 25-27). Wenn die temporäre Liste allerdings noch aktivierte Fragen enthält, dann wird nach demselben Muster die Liste der aktivierten erforderlichen Fragen durchlaufen (Z. 29-38). Ist diese am Ende der For-Schleife leer, sind zumindest keine erforderlichen Fragen mehr unbeantwortet. Die Ampel schaltet demzufolge auf Gelb und der Bericht gilt als vollständig, da es sich bei den offenen Fragen lediglich um empfohlene Angaben handelt. Sind nach Durchlauf der For-Schleife noch Fragen in lstOpenRequiredQuestions vorhanden, gilt der Bericht als inhaltlich unvollständig und die Ampel schaltet beziehungsweise bleibt auf Rot.

searchMappings(): Suchen nach Mapping-Vorschlägen

Die Prozedur searchMappings() verwendet die Klasse SPARQLQuery, um externe Wissensbasen nach ähnlichen Konzepten zu durchsuchen. Die eigentliche Abfrage erfolgt dabei über die entsprechenden SPARQL-Endpoints. In Listing A.5 ist die Funktion findClassIn SNOMEDCT(String strClass) dargestellt, welche die auf der BioPortal zur Verfügung stehende Version von SNOMED CT nach Klassen durchsucht, die den übergebenen String strClass in ihrem über die Relation skos:prefLabel verknüpften Label enthalten. Die entsprechende SPARQL-Abfrage wird zunächst als String formuliert (Z. 3-11) und anschließend an den entsprechenden Endpoint geschickt (Z. 14). Die gefundenen Klassen werden in einer Ergebnisliste zusammengestellt und an die Prozedur searchMappings() zurückgegeben (Z. 16-25).

Diese stellt anschließend jeden gefundenen Mapping-Vorschlag als Checkbox auf der Benutzeroberfläche dar. Durch Auswahl der Checkbox wir das entsprechende Tripel in die Wissensbasis gespeichert. Abb. A.3 zeigt die Darstellung von Mapping-Vorschlägen am Beispiel der Frage nach dem Gewicht der zu untersuchenden Probe. Durch die Auswahl der oberen beiden Vorschläge werden die entsprechenden Mappings übernommen und anschließend auch unter dem Tab „Current Mappings" angezeigt. Wie in Abb. A.4 ersichtlich, werden Mappings unter Verwendung der Relation `skos:closeMatch` realisiert. Diese verknüpft laut

SKOS-Spezifikation Konzepte miteinander, welche nicht vollständig identisch sind, aber ausreichend ähnlich, um in einigen Information-Retrieval-Anwendungen synonym verwendet werden zu können.[10] Folglich handelt es sich bei `skos:closeMatch` um eine symmetrische Relation. Wenn A ähnlich B ist und B ähnlich C, ist allerdings nicht sichergestellt, dass A damit auch ähnlich zu C ist. Um zu vermeiden, dass bei der Verkettung von mehreren Konzepten durch `skos:closeMatch` semantische Fehler entstehen, ist die Relation nicht als transitiv definiert.

[10]SKOS Mapping Properties: https://www.w3.org/TR/skos-reference/#mapping

5 Fazit und Ausblick

5.1 Ergebnisbeschreibung

Zusammenfassend lässt sich sagen, dass die im Zuge dieser Arbeit implementierte Software dazu in der Lage ist, pathologische Befundberichte im betrachteten Diskursbereich von Endometrium-, Haut- und Prostatakrebs dynamisch zu erstellen und in einem maschinenverständlichen Modell zu repräsentieren. Dabei werden alle in Abschnitt 2.3 festgelegten Programmanforderungen erfüllt.

Jeder erstellte Report ist in der Web Ontology Language (OWL) formuliert und entspricht damit einer Wissensbasis mit formal definierter Semantik (vgl. Abschnitt 4.1.1.1, speziell Abb. 4.3). Als inhaltliche Grundlage wurde sich für die entsprechenden Vorlagen der ICCR (International Collaboration on Cancer Reporting) entschieden, um mit den erstellten Dokumenten den Anforderungen möglichst global verbreiteter Expertenmeinungen gerecht zu werden. Die ebenfalls in OWL modellierten Vorlagen zur Erstellung eines solchen Berichts konnten softwaregesteuert im Zuge eines Ontology-Learning-Prozesses erstellt werden, der die Teilbereiche Ontology Enrichment und Ontology Population umfasst (vgl. 3.2). Dazu ist zunächst eine Report-Basis-Ontologie als Grundgerüst für die Repräsentation von pathologischen Befundberichten definiert worden (siehe Abschnitt 4.1.1.1). Der Vergleich der in Abb. 4.2 dargestellten Metrik dieses Grundgerüsts mit der Metrik der erstellten Report-Vorlage für Prostatakrebsbefundberichte in Abb. 5.1 verdeutlicht, dass der implementierte Ontology-Learning-Prozess 1056 Axiome automatisch generiert hat. Dies erspart den manuellen Modellierungsaufwand und reduziert damit Zeit und Kosten bei der Erstellung von Report-Vorlagen aus zukünftig spezifizierten ICCR-Datensätzen. Voraussetzung dafür ist, dass die bisherige Tabellen-Struktur der Vorlagen unverändert bleibt.

Der strukturelle Aufbau eines Berichts ist in der Report-Basis-Ontologie modelliert (Abschnitt 4.1.1.1, Abb. 4.1) und gibt die einzelnen verwendbaren Report-Elemente vor. Jede Frage und jede der vier beschriebenen Typen von Antwortmöglichkeiten wird als Instanz einer definierten Klasse erzeugt. So kann durch das Bereitstellen von Optionsfeldern oder

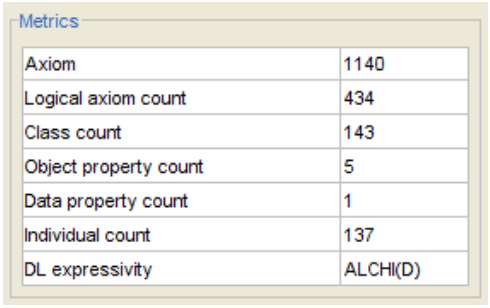

Abbildung 5.1: Metrik der erlernten Ontologie zur Erstellung eines Prostatektomie-Befundberichts

Auswahlkästchen die Vielfalt an konkreten Antworten (Instanzen) durch vorgegebene Möglichkeiten (Klassen) eingegrenzt werden, was eine einheitliche Struktur erzeugt. Einmal innerhalb der spezifischen Report-Typ-Ontologie definierte Fragen-Klassen werden bei jeder Neuerstellung eines Einzelberichts dieses Typs dokumentenübergreifend einheitlich repräsentiert. So kann jede Instanz dieser Klasse auf gleiche Weise dargestellt werden. Neben den einschränkenden Elementen müssen frei formulierbare Angaben an einigen Stellen weiterhin möglich bleiben. Die Variabilität der Antwort auf die Frage nach dem Gewicht oder der Größe einer Probe ist zu groß, als dass sie sich sinnvoll durch ein Optionsfeld oder Kontrollkästchen modellieren lässt. Deshalb konnte auf die Verwendung von Feldern zur Eingabe von Fließtext innerhalb eines Berichts nicht vollständig verzichtet werden. Da aber jedes Textfeld einer bestimmten Frage zugeordnet ist, kann zumindest dokumentenübergreifend referenziert werden, an welcher Stelle sich die Antwort auf diese Frage befindet. Bedingt durch das Speicherformat erhalten Reportelemente eine globale URI-Adresse, durch welche Sie dokumentenübergreifend referenziert werden können.

Dank der formal definierte Semantik der Report-Basis-Ontologie lassen sich die erstellten Berichte unter Verwendung des HermiT-Reasoners auf Vollständigkeit überprüfen. Die Definition darüber, welche Inhalte in einem zu betrachtenden Report-Typ empfohlen oder zwingend erforderlich sind, ist den spezifischen ICCR-Vorlagen entnommen. Dadurch, dass durch logische Schlussfolgerungen stets nur die Fragestellungen im Report auftauchen, die nach derzeitigem Stand der gegebenen Antworten relevant sind, bekommt der Prozess der Berichterstellung einen dynamischen Charakter.

Die Möglichkeit des Verknüpfens der durch Klassen repräsentierten Fragen und Antwortmöglichkeiten eines Befundberichts mit externen Wissensbasen ist ebenfalls prototypisch implementiert. Die Software ermöglicht Mappings der modellierten Klassen zu SNOMED CT, NCIT und PathLex. Für die Report-Vorlage zur Erstellung eines Prostatektomie-Berichts sind beispielhaft einige Mappings innerhalb der Ontologie definiert.

5.2 Ausblick

Da ein Ontology-Learning-Prozess stets eine Art Reverse-Engineering darstellt (Abschnitt 3.2.4), ist es empfehlenswert, zukünftige Report-Vorlagen direkt in Form einer entsprechenden Ontologie zu spezifizieren, um Fehler bei der Konzept-Modellierung zu vermeiden.

Da sich jedes Element eines Berichts über dessen URI referenzieren lässt, ist eine Einbindung dieser Adresse als Codierung in ein gemäß APSR-Spezifikation verfasstes Dokument denkbar. Das hat den Vorteil, dass das CDA-Format im medizinischen Bereich bereits weit verbreitet ist. Bei Verwendung dieser Berichtstruktur ohne Berücksichtigung der Report-Basis-Ontologie ist allerdings die Möglichkeit des Vollständigkeits-Checks durch logisches Schlussfolgern nicht gegeben (vgl. Abschnitt 3.1 Reasoning).

Die Implementierung der Suche nach Mapping-Vorschlägen erzielt für manche Report-Inhalte unübersichtlich viele oder gar keine Treffer. Hier ist eine Verbesserung der SPRQL-Abfragen denkbar, welche nicht nur die durch die Annotationrelation `skos:prefLabel` mit den potentiellen Ziel-Entitäten verbundenen Literale auf entsprechende Inhalte durchsucht, sondern darüber hinaus auch die semantischen Beziehungen der Klassen betrachtet, um möglichst viele und möglichst präzise definierbare Übereinstimmungen zu finden. Abb. A.1 zeigt Analyse-Methoden, welche zu diesem Zweck in das Programm integriert werden können.

Die Gesamtheit der erfüllten Anforderungsliste ist Grundvoraussetzung dafür, dass die erstellten Befundberichte computergesteuert nach definierbaren Inhalten durchsucht werden können. Gemeint ist hiermit nicht ausschließlich die textbasierte Suche nach Berichtsinhalten, sondern eine ontologiebasierte Abfrage einer Menge von Berichten nach unterschiedlichen definierbaren Kriterien. Denkbar ist beispielsweise die Suche nach allen Patienten, die als potentielle Kandidaten einer bestimmten Studie in Frage kommen. Hierbei können die vorgegebenen Studienvoraussetzungen mit den Angaben aus gespeicherten Befundberichten vergleichen und passende Kandidaten identifiziert werden. Eine solche Suche stellt derzeit auf Grund von unterschiedlichen Wortschätzen der Pathologen und synonym verwendeten Begriffen eine große Herausforderung dar, lässt sich jedoch über SPARQL-Anfragen an Berichte in beschriebener OWL-Struktur realisieren, da sich gesuchte Berichtsinhalte über ihre URI-Adressen dokumentenübergreifend einheitlich ansprechen lassen.

In Hinblick auf die zukünftige Arbeitsweise im medizinischen Umfeld kann die beschriebene Modellierung von Dokumenten dabei nützlich sein, das derzeitige papiergebundene Verfahren zur Erstellung von Befundberichten durch ein einheitliches digitales Verfahren abzulösen und die automatische Verarbeitung der enthaltenen Daten durch die Möglichkeiten ontologiebasierter Methoden zu erweitern.

A Anhang

A.1 Glossar

Tabelle A.1: Glossar

Begriff	Definition
APSR	APSR steht für „**A**natomic **P**athology **S**tructured **R**eport" und bezeichnet demnach einen strukturierten Befundbericht im Bereich der anatomischen Pathologie. Die Erweiterung „Anatomic Pathology Structured Reports (APSR)" des technischen Rahmenwerkes der IHE Anatomic Pathology beschreibt ein Anwendungsprofil für derartige strukturierte Befundberichte. Sie beinhaltet 22 auf dem Standard CDA R2 basierende APSR-Vorlagen (eine allgemeine, eine für Krebs im Allgemeinen und 20 organspezifische für die Dokumentation von Krebsbefunden).
CDA R2	Die **C**linical **D**ata **A**rchitecture **R**elease **2** beschreibt einen XML-basierten Standard, der die Struktur und Semantik von Dokumenten im Gesundheitswesen definiert, um deren Austausch zu vereinfachen. Der strukturierte Part eines CDA-Dokuments greift dabei auf Terminologien wie LOINC und SNOMED CT zurück. [**?**]
IHE	**I**ntegrating the **H**ealthcare **E**nterprise (IHE) bezeichnet eine Initiative von Anwendern und Herstellern, die das Ziel hat, die Art und Weise des Informationsaustausches zwischen Computersystemen im Gesundheitswesen zu verbessern. Die IHE fördert und koordiniert dabei den Einsatz von Standards wie DICOM und HL7 zur Verbesserung der Interoperabilität der Systeme und somit zur Unterstützung einer optimalen Patientenversorgung. [IHE]
IHTSDO	Die IHTSDO (**I**nternational **H**ealthcare **T**erminology **S**tandards **D**evelopment **O**rganization) ist eine Non-Profit-Organisation mit dem Ziel, globale Standards für Begriffe im Gesundheitsbereich zu bestimmen. Sie ist für die Pflege und Weiterentwicklung der Terminologie SNOMED CT verantwortlich. [IHT]
PathLex	PathLex ist eine Terminologie, welche den Wissensbereich der Beobachtungen und Probenentnahmeverfahren im Gebiet der anatomischen Pathologie umfasst. [DM11, S. 30]

Begriff	Definition
SNOMED CT	**S**ystematized **No**menclature of **Med**icine - **C**linical **T**erms (deutsch: Systematisierte Nomenklatur der Medizin - Klinische Begriffe) ist die weltweit umfassendste Terminologie im klinischen Gesundheitsbereich. [SNOb] Ursprünglich durch das **C**ollege of **A**merican **P**athologists (CAP) als systematisierte Nomenklatur der Pathologie (SNOP) entwickelt, wird sie heute von der IHTSDO gepflegt und herausgegeben. [SNOa]
TNM-Klassifikation	Die TNM-Klassifikation wird verwendet, um maligne (bösartige) Tumore in der Medizin in vordefinierte Stadien einzuteilen. Die drei wichtigsten Kategorien des TNM-Systems entsprechen den drei Buchstaben: T = Tumor, Ausdehnung und Verhalten des Primärtumors N = Nodus (englisch: Nodes = Lymphknoten), Fehlen bzw. Vorhandensein von regionären Lymphknotenmetastasen M = Metastasen, Fehlen bzw. Vorhandensein von Fernmetastasen

A.2 Namensräume

Tabelle A.2: Namensräume verwendeter formaler Sprachen

Präfix	Namensraum
:	http://www.semanticweb.org/ontologies/2017/03/AnatomicPathologyReportBasis# Im Zuge dieser Arbeit entwickelte Report-Basis-Ontologie. Da sie hier als Basis gilt, ist kein gesonderter Namensraum notwendig.
ncit:	http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#
owl:	http://www.w3.org/2002/07/owl#
pl:	http://www.semanticweb.org/david/ontologies/2013/0/pathLex.owl#
rdf:	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs:	http://www.w3.org/2000/01/rdf-schema#
sct:	http://purl.bioontology.org/ontology/SNOMEDCT/ (Namensraum im Kontext der BioPortal-Website)
skos:	http://www.w3.org/2004/02/skos/core#
xsd:	http://www.w3.org/2001/XMLSchema#

A.3 Abbildungen

Ontologie-Matching

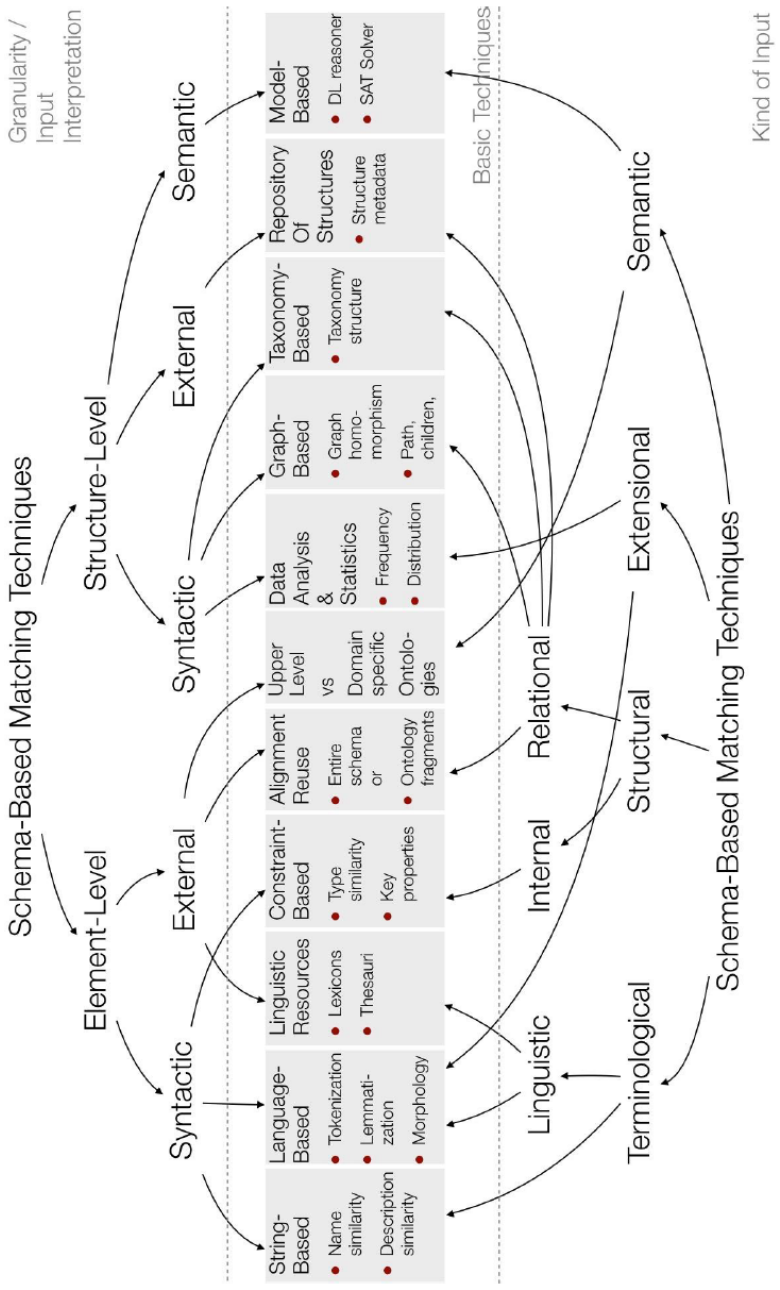

Abbildung A.1: Einteilung der Ontologie-Matching-Techniken nach Euzenat und Shvaiko [ES07]

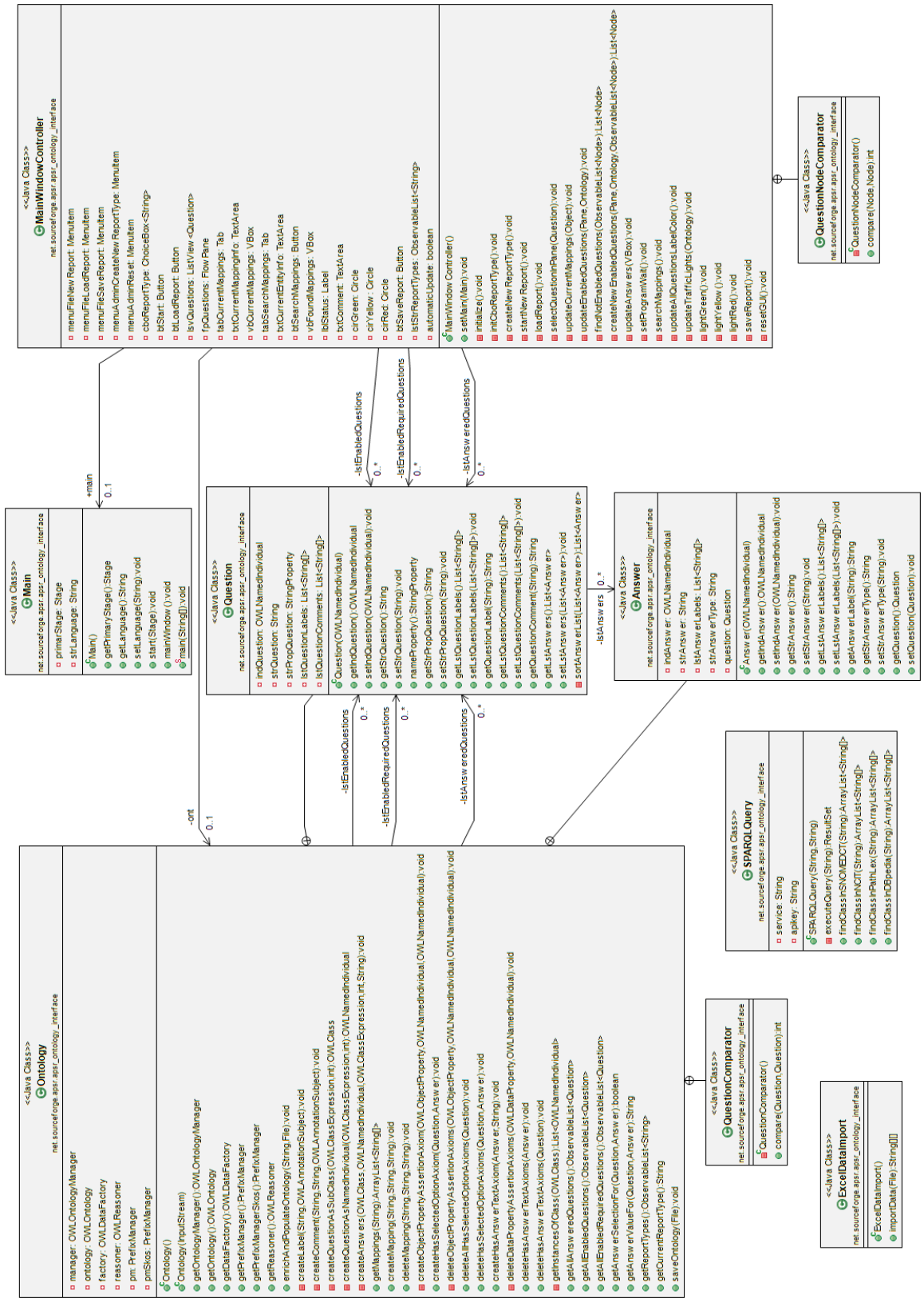

Abbildung A.2: Klassendiagramm des Softwaretools

Darstellung von Mappings auf der Benutzeroberfläche des Programms

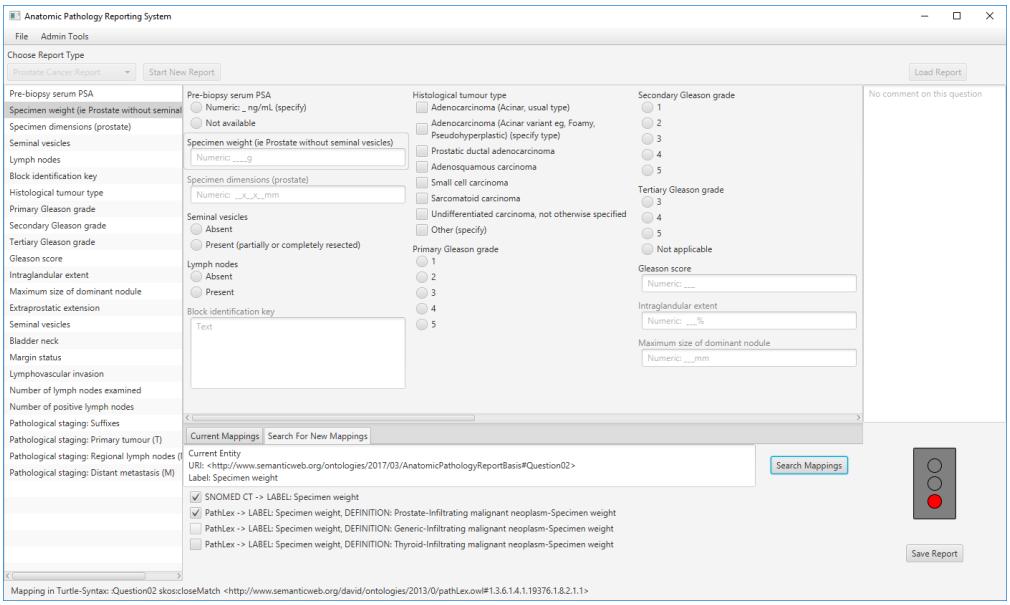

Abbildung A.3: Mapping-Vorschläge für die Frage nach dem Gewicht der Probe

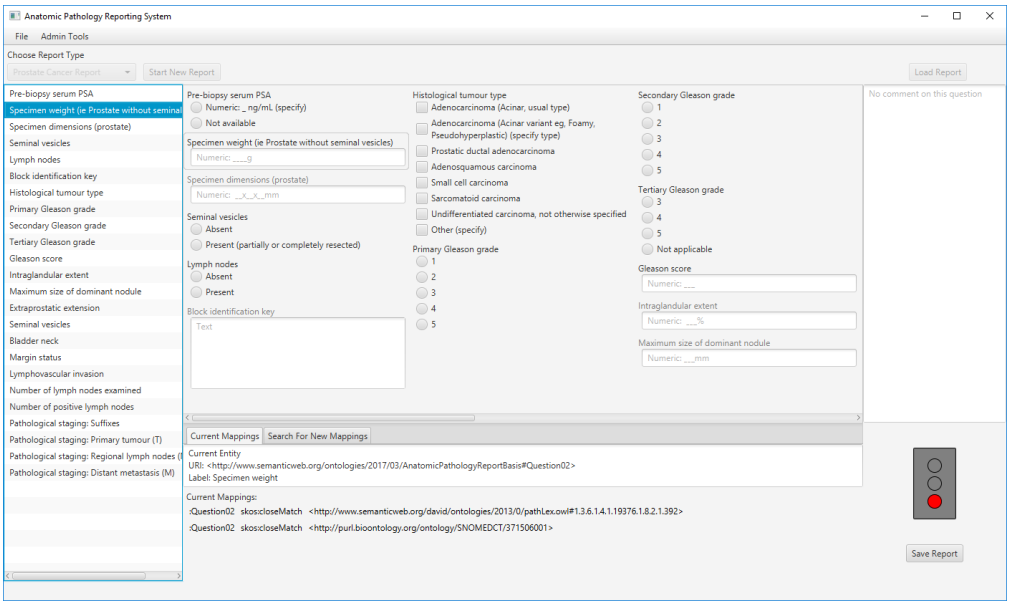

Abbildung A.4: Vorhandene Mappings für die Frage nach dem Gewicht der Probe

A.4 Quellcodes

Listing A.1: Auszug aus ICCR-Datensatz in Report-Ontologie APRBasis.owl

```
# 01
:indQuestion01 a owl:NamedIndividual , :Question01 ;
               :hasOption :indAnswer01.01  , :indAnswer01.02 ;
               rdfs:comment """Pre-biopsy serum PSA[...]"""@en;
               rdfs:label "Pre-biopsy serum PSA"@en .

:indAnswer01.01 a owl:NamedIndividual , :RadioButtonAnswer ;
                rdfs:label "Numeric: ___ng/mL (specify)"@en .
:indAnswer01.02 a owl:NamedIndividual , :RadioButtonAnswer ;
                rdfs:label "Not available"@en .

# 02
:indQuestion02 a owl:NamedIndividual , :Question02 ;
               :hasOption :indAnswer02 ;
               rdfs:label "Specimen weight (ie Pro[...]"@en .

:indAnswer02 a owl:NamedIndividual , :NumberAnswer ;
             rdfs:label "Numeric: ____g"@en .

# 03
:indQuestion03 a owl:NamedIndividual , :Question03 ;
               :hasOption :indAnswer03 ;
               rdfs:label "Specimen dimensions (prostate)"@en .

indAnswer03 a owl:NamedIndividual , :NumberAnswer ;
            rdfs:label "Numeric:  __x__x__mm"@en .

# 04
:indQuestion04 a owl:NamedIndividual , :Question04 ;
               :hasOption :indAnswer04.01 , :indAnswer04.02 ;
               rdfs:label "Seminal vesicles"@en .

:indAnswer04.01 a owl:NamedIndividual , :RadioButtonAnswer ;
                rdfs:label "Absent"@en .
:indAnswer04.02 a owl:NamedIndividual , :RadioButtonAnswer ;
                rdfs:label "Present (partially or [...]"@en .

# 05
:indQuestion05 a owl:NamedIndividual , :Question05 ;
               :hasOption :indAnswer05.01 , :indAnswer05.02 ;
               rdfs:label "Lymph nodes"@en .

:indAnswer05.01 a owl:NamedIndividual , :RadioButtonAnswer ;
                rdfs:label "Absent"@en .
:indAnswer05.02 a owl:NamedIndividual , :RadioButtonAnswer ;
                rdfs:label "Present"@en ;
                :enablesCandidate :indQuestion06 .

# 06
:indQuestion06 a owl:NamedIndividual , :Question06 ;
               :hasOption :indAnswer06.01 , :indAnswer06.02 , :
                  indAnswer06.03 ;
```

```
52                rdfs:label "Laterality"@en .
53
54 :indAnswer06.01 a owl:NamedIndividual , :RadioButtonAnswer ;
55                 rdfs:label "Left"@en .
56 :indAnswer06.02 a owl:NamedIndividual , :RadioButtonAnswer ;
57                 rdfs:label "Right"@en .
58 :indAnswer06.03 a owl:NamedIndividual , :RadioButtonAnswer ;
59                 rdfs:label "Bilateral"@en .
60
61 # Die Frage nach der spezifischen Befundberichtsvoralge ist von
      Beginn an eine
62 # aktivierte erforderliche Frage (:EnabledRequiredQuestion).
63 # In diesem Beispiel ist :prostateCancerReport die ausgewählte
      Antwort.
64 :pathologyCancerReportType a owl:NamedIndividual , :Question ;
65     :hasOption :endometrialCancerReport , :melanomaReport , :
         prostateCancerReport ;
66     :hasSelectedOption :prostateCancerReport .
67
68 # Zu Beginn der Erstellung eines Prostatektomie-Berichts ist
      Frage 3 bereits als
69 # aktivierte und die Fragen 1, 2, 4 und 5 als aktivierte
      erforderliche Fragen instanziiert.
70 :prostateCancerReport a owl:NamedIndividual , :RadioButtonAnswer,
      :SelectedAnswer ;
71     :enablesCandidate :indQuestion03 ;
72     :enablesRequiredCandidate :indQuestion01 , :indQuestion02 ,
73                               :indQuestion04 , :indQuestion05 .
```

Listing A.2: Funktion enrichAndPopulateOntology(String strReportType, File excelFilePath)

```
1  public void enrichAndPopulateOntology(String strReportType, File
       excelFilePath) throws Exception {
2      OWLOntologyManager manager = this.getOntologyManager();
3      OWLOntology ontology = this.getOntology();
4      OWLDataFactory factory = this.getDataFactory();
5      PrefixManager pm = this.getPrefixManager();
6      OWLReasoner reasoner = this.getReasoner();
7
8      //Get classes.
9      OWLClass clsQuestion = factory.getOWLClass(":Question", pm);
10     OWLClass clsAnsweredQuestion = factory.getOWLClass(":
          AnsweredQuestion", pm);
11     OWLClass clsAnswer = factory.getOWLClass(":Answer", pm);
12     OWLClass clsSelectedAnswer = factory.getOWLClass(":
          SelectedAnswer", pm);
13
14     //Get object properties.
15     OWLObjectProperty obpHasOption = factory.getOWLObjectProperty
          (":hasOption", pm);
16     OWLObjectProperty obpHasSelectedOption = factory.
          getOWLObjectProperty(":hasSelectedOption", pm);
17     OWLObjectProperty obpEnablesCandidate = factory.
          getOWLObjectProperty(":enablesCandidate", pm);
```

```
18      OWLObjectProperty obpEnablesRequiredCandidate = factory.
           getOWLObjectProperty(":enablesRequiredCandidate", pm);
19      [...]
20      //Excel data import
21      ExcelDataImport edi = new ExcelDataImport();
22      String[][] output = edi.importData(excelFilePath);
23      int entityNumber = 0;
24
25      for (int i=0; i<output.length; i++) {
26          //System.out.println(output[i][1]);
27          String strOpportunity = output[i][0];
28          String strElementName = output[i][1];
29          String strValues = output[i][2];
30          String strCommentary = output[i][3];
31          String strImplementationNotes = output[i][4];
32          [..]
33          entityNumber = entityNumber + 1;
34          //createSubClass of the class clsQuestion
35              OWLClass subClsQuestion = createQuestionAsSubClass(
                   clsQuestion, entityNumber);
36              //createNamedIndividual of the class clsQuestion
37              OWLNamedIndividual indQuestion =
                   createQuestionAsNamedIndividual(subClsQuestion,
                   entityNumber);
38
39              //create labels
40              createLabel(strElementName, subClsQuestion.getIRI());
41              createLabel(strElementName, indQuestion.getIRI());
42
43              //create comments
44              createComment(strCommentary, strImplementationNotes,
                   subClsQuestion.getIRI());
45              createComment(strCommentary, strImplementationNotes,
                   indQuestion.getIRI());
46
47              //Create object property assertion axiom
48                  if ((!strImplementationNotes.toLowerCase().
                       contains("record if")) [...]) {
49              //Create obpEnablesRequiredCandidate or
                   obpEnablesCandidate
50                  if (strOpportunity.toLowerCase().contains("
                       required")) {
51                  //Create object property assertion axiom.
52                  OWLObjectPropertyAssertionAxiom
                       opaEnablesRequiredCandidate = factory.
                       getOWLObjectPropertyAssertionAxiom(
                       obpEnablesRequiredCandidate, indReportChoice,
                       indQuestion);
53                  //Add object property assertion axiom to the
                       ontology.
54                                  manager.addAxiom(ontology,
                                       opaEnablesRequiredCandidate);
55              } else {
56                  //Create object property assertion axiom.
57                  OWLObjectPropertyAssertionAxiom
                       opaEnablesCandidate = factory.
```

```
                    getOWLObjectPropertyAssertionAxiom(
                    obpEnablesCandidate, indReportChoice,
                    indQuestion);
58                //Add object property assertion axiom to the
                    ontology.
59                manager.addAxiom(ontology, opaEnablesCandidate);
60            }
61        }
62
63        //create SubClasses as well as NamedIndividuals and
              relate individuals on property obpHasOption to the
              corresponding indQuestion (e.g. indQuestion01)
64        createAnswers(subClsQuestion, indQuestion, clsAnswer,
              entityNumber, strValues);
65
66        //Create object property assertion axiom on property
              obpEnablesCandidate.
67        if (((strImplementationNotes.toLowerCase().contains("
              record if")) [...])) {
68            String strIndQuestion = strImplementationNotes.
                  substring(strImplementationNotes.indexOf(" if ") +
                   4, strImplementationNotes.indexOf(" is "));
69            ArrayList<String> arrIndAnswers = new ArrayList<
                  String>();
70            [...]
71            //Create object property assertion axiom.
72                OWLObjectPropertyAssertionAxiom opa = factory.
                      getOWLObjectPropertyAssertionAxiom(
                      obpEnablesCandidate, indFound, indQuestion);
73                //Add object property assertion axiom to the
                      ontology.
74                manager.addAxiom(ontology, opa);
75                [...]
76        }
77    }
78 }
79 private OWLNamedIndividual createQuestionAsNamedIndividual(
       OWLClassExpression clsQuestion, int entityNumber) {
80    OWLOntologyManager manager = this.getOntologyManager();
81    OWLOntology ontology = this.getOntology();
82    OWLDataFactory factory = this.getDataFactory();
83    PrefixManager pm = this.getPrefixManager();
84
85    String strI = (entityNumber < 10) ? "0" + Integer.toString(
          entityNumber) : Integer.toString(entityNumber);
86    //Create named individual.
87    OWLNamedIndividual indQuestion = factory.
          getOWLNamedIndividual(":indQuestion" + strI, pm);
88    //Create class assertion axiom (rdf:type).
89    OWLClassAssertionAxiom ca = factory.getOWLClassAssertionAxiom
          (clsQuestion, indQuestion);
90    //Add class assertion axiom to the ontology.
91    manager.addAxiom(ontology, ca);
92    return indQuestion;
93 }
94 private void createLabel(String label, OWLAnnotationSubject
```

```
    entityIRI) {
95      OWLOntologyManager manager = this.getOntologyManager();
96      OWLOntology ontology = this.getOntology();
97      OWLDataFactory factory = this.getDataFactory();
98
99      OWLAnnotationProperty annProLabel = factory.
            getOWLAnnotationProperty(OWLRDFVocabulary.RDFS_LABEL.
            getIRI());
100     //Create label.
101     OWLAnnotation annLabel = factory.getOWLAnnotation(annProLabel
            , factory.getOWLLiteral(label, "en"));
102     //Create annotation assertion axiom.
103     OWLAnnotationAssertionAxiom aaLabel = factory.
            getOWLAnnotationAssertionAxiom(entityIRI, annLabel);
104     //Add annotation assertion axiom to the ontology.
105     manager.addAxiom(ontology, aaLabel);
106 }
```

Listing A.3: Funktion updateTrafficLights(Ontology ont)

```
1  //enabled questions
2  private ObservableList<Question> lstEnabledQuestions =
       FXCollections.observableArrayList();
3  //enabled required questions
4  private ObservableList<Question> lstEnabledRequiredQuestions =
       FXCollections.observableArrayList();
5  //answered questions
6  private ObservableList<Question> lstAnsweredQuestions =
       FXCollections.observableArrayList();
7
8  private void updateTrafficLights(Ontology ont) {
9       lstEnabledQuestions = ont.getAllEnabledQuestions();
10      lstEnabledRequiredQuestions = ont.
            getAllEnabledRequiredQuestions();
11      lstAnsweredQuestions = ont.getAllAnsweredQuestions();
12      List<Question> lstOpenQuestions = FXCollections.
            observableArrayList();
13      List<Question> lstOpenRequiredQuestions = FXCollections.
            observableArrayList();
14
15      for (Question qE : lstEnabledQuestions) {
16          //write all EnabledQuestions in list OpenQuestions
17          lstOpenQuestions.add(qE);
18          //check whether there are open EnabledQuestions or not
19          for (Question qA : lstAnsweredQuestions) {
20              if (qE.getIndQuestion().equals(qA.getIndQuestion()))
                    {
21                  lstOpenQuestions.remove(qE);
22              }
23          }
24      }
25      if (lstOpenQuestions.isEmpty()) {
26          //the report is a complete report (in terms of content)
27          lightGreen();
28      } else {
29          for (Question qER : lstEnabledRequiredQuestions) {
```

```
30              //write EnabledRequiredQuestions in list
                    OpenRequiredQuestions
31              lstOpenRequiredQuestions.add(qER);
32              //check whether there are open
                    EnabledRequiredQuestions or not
33              for (Question qA : lstAnsweredQuestions) {
34                  if (qER.getIndQuestion().equals(qA.getIndQuestion
                        ())) {
35                      lstOpenRequiredQuestions.remove(qER);
36                  }
37              }
38          }
39          if (lstOpenRequiredQuestions.isEmpty()) {
40              //the report is a sufficient report (in terms of
                    content)
41              lightYellow();
42          } else {
43              //the report is an insufficient report (in terms of
                    content)
44              lightRed();
45          }
46      }
47  }
48  private void lightGreen(){
49      cirGreen.setFill(Color.GREEN);
50      cirYellow.setFill(Color.GREY);
51      cirRed.setFill(Color.GREY);
52  }
53  private void lightYellow(){
54      cirGreen.setFill(Color.GREY);
55      cirYellow.setFill(Color.YELLOW);
56      cirRed.setFill(Color.GREY);
57  }
58  private void lightRed(){
59      cirGreen.setFill(Color.GREY);
60      cirYellow.setFill(Color.GREY);
61      cirRed.setFill(Color.RED);
62  }
```

Listing A.4: Funktion getInstancesOfClass(OWLClass cls)

```
1  private List<OWLNamedIndividual> getInstancesOfClass(OWLClass cls
       ) {
2      List<OWLNamedIndividual> lstIndividuals = new ArrayList<
           OWLNamedIndividual>();
3      OWLReasoner reasoner = this.getReasoner();
4      reasoner.flush();
5
6      //Asking for the instances of the class cls.
7      NodeSet<OWLNamedIndividual> individualsNodeSet = reasoner.
           getInstances(cls, false);
8      Set<OWLNamedIndividual> individuals = individualsNodeSet.
           getFlattened();
9      if (individuals.toString() != "[]") {
10         for (OWLNamedIndividual ind : individuals) {
11             if (!lstIndividuals.contains(ind)) {
12                 lstIndividuals.add(ind);
13             }
14         }
15     }
16     //sort list alphabetically (a...z)
17     Collections.sort(lstIndividuals);
18     return lstIndividuals;
19 }
```

Listing A.5: Funktion findClassInSNOMEDCT(String strClass)

```
1  public ArrayList<String[]> findClassInSNOMEDCT(String strClass) {
2      ArrayList<String[]> allResults = new ArrayList<String[]>();
3      String strQuery = "PREFIX rdfs: <http://www.w3.org/2000/01/
           rdf-schema#> \n" +
4          "PREFIX skos: <http://www.w3.org/2004/02/skos/core#> \n"+
5          "SELECT DISTINCT ?cls ?prefLab \n" +
6          "FROM <http://bioportal.bioontology.org/ontologies/
               SNOMEDCT> \n" +
7          "WHERE { \n" +
8              "?cls skos:prefLabel ?prefLab . \n" +
9              "FILTER regex(?prefLab, '" + strClass.trim() + "', 'i
                   ' ) \n" +
10         "} \nORDER BY ?prefLab \n" +
11         "LIMIT 100";
12
13     Query query = QueryFactory.create(strQuery) ;
14     QueryEngineHTTP qexec = QueryExecutionFactory.
           createServiceRequest("http://sparql.bioontology.org/sparql
           ", query);
15     ResultSet results = qexec.execSelect();
16     while(results.hasNext()){
17         QuerySolution soln = results.nextSolution();
18         RDFNode uri = soln.get("cls");
19         RDFNode lab = soln.get("prefLab");
20         String[] str = {uri.toString(), lab.toString()};
21         allResults.add(str);
22     }
23     return allResults;
24 }
```

Abbildungsverzeichnis

Tabellenverzeichnis

Programm-Listings

Literaturverzeichnis

[AAH+09] Agirre, Eneko ; Alfonseca, Enrique ; Hall, Keith ; Kravalova, Jana ; Pasca, Marius ; Soroa, Aitor: A Study on Similarity and Relatedness Using Distributional and WordNet-based Approaches. In: *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2009

[AH08] Allemang, Dean ; Hendler, James: *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 2008

[Als87] Alshawi, Hiyan: Processing Dictionary Definitions with Phrasal Pattern Hierarchies. In: *Comput. Linguist.; 13(3-4): 195-202* (1987)

[BBLPC14] Beckett, David ; Berners-Lee, Tim ; Prud'hommeaux, Eric ; Carothers, Gavin: *RDF 1.1 Turtle - Terse RDF Triple Language*. `https://www.w3.org/TR/turtle`. Version: Februar 2014

[BCF+08] Buitelaar, Paul ; Cimiano, Philipp ; Frank, Anette ; Hartung, Matthias ; Racioppa, Stefania: Ontology-based information extraction and integration from heterogeneous data sources. In: *Int. J. Hum.-Comput. Stud.; 66(11): 759-788* (2008)

[BCW03] *Kapitel* Background and foreground knowledge in dynamic ontology construction. In: Brewster, Christopher ; Ciravegna, Fabio ; Wilks, Yorick: *Proceedings of the Semantic Web Workshop*. 2003

[Bec03] Beckermann, Ansgar: *Einführung in die Logik*. De Gruyter, 2003 (De-Gruyter-Studienbuch). – ISBN 9783110179651

[BGM05] Brank, Janez ; Grobelnik, Marko ; Mladenić, Dunja: Proceedings of the Conference on Data Mining and Data Warehouses (SiKDD). In: *Proceedings of the Conference on Data Mining and Data Warehouses (SiKDD)*, 2005

[Bor97] Borst, Willem N.: *Construction of Engineering Ontologies for Knowledge Sharing and Reuse*. University of Twente, Enschede. http://eprints.eemcs.utwente.nl/17377/01/t0000004.pdf. Version: 1997

[BOS04] Buitelaar, Paul ; Olejnik, Daniel ; Sintek, Michael: A Protégé Plug-In for Ontology Extraction from Text Based on Linguistic Analysis. In: *Proceedings of the 1st European Semantic Web Symposium (ESWS)*, 2004, S. 31–44

[BS01] Buitelaar, Paul ; Sacaleanu, Bogdan: Ranking and Selecting Synsets by Domain Relevance. In: *Proceedings NAACL WordNet Workshop*, 2001

[BSW06] Boer, Viktor de ; Someren, Maarten van ; Wielinga, Bob J.: Relation Instantiation for Ontology Population Using the Web. In: *KI 2006: Advances in Artificial Intelligence - Proceedings of the 29th Annual German Conference on AI*, 2006, S. 202–213

[BYRN99] Baeza-Yates, Ricardo ; Ribeiro-Neto, Berthier: *Modern Information Retrieval*. Addison-Wesley, 1999

[Cal84] Calzolari, Nicoletta: Detecting Patterns in a Lexical Data Base. In: *Proceedings of the 10th International Conference on Computational Linguistics and 22Nd Annual Meeting on Association for Computational Linguistics*, 1984

[CHR06] Cimiano, Philipp ; Hartung, Matthias ; Ratsch, Esther: Finding the appropriate generalization level for binary ontological relations extracted from the Genia corpus. In: *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, 2006

[CHS05] Cimiano, Philipp ; Hotho ndreas ; Staab, Steffen: Learning Concept Hierarchies from Text Corpora Using Formal Concept Analysis. In: *J. Artif. Int. Res.; 24(1): 305-339* (2005)

[Cim06] Cimiano, Philipp: *Ontology Learning and Population from Text: Algorithms, Evaluation and Applications*. Springer, 2006

[CMSV09] Cimiano, Philipp ; Mädche, Alexander ; Staab, Steffen ; Völker, Johanna: *Ontology Learning*. https://userpages.uni-koblenz.de/~staab/Research/Publications/2009/handbookEdition2/ontology-learning-handbook2.pdf. Version: 2009

[DAB+06] Dolin, Robert H. ; Alschuler, Liora ; Boyer, Sandy ; Beebe, Calvin ; Behlen, Fred M. ; Biron, Paul V. ; (Shvo), Amnon S.: *HL7 Clinical Document Architecture, Release 2*. J Am Med Inform Assoc.; 13(1): 30-39, 2006

[Dan10] Daniel, Christel: *PathLex, a single lexicon in the Anatomic Pathology domain.* IHE Anatomic Pathology. `http://www.hl7.org/documentcenter/public/wg/anatomicpath/20100831_PathLex.docx`. Version: 2010

[dFE08] d'Amato, Claudia ; Fanizzi, Nicola ; Esposito, Floriana: Query Answering and Ontology Population: An Inductive Approach. In: *Proceedings of the 5th European Semantic Web Conference (ESWC)*, 2008, S. 288–302

[DM11] Daniel, Christel ; Macary, Francois: *IHE Anatomic Pathology Technical Framework Supplement - Anatomic Pathology Structured Reports (APSR) - Trial Implementation, Rev. 1.1.* `http://www.ihe.net/Technical_Framework/upload/IHE_PAT_Suppl_APSR_Rev1-1_TI_2011_03_31.pdf`. `http://www.ihe.net/Technical_Framework/upload/IHE_PAT_Suppl_APSR_Rev1-1_TI_2011_03_31.pdf`. Version: März 2011

[EDH+13] Ellis, D. ; Dvorak, J. ; Hirschowitz, L. ; Judge, M. ; Kwiatkowski, A. ; Srigley, J. ; Washington, M.K. ; Wells, M.: The international collaboration on cancer reporting (ICCR): development of evidence-based core data sets for pathology cancer reporting. In: *Pathology 45(1), S9* (2013)

[ES07] Euzenat, Jérôme ; Shvaiko, Pavel: *Ontology matching.* Heidelberg : Springer, 2007

[FAM00] Frantzi, Katerina ; Ananiadou, Sophia ; Mima, Hideki: *Automatic Recognition of Multi-Word Terms: the C-value/NC-value Method.* International Journal on Digital Libraries, 3(2):115-130, 2000

[Feh13] Fehr, Elfriede: *Semantik von Programmiersprachen.* Springer Berlin Heidelberg, 2013 (Studienreihe Informatik). – ISBN 9783642702716

[Fel98] Fellbaum, Christiane: *WordNet: An Electronic Lexical Database.* MIT Press, 1998

[Fil12] Fill, Alwin: *Linguistische Promenade - eine vergnügliche Wanderung durch die Sprachwissenschaft von Platon zu Chomsky.* LIT Verlag Münster, 2012

[Fir57] Firth, John R.: A synopsis of linguistic theory 1930-55. In: *Studies in Linguistic Analysis* (1957)

[Fis87] Fisher, Douglas H.: Knowledge acquisition via incremental conceptual clustering. In: *Mach. Learn.; 2(2): 139-172* (1987)

[FKM+06] Fokoue, Achille ; Kershenbaum, Aaron ; Ma, Li ; Schonberg, Edith ; Srinivas, Kavitha: *The Summary Abox: Cutting Ontologies Down to*

Size. http://iswc2006.semanticweb.org/items/Kershenbaum2006qo.pdf. Version: 2006

[GHM+14] Glimm, Birte ; Horrocks, Ian ; Motik, Boris ; Stoilos, Giorgos ; Wang, Zhe: HermiT: An OWL 2 Reasoner. In: *Journal of Automated Reasoning 53(3)* (2014)

[GK05] Geleijnse, Gijs ; Korst, Jan: Automatic Ontology Population by Googling. In: *Proceedings of the 17th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC)*, 2005

[Gru93] Gruber, Thomas R.: *A Translation Approach to Portable Ontology Specifications*. Knowledge Acquisition, 5(2):199-220. http://tomgruber.org/writing/ontolingua-kaj-1993.pdf. Version: 1992, überarbeitet 1993

[GW99] Ganter, Bernhard ; Wille, Rudolf: *Formal Concept Analysis - Mathematical Foundations*. Springer, 1999

[Har54] Harris, Zellig S.: Distributional Structure. In: *Word* 10 (1954), Nr. 2-3, S. 146–162

[HB11] Horridge, Matthew ; Bechhofer, Sean: The OWL API: A Java API for OWL Ontologies. In: *Semant. web; 2(1):11-21* (2011)

[Hea92] Hearst, Marti A.: Automatic Acquisition of Hyponyms from Large Text Corpora. In: *Proceedings of the 14th Conference on Computational Linguistics - Volume 2*, 1992

[HF97] Hamp, Birgit ; Feldweg, Helmut: GermaNet - a Lexical-Semantic Net for German. In: *Proceedings of the ACL workshop Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*, 1997

[HLA09] Hellmann, Sebastian ; Lehmann, Jens ; Auer, Sören: Learning of OWL Class Descriptions on Very Large Knowledge Bases. In: *International journal on Semantic Web and information systems.; 5(2): 25-48* (2009)

[IHE] *Integrating the Healthcare Enterprise (IHE)*. IHE. http://www.ihe.net. – Stand vom 15.02.2016

[IHT] *IHTSDO*. IHTSDO. http://www.ihtsdo.org/about-ihtsdo. – Stand vom 19.02.2016

[Jac01] Jaccard, Paul: Distribution de la flore alpine dans le bassin des Dranses et dans quelques régions voisines. In: *Bulletin de la Société Vaudoise des Sciences Naturelles 37, 241-272* (1901)

[JACH02] Jonyer, Istvan ; Arlington, Arlington ; Cook, Diane J. ; Holder, Lawrence B.: Graph-based Hierarchical Conceptual Clustering. In: *J. Mach. Learn. Res.; 2: 19-43* (2002)

[KVM00] Kietz, Jörg-Uwe ; Volz, Raphael ; Maedche, Alexander: Extracting a Domain-Specific Ontology from a Corporate Intranet. In: *Proceedings of the 2nd Learning Language in Logic (LLL) Workshop, Lissabon*, 2000

[LB10] Lehmann, Jens ; Bühmann, Lorenz: ORE - a Tool for Repairing and Enriching Knowledge Bases. In: *Proceedings of the 9th International Semantic Web Conference on The Semantic Web - Volume Part II*, 2010

[Lee99] Lee, Lillian: Measures of Distributional Similarity. In: *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics*, 1999

[LV14] Lehmann, Jens ; Völker, Johanna: An Introduction to Ontology Learning. Version: 2014. `http://jens-lehmann.org/files/2014/pol_introduction.pdf`. In: *Perspectives on Ontology Learning*. AKA / IOS Press, 2014, ix-xvi

[MS00] Maedche, Alexander ; Staab, Steffen: *Discovering Conceptual Relations from Text*. 2000

[MS01] Maedche, Alexander ; Staab, Steffen: Ontology Learning for the Semantic Web. In: *IEEE Intelligent Systems* 16 (2001), S. 72–79

[NM01] Noy, Natalya F. ; McGuinness, Deborah L.: Ontology Development 101: A Guide to Creating Your First Ontology. Version: 2001. `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.136.5085&rep=rep1&type=pdf`. 2001. – Forschungsbericht

[OLA+13] Oemig, Frank ; Lang, Stefan ; Altmann, Udo ; Amenda, Esther ; Fontein, Silke ; Lang, Stefan ; Schmitz, Matthias ; Schütze, Bernd ; Thiele, Claas: *Übermittlung onkologischer Daten auf der Basis von CDA R2*. `http://wiki.hl7.de/index.php?title=IG:%C3%9Cbermittlung_onkologischer_Daten`. Version: 2013

[OR23] Ogden, C.K. ; Richards, I. A.: *The Meaning of Meaning: A Study of the Influence of Language Upon Thought and of the Science of Symbolism*. 1923 `http://s-f-walker.org.uk/pubsebooks/pdfs/ogden-richards-meaning-all.pdf`

[OWL04] OWL Working Group, W3C: *OWL Web Ontology Language: Guide*. W3C Re-

commendation, 2004 https://www.w3.org/TR/owl2-overview/. – Available at http://www.w3.org/TR/owl2-overview/

[OWL12] OWL Working Group, W3C: *OWL 2 Web Ontology Language: Document Overview.* W3C Recommendation, 2012 https://www.w3.org/TR/owl2-overview/. – Available at http://www.w3.org/TR/owl2-overview/

[PFS+05] Predoiu, Livia ; Feier, Cristina ; Scharffe, Francois ; Bruijn, Jos de ; Martin-Recuerda, Francisco ; Manov, Dimitar ; Ehrig, Marc: *State-of-the-art survey on Ontology Merging and Aligning V2.* http://www.sekt-project.com/rd/deliverables/wp04/sekt-d-4-2-2-SOA%20Survey%20on%20Ontology%20Merging%20and%20Aligning%20v2.pdf. Version: 2005

[PPM04] Pedersen, Ted ; Patwardhan, Siddharth ; Michelizzi, Jason: WordNet::Similarity: Measuring the Relatedness of Concepts. In: *Demonstration Papers at HLT-NAACL*, 2004

[RMJ+06] Rosenbloom, S. T. ; Miller, Randolph A. ; Johnson, Kevin B. ; Elkin, Peter L. ; Brown, Steven H.: *Interface Terminologies: Facilitating Direct Entry of Clinical Data into Electronic Health Record Systems.* J Am Med Inform Assoc.; 13(3): 277-288, 2006

[Sac15a] Sack, Harald: *Knowledge Engineering with Semantic Web Technologies - Lecture 5: Ontological Engineering - 5.3 Ontology Learning.* https://open.hpi.de/files/52599792-6534-44ab-af5e-3c7b2058a629. Version: 2015

[Sac15b] Sack, Harald: *Knowledge Engineering with Semantic Web Technologies - Lecture 5: Ontological Engineering - 5.4 Ontology Alignment.* https://open.hpi.de/files/a08c612a-5384-412a-b102-6ca9b8c1cd26. Version: 2015

[Sac16] Sack, Harald: *Linked Data Engineering - Lecture 2: RDF and RDFS - 2.6 Logical Inference with RDFS.* 2016

[SAMK05] Spasic, Irena ; Ananiadou, Sophia ; McNaught, John ; Kumar, Anand: *Text mining and ontologies in biomedicine: Making sense of raw text.* Brief Bioinform.; 6(3): 239-251. https://www.ncbi.nlm.nih.gov/pubmed/16212772. Version: 2005

[SB05] Schutz, Alexander ; Buitelaar, Paul: RelExt: A Tool for Relation Extraction from Text in Ontology Extension. In: *Proceedings of the 4th International Conference on The Semantic Web*, 2005

[SCE15] Schlegel, Daniel R. ; Crowner, Chris ; Elkin, Peter L.: Automatically Expanding the Synonym Set of SNOMED CT using Wikipedia. In: *Stud Health Technol Inform.; 216:619-23* (2015)

[SHB05] Sistrom, Chris L. ; Honeyman-Buck, Janice: *Free text versus structured format: information transfer efficiency of radiology reports.* AJR Am J Roentgenol.;185(3):804-12. `https://www.ncbi.nlm.nih.gov/pubmed/16120938`. Version: September 2005

[SM03] Schadow, Gunther ; McDonald, Clement J.: *Extracting Structured Information from Free Text Pathology Reports.* AMIA Annu Symp Proc.; 584-588. `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1480213/`. Version: 2003

[SNOa] *History Of SNOMED CT.* IHTSDO. `http://www.ihtsdo.org/snomed-ct/what-is-snomed-ct/history-of-snomed-ct`. – Stand vom 18.02.2016

[SNOb] *SNOMED CT - The Global Language of Healthcare.* IHTSDO. `http://www.ihtsdo.org/snomed-ct`. – Stand vom 17.02.2016

[Sow00] Sowa, John F.: *Ontology, Metadata, and Semiotics.* http://www.jfsowa.com/ontology/ontometa.htm. `http://www.jfsowa.com/ontology/ontometa.htm`. Version: 2000. – Stand vom 08.01.2017

[SPNS09] Stenzhorn, Holger ; Pacheco, Edson J. ; Nohama, Percy ; Schulz, Stefan: *Automatic Mapping of Clinical Documentation to SNOMED CT.* Stud Health Technol Inform.; 150: 228-232, 2009

[Sta16] Stangl, Werner: *Ontologie.* Lexikon für Psychologie und Pädagogik. `http://lexikon.stangl.eu/15645/ontologie/`. Version: Stand vom 16.11.2016

[VHC07] *Kapitel* Acquisition of OWL DL Axioms from Lexical Resources. In: Völker, Johanna ; Hitzler, Pascal ; Cimiano, Philipp: *The Semantic Web: Research and Applications - 4th European Semantic Web Conference, ESWC.* Springer Berlin Heidelberg, 2007, S. 670–685

[VNCN06] Velardi, Paola ; Navigli, Roberto ; Cucchiarelli, Alessandro ; Neri, Francesca: Evaluation of OntoLearn, a methodology for automatic population of domain ontologies. In: *Ontology Learning from Text: Methods, Applications and Evaluation.* IOS Press, 2006

[VOS03] Volz, Raphael ; Oberle, Daniel ; Studer, Rudi: *Views for Light-Weight Web Ontologies.* 2003

[Vra09] *Kapitel* Ontology Evaluation. In: Vrandečić, Denny: *Handbook on Ontologies, International Handbooks on Information Systems*. Springer, 2009, S. 293–313

[VVSH07] Völker, Johanna ; Vrandečić, Denny ; Sure, York ; Hotho, Andreas: Learning disjointness. In: *Proceedings oh the 4th European Semantic Web Conference*, Springer, 2007, S. 175–189

[Wid03] Widdows, Dominic: Unsupervised Methods for Developing Taxonomies by Combining Syntactic and Statistical Information. In: *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, 2003, S. 197–204

[Wik16] Wikipedia: *Klassifizierung*. `https://de.wikipedia.org/w/index.php?title=Klassifizierung&oldid=158771834`. Version: Letzte Revision: 2016

[Wil08] William Sealy Gosset (alias Student): The Probable Error of a Mean. In: *Biometrika*, 1908

[WKR10] Witte, René ; Khamis, Ninus ; Rilling, Juergen: Flexible Ontology Population from Text: The OwlExporter. In: *Int. Conf. on Language Resources and Evaluation (LREC)*, 2010

[WPT+09] Wei, Xing ; Peng, Fuchun ; Tseng, Huihsin ; Lu, Yumao ; Dumoulin, Benoit: Context Sensitive Synonym Discovery for Web Search Queries. In: *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, 2009

[XLLL08] Xu, Kaiquan ; Liao, Stephen S. ; Lau, Raymond Y. K. ; Liao, Lejian: Knowledge Acquisition With Supervised Ontology Population. In: *Proceedings of Pacific Asia Conference on Information Systems (PACIS)*, 2008

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit zum Thema

Konzeption und prototypische Implementierung eines Softwaretools zur dynamischen Erstellung von Befundberichten mit Hilfe ontologiebasierter Methoden

vollkommen selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, sowie Zitate kenntlich gemacht habe. Die Arbeit wurde in dieser oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Brandenburg an der Havel, den 02.03.2017

_____

Unterschrift